Dual System Encryption Framework in Prime-Order Groups

Nuttapong Attrapadung
AIST, Japan
n.attrapadung@aist.go.jp

Abstract

We propose a new generic framework for achieving fully secure attribute based encryption (ABE) in *prime-order* bilinear groups. It is generic in the sense that it can be applied to ABE for *arbitrary* predicate. All previously available frameworks that are generic in this sense are given only in *composite-order* bilinear groups. These consist of the frameworks proposed by Wee (TCC'14) and Attrapadung (Eurocrypt'14). Both frameworks provide abstractions of dual-system encryption techniques introduced by Waters (Crypto'09). Our framework can be considered as a prime-order version of Attrapadung's framework and works in a similar manner: it relies on a main component called *pair encodings*, and it generically compiles any secure pair encoding scheme for a predicate in consideration to a fully secure ABE scheme for that predicate. One feature of our new compiler is that although the resulting ABE schemes will be newly defined in prime-order groups, we require essentially the same security notions of pair encodings as before. Beside the security of pair encodings, our framework assumes only the Matrix Diffie-Hellman assumption, introduced by Escala *et al.* (Crypto'13), which is a weak assumption that includes the Decisional Linear assumption as a special case.

As for its applications, we can plug in available pair encoding schemes and automatically obtain the first fully secure ABE realizations in prime-order groups for predicates of which only fully secure schemes in composite-order groups were known. These include ABE for regular languages, ABE for monotone span programs (and hence Boolean formulae) with short ciphertexts or keys, and completely unbounded ABE for monotone span programs.

As a side result, we establish the first generic implication from ABE for monotone span programs to ABE for branching programs. Consequently, we obtain fully-secure ABE for branching programs in some new variants, namely, unbounded, short-ciphertext, and short-key variants. Previous ABE schemes for branching programs are bounded and require linear-size ciphertexts and keys.

Keywords. Dual system encryption, Prime-order bilinear groups, Attribute-based encryption, Full security, Generic framework.

1 Introduction

Dual system encryption techniques introduced by Waters [49] have been successful approaches for constructing fully secure attribute based encryption (ABE) systems that are based on bilinear groups. Despite being versatile as they can be applied to ABE systems for many predicates, until only recently, however, there were no known generic frameworks that can use the techniques in a black-box and modular manner. Wee [51] and Attrapadung [1] recently proposed such generic frameworks that abstract the dual system techniques by decoupling what seem to be essential underlying primitives and characterizing their sufficient conditions so as to obtain fully-secure ABE automatically via generic constructions. However, their frameworks are inherently constructed over bilinear groups of *composite-order*. Although composite-order bilinear groups are more intuitive to work with, especially in the case of dual system techniques, *prime-order* bilinear groups are more preferable as they provide more efficient and compact instantiations. This has been motivated already in a line of research [18, 41, 39, 47, 33, 29]. In this work, our goal is to propose a generic framework for dual-system encryption in prime-order groups.

Both generic frameworks [51, 1] work similarly but with the difference that the latter [1] captures also dual system techniques with *computational approaches*, which are generalized from techniques implicitly used in the ABE of Lewko and Waters [37]. (The former [51] only captures the traditional dual systems, which implicitly use information-theoretic approaches). Using computational approaches, the framework of [1] is able to obtain the first fully secure schemes for many ABE primitives for which only selectively secure constructions were known before, including ABE for regular languages [50], ABE for Boolean formulae[1] with constant-size ciphertexts [5], and (completely) unbounded ABE for Boolean formulae [36, 45]. Moreover, Attrapadung and Yamada [6] recently show that, within the framework of [1], we can generically convert ABE to its *dual* scheme; that is, key-policy (KP) type to ciphertext-policy (CP) type, and vice versa. They also show a conversion to its *dual-policy* [3] type (that is, the conjunctive of both key-policy and ciphertext-policy). Many instantiations were then achieved in [6], including the first CP-ABE for Boolean formulae with constant-size keys. Due to its generality, we choose to build upon the work of [1].

1.1 Our Contributions

New Framework. We present a new generic framework for achieving fully secure ABE in *prime-order groups*. It is generic in the sense that it can be applied to ABE for *arbitrary* predicate. Our framework extends the framework of [1], which was constructed in composite-order groups, and works in a similar manner as follows. First, the main component is a primitive called *pair encoding* scheme defined for a predicate. Second, we provide a generic construction that compiles any secure pair encoding scheme for a predicate R to a fully secure ABE scheme for the same predicate R. The *security* requirement for the underlying encoding scheme is exactly the same as that in the framework of [1]. On the other hand, we restrict the *syntax* of encodings into a class we call *regular encodings*, via some simple requirements. This confinement, however, seems natural and does not affect any concrete pair encoding schemes proposed so far [51, 1, 6]. Beside the security of pair encodings, our framework assumes only the Matrix Diffie-Hellman assumption, given by Escala *et al.* (Crypto'13). This assumption

[1]Or more precisely, ABE for monotone span programs. We will use both terms interchangeably.

can be considered as a framework of assumptions, and our scheme can rely on any instance of them, including the most standard one, namely, the Decisional Linear assumption.

New Instantiations. By using exactly the same encoding instantiations in [1, 6], we thus automatically obtain fully secure ABE schemes, *for the first time in prime-order groups*, for various predicates. These include the first fully-secure schemes for

- KP-ABE and CP-ABE for regular languages,
- KP-ABE for Boolean formulae with constant-size ciphertexts,
- CP-ABE for Boolean formulae with constant-size keys,
- Completely unbounded KP-ABE and CP-ABE for Boolean formulae,

all in prime-order groups, which should admit better efficiency and compactness. The assumptions for respective encodings are the same as those in [1] (albeit with a minor syntactic change to prime-order groups). Moreover, via the dual-policy conversion of [6], we also obtain their respective dual-policy variants. These instantiations are explained in §6 and summarized in Table 2 (with comparisons to composite-order schemes in Table 1).

New Predicates: Unbounded ABE for Branching Programs, and More. Besides the above new instantiations for existing predicates, we also consider new predicates. More precisely, we propose some new variants of ABE for Branching Program (ABE-BP), namely, *unbounded*, *short-ciphertext*, and *short-key* variants. Unbounded ABE-BP refers to a system that allows an encryptor to associate a ciphertext with an input string of any length (in the case of key-policy). We obtain the first (fully-secure, prime-order) schemes for

- Unbounded KP-ABE and CP-ABE for branching programs,
- KP-ABE for branching programs with constant-size ciphertexts,
- CP-ABE for branching programs with constant-size keys.

We note that these are the first such schemes for respective variants even among composite-order or selectively secure schemes. In particular, the only previous schemes, KP-ABE-BP of [27, 31], are of bounded type and require linear-size ciphertexts and keys.[2]

New Implication. We obtain the above new ABE-BP variants by establishing the first implication from ABE for monotone span programs to ABE-BP. This implication is generic as it is not confined in the encoding framework, and hence can be of an independent interest.

1.2 Difficulties and Our Approaches

Recap the Composite-order Framework of [1]. We consider an ABE primitive for predicate R. In such a scheme, a ciphertext is associated with attribute Y, while a key is associated with attribute X, and the decryption of the ciphertext under the key can be done

[2]Note that we consider only *Boolean* branching programs here as in [27], in contrast with [31], where *arithmetic* branching programs are also considered.

if and only if $R(X, Y) = 1$. In the framework of [1], a ciphertext CT encrypting M, and a key SK take the forms of

$$\mathsf{CT} = (\boldsymbol{C}, C_0) = (g_1^{\boldsymbol{c}(\boldsymbol{s},\boldsymbol{h})},\ Me(g_1, g_2)^{\alpha s_0}), \qquad\qquad \mathsf{SK} = g_2^{\boldsymbol{k}(\alpha,\boldsymbol{r},\boldsymbol{h})}$$

where $\boldsymbol{c}$ and $\boldsymbol{k}$ are *encodings* of attributes Y and X associated to a ciphertext and a key, respectively. Here, g_1, g_2 are generators of subgroups of order p_1 of $\mathbb{G}_1, \mathbb{G}_2$, which are asymmetric bilinear groups of composite order $N = p_1p_2p_3$ with bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$.[3] The bold fonts denote vectors. Intuitively, α plays the role of a master key, $\boldsymbol{h}$ represents common variables (or called parameters). These define a public key $\mathsf{PK} = (g_1^{\boldsymbol{h}}, e(g_1, g_2)^{\alpha})$. $\boldsymbol{s}, \boldsymbol{r}$ represents randomness in the ciphertext and the key, respectively, with s_0 being the first element in $\boldsymbol{s}$. The pair $(\boldsymbol{c}, \boldsymbol{k})$ form a *pair encoding* scheme for predicate R. It is exactly this primitive on which the framework of [1] studies and give sufficient conditions so that, roughly speaking, the ABE scheme defined with CT, SK as above would be fully secure (see the full description in §C). The framework defines *semi-functional* ciphertexts and keys (of type 1,2,3) directly from the encoding albeit over another subgroup (of order p_2), as follows. (Here we also write the normal elements as type 0.)

$$\begin{array}{rl} \boldsymbol{C}_{\mathsf{type0}} = g_1^{\boldsymbol{c}(\boldsymbol{s},\boldsymbol{h})} & \\ & \Big) \text{ Subgroup Decision} \\ \boldsymbol{C}_{\mathsf{type1}} = g_1^{\boldsymbol{c}(\boldsymbol{s},\boldsymbol{h})}\hat{g}_1^{\boldsymbol{c}(\hat{\boldsymbol{s}},\hat{\boldsymbol{h}})} & \end{array}$$

$$\begin{array}{rl} \mathsf{SK}_{\mathsf{type0}} = g_2^{\boldsymbol{k}(\alpha,\boldsymbol{r},\boldsymbol{h})} & \\ & \Big) \text{ Subgroup Decision} \\ \mathsf{SK}_{\mathsf{type1}} = g_2^{\boldsymbol{k}(\alpha,\boldsymbol{r},\boldsymbol{h})}\hat{g}_2^{\boldsymbol{k}(0,\hat{\boldsymbol{r}},\hat{\boldsymbol{h}})} & \\ & \Big) \text{ Security of Encoding} \\ \mathsf{SK}_{\mathsf{type2}} = g_2^{\boldsymbol{k}(\alpha,\boldsymbol{r},\boldsymbol{h})}\hat{g}_2^{\boldsymbol{k}(\hat{\alpha},\hat{\boldsymbol{r}},\hat{\boldsymbol{h}})} & \\ & \Big) \text{ Subgroup Decision} \\ \mathsf{SK}_{\mathsf{type3}} = g_2^{\boldsymbol{k}(\alpha,\boldsymbol{r},\boldsymbol{h})}\hat{g}_2^{\boldsymbol{k}(\hat{\alpha},\boldsymbol{0},\boldsymbol{0})} & \end{array}$$

where $\hat{g}_1, \hat{g}_2$ are generators of subgroup of order p_2 of $\mathbb{G}_1, \mathbb{G}_2$, respectively. A security proof in the dual system approaches is structured by using a sequence of hybrid games where each game switches normal to semi-functional elements, so that in the final game, all the elements will be semi-functional and the security can be proved trivially. The framework of [1] makes it clear which game transitions would use which underlying assumptions: we write them along with the definition above. More importantly, it decouples the dual system techniques in such a way that the security of encoding will be used in exactly one type of transition (type 1 to 2 as shown in the diagram), while other transitions will be generically based on subgroup decision assumptions provided by the composite-order bilinear groups. Indeed, the security of encoding is defined to be just what we need for that transition. That is, the security of encoding states that given $\hat{g}_1^{\boldsymbol{c}(\hat{\boldsymbol{s}},\hat{\boldsymbol{h}})}$ and $\hat{g}_2^{\boldsymbol{k}(\hat{\alpha},\hat{\boldsymbol{r}},\hat{\boldsymbol{h}})}$ where $\boldsymbol{c}, \boldsymbol{k}$ encodes (adversarially chosen) Y, X such that $R(X, Y) = 0$, together with generators of every subgroup, the adversary cannot guess if $\hat{\alpha} = 0$ or $\hat{\alpha}$ is random.

Our Goal. Towards translating to a new prime-order based framework, we would like to use the definition and the security of encoding "as is", since this will allow us to instantly

[3] Although the framework of [1] was originally formalized using symmetric groups, generalizing to asymmetric groups is straightforward. For self-containment, we describe the scheme of [1] in asymmetric groups in §C.

instantiate the encoding schemes already proposed and proved security in [1]. If we can leave encoding "as is", we will only have to replace subgroup decision assumptions provided by composite-order groups with some mechanisms from prime-order groups that mimic them.

A First Attempt: Dual-Pairing Vector Space Approaches. One candidate approach for mimicking subgroup decision assumptions in prime-order groups is to use *dual-pairing vector space* techniques initiated by Okamoto and Takashima [40, 41] and extended by Lewko [33]. However, we expect that this would force us to work with one of the encoding (in the pair encoding) in an "orthogonal form" in order to enable inner-product spaces, which seems to be essential in this approach. This is best described by using an example. Consider a pair encoding scheme that underlies IBE of Boneh and Boyen [8], and Lewko and Waters [34] (and hence for the equality predicate). Their encoding is defined as: $\boldsymbol{c}(s, \boldsymbol{h}) = (s, s(h_1 + h_2Y))$ and $\boldsymbol{k}(\alpha, r, \boldsymbol{h}) = (\alpha + r(h_1 + h_2X), r)$, where $\boldsymbol{h} = (h_1, h_2)$. From what we understood, essentially, Lewko [33] converts the IBE scheme to the prime-order setting by (implicitly) considering a vector $\boldsymbol{c}' = (s, sY, -s)$ for $\boldsymbol{c}$ and $\boldsymbol{k}' = (\alpha, r, rX)$ for $\boldsymbol{k}$ and using the fact that the inner product of both vectors becomes αs if $X = Y$ to implement the scheme. In essence, while $\boldsymbol{k}'$ is directly picked from the coefficients in $\boldsymbol{k}$; contrastingly, $\boldsymbol{c}'$ is defined by $(Y, -1)$, which is exactly an orthogonal vector of $(1, Y)$. Orthogonal transformation is certainly doable for simple predicates and encodings, but it could be paramount for enormous encoding schemes (such as in [50, 1]). More importantly, the transformation would change the form of encoding, and we could not use the definition and security of encoding "as is".

Next Attempt: Dual-system Groups. Due to the above difficulty, we then turn to use a variant of the dual-pairing vector space approach recently devised by Chen and Wee [15], called *dual-system groups*. Their notion is generic in a complementary way to the frameworks of [51, 1] in the sense that it *unifies composite-order and prime-order* bilinear group properties for dual system encryption techniques but are applied to only ABE with *specific predicates*, namely, HIBE and spatial encryption, while the frameworks of [51, 1] unify dual system encryption techniques for *any predicate* but *only over composite-order* groups.

We briefly describe their basic idea here (in our notation). Subgroup decision assumptions in composite-order groups will be emulated using the d-Decisional Linear assumption (d-DLIN) in prime-order groups, for any $d \geq 2$. The group $\mathbb{G}_1$ will be emulated by the full *column space* of a random invertible matrix $\boldsymbol{B} \in \mathbb{Z}_p^{(d+1)\times(d+1)}$ (in the exponent). The subgroup of order p_1 and p_2 of $\mathbb{G}_1$ will then be emulated by using the column spaces of d *leftmost columns* of $\boldsymbol{B}$ and of the one *rightmost column* of $\boldsymbol{B}$, respectively. The group $\mathbb{G}_2$ will be emulated similarly but by the matrix $\boldsymbol{B}^{-\top}$. More concretely, we consider bilinear groups $(\bar{\mathbb{G}}_1, \bar{\mathbb{G}}_2, \bar{\mathbb{G}}_T)$ of prime order p with generator $\bar{g}_1, \bar{g}_2$. The roles of generators $g_1, \hat{g}_1, g_2, \hat{g}_2$ (of subgroups of order p_1, p_2 in $\mathbb{G}_1$ and p_1, p_2 in $\mathbb{G}_2$, resp.) in composite-order groups will be played by the following elements in prime-order groups:

$$g_1 \mapsto \bar{g}_1^{\boldsymbol{B}\left(\begin{smallmatrix}\boldsymbol{I}_d\\0\end{smallmatrix}\right)}, \qquad \hat{g}_1 \mapsto \bar{g}_1^{\boldsymbol{B}\left(\begin{smallmatrix}\boldsymbol{0}\\1\end{smallmatrix}\right)}, \qquad g_2 \mapsto \bar{g}_2^{\boldsymbol{B}^{-\top}\left(\begin{smallmatrix}\boldsymbol{I}_d\\0\end{smallmatrix}\right)}, \qquad \hat{g}_2 \mapsto \bar{g}_2^{\boldsymbol{B}^{-\top}\left(\begin{smallmatrix}\boldsymbol{0}\\1\end{smallmatrix}\right)}.$$

We note that $\left(\begin{smallmatrix}\boldsymbol{I}_d\\0\end{smallmatrix}\right)$ denotes the $(d+1)\times d$ matrix where the first d rows comprise the identity matrix while the last row is zero. It functions as a left-projection map.[4] Similarly, $\left(\begin{smallmatrix}\boldsymbol{0}\\1\end{smallmatrix}\right)$ is the $(d+1)\times 1$ matrix where the last row is 1; it functions as a right-projection map.

[4]That is, $X\left(\begin{smallmatrix}\boldsymbol{I}_d\\0\end{smallmatrix}\right) \in \mathbb{Z}_p^{(d+1)\times d}$ is the matrix consisting of all left d columns of X for any $X \in \mathbb{Z}_p^{(d+1)\times(d+1)}$.

Why the Prime-order Dual-system Groups are Suitable. Although the dual system groups were proposed for applying to HIBE in mind, their idea can be generalized to work with elements in the pair encoding notion as follows. The role of parameter $h \in \mathbb{Z}_N$ in composite-order setting will then be played by a *matrix* $\boldsymbol{H} \in \mathbb{Z}_p^{(d+1)\times(d+1)}$, while the role of randomness s, r (in the normal components) in a ciphertext and a key will be played by *vectors*, say $\boldsymbol{v}_s, \boldsymbol{v}_r \in \mathbb{Z}_p^{d\times 1}$, respectively. It is this nature of translating element in a precise way that allows us to work with encoding "as is" by just substituting variables with the translated ones, *e.g.,* our ciphertext encoding function $\boldsymbol{c}((s_0, \ldots, s_\ell), (h_1, \ldots, h_n))$ will become $\boldsymbol{c}((\boldsymbol{v}_{s_0}, \ldots, \boldsymbol{v}_{s_\ell}), (\boldsymbol{H}_1, \ldots, \boldsymbol{H}_n))$.

In order to be fully compatible with the framework of [1], however, the prime-order dual system groups should not only translate the elements but also fully mimic composite-order groups in both aspects of *properties* and *procedures* that are required by both dual system techniques in general and the framework of [1] in particular. As an example of mimicking *properties*, we have orthogonality between generators from different subspaces, mimicking $e(g_1, \hat{g}_2) = 1$:

$$e(\bar{g}_1^{\boldsymbol{B}\left(\begin{smallmatrix}\boldsymbol{I}_d\\0\end{smallmatrix}\right)}, \bar{g}_2^{\boldsymbol{B}^{-\top}\left(\begin{smallmatrix}\boldsymbol{0}\\1\end{smallmatrix}\right)}) = e(\bar{g}_1, \bar{g}_2)^{(\boldsymbol{0}\ 1)\boldsymbol{B}^{-1}\boldsymbol{B}\left(\begin{smallmatrix}\boldsymbol{I}_d\\0\end{smallmatrix}\right)} = 1,$$

where for matrices $\boldsymbol{X}, \boldsymbol{Y}$ with an equal number of rows, we define $e(g_1^{\boldsymbol{X}}, g_2^{\boldsymbol{Y}}) = e(g_1, g_2)^{\boldsymbol{Y}^\top \boldsymbol{X}}$.

As an example of mimicking *procedures*, the "exponentiation" can be done as follows:

$$g_1^s \mapsto \bar{g}_1^{\boldsymbol{B}\left(\begin{smallmatrix}\boldsymbol{I}_d\\0\end{smallmatrix}\right)\boldsymbol{v}_s} = \bar{g}_1^{\boldsymbol{B}\left(\begin{smallmatrix}\boldsymbol{v}_s\\0\end{smallmatrix}\right)}, \qquad g_2^r \mapsto \bar{g}_2^{\boldsymbol{B}^{-\top}\left(\begin{smallmatrix}\boldsymbol{I}_d\\0\end{smallmatrix}\right)\boldsymbol{v}_r} = \bar{g}_2^{\boldsymbol{B}^{-\top}\left(\begin{smallmatrix}\boldsymbol{v}_r\\0\end{smallmatrix}\right)}.$$

In particular, one of the most important properties that the prime-order dual system groups mimic from composite-order groups are *parameter-hiding* and *associativity*. We elaborate the translation of these properties to the prime-order groups in §4.1. *It turns out that, however, the "out-of-the-box" prime-order dual-system groups are still not sufficient for applying to the framework of [1].* We elaborate them as follows.

Challenges and Our Ideas in Applying Dual-system Groups to [1]. We describe an issue on *procedures* first. In the framework of [1], the encoding $\boldsymbol{c}(\boldsymbol{s}, \boldsymbol{h})$ is defined to be an ordered set of polynomials, each of which contains only monomials of the form $s_j, h_k s_j$, where we write $\boldsymbol{h} = (h_1, \ldots, h_n)$, $\boldsymbol{s} = (s_0, \ldots, s_\ell)$. Moreover, in the resulting ABE, we define public key to contain $g_1^{\boldsymbol{h}}$ and ciphertext to contain $g_1^{\boldsymbol{c}(\boldsymbol{s},\boldsymbol{h})}$; therefore, we would require exponentiations in the translated prime-order groups that mimic $g_1^{h_k}, g_1^{s_j}, (g_1^{h_k})^{s_j}$. We have already seen the case of $g_1^{s_j}$ above. However, it turns out that the procedure to do exponentiation as $g_1^{h_k}$ is not implied by the formulation of [15]. In their work, the elements that contain parameter variables, say $\boldsymbol{H}_k$ (which mimics h_k), have the form (in our notation): $\bar{g}_1^{\boldsymbol{B}\boldsymbol{H}_k\left(\begin{smallmatrix}\boldsymbol{I}_d\\0\end{smallmatrix}\right)}$. This cannot be computed from $\bar{g}_1^{\boldsymbol{B}\left(\begin{smallmatrix}\boldsymbol{I}_d\\0\end{smallmatrix}\right)}$ (the generator) and $\boldsymbol{H}_k$ (the parameter variable to be lifted), since we cannot insert matrix in the middle for multiplication. To this end, we newly define exponentiation of $\bar{g}_1^{\boldsymbol{B}\left(\begin{smallmatrix}\boldsymbol{I}_d\\0\end{smallmatrix}\right)}$ by $\boldsymbol{H}_k$ to be $\bar{g}_1^{\boldsymbol{H}_k\boldsymbol{B}\left(\begin{smallmatrix}\boldsymbol{I}_d\\0\end{smallmatrix}\right)}$, where we use the *left multiplication* instead. This will also result in new forms of associativity and parameter-hiding that are deviated from those of [15], see Eq.(5) and Lemma 2. We may consider this as merely a syntactical modification, but it also somewhat simplifies the formulation of [15].

Dealing with Subgroup Decision Assumptions. The next issue regarding *properties* is more important. It turns out that the subgroup decision assumptions-like properties as provided by the dual system group formalization in [15] are not sufficient for translating the framework of [1] to prime-order settings. This is since, to the best of our knowledge, such properties in [15] would guarantee indistinguishability for elements that have *only one element of randomness* in the encoding. More precisely, the out-of-the-box formalization in [15] only guarantees the indistinguishability:

$$\left\{\bar{g}_1^{\boldsymbol{B}\binom{\boldsymbol{v}_s}{0}}, \bar{g}_1^{\boldsymbol{B}\boldsymbol{H}_1\binom{\boldsymbol{v}_s}{0}}, \ldots, \bar{g}_1^{\boldsymbol{B}\boldsymbol{H}_n\binom{\boldsymbol{v}_s}{0}}\right\} \quad , \quad \left\{\bar{g}_1^{\boldsymbol{B}\binom{\boldsymbol{v}_s}{\hat{s}}}, \bar{g}_1^{\boldsymbol{B}\boldsymbol{H}_1\binom{\boldsymbol{v}_s}{\hat{s}}}, \ldots, \bar{g}_1^{\boldsymbol{B}\boldsymbol{H}_n\binom{\boldsymbol{v}_s}{\hat{s}}}\right\} \tag{1}$$

where $\hat{s} \xleftarrow{\$} \mathbb{Z}_p$. This is called *left-subgroup indistinguishability* in [15]. We note that s reflects the *one randomness element* in the encoding of $\boldsymbol{c}(\boldsymbol{s}, \boldsymbol{h})$. In other words, we can use this out-of-the-box property from [15] to deal only with encoding $\boldsymbol{c}$ that has a vector $\boldsymbol{s}$ having only one variable s. In order to deal with $\boldsymbol{s}$ that contains *any number of elements*, as required for general pair encodings defined by [1], we introduce a new technique that uses *random self-reducibility* of the underlying assumption.[5] More precisely, we use the *Matrix Diffie-Hellman Assumption* [17]. We informally recap it as follows. It is defined by a distribution $\mathcal{D}_d$ that outputs matrices of certain forms in $\mathbb{Z}_p^{(d+1)\times(d+1)}$. The assumption states that the adversary cannot distinguish

$$\left\{\bar{g}_1^{\boldsymbol{T}}, \bar{g}_1^{\boldsymbol{T}\binom{\boldsymbol{y}}{0}}\right\} \qquad \text{and} \qquad \left\{\bar{g}_1^{\boldsymbol{T}}, \bar{g}_1^{\boldsymbol{T}\binom{\boldsymbol{y}}{\hat{y}}}\right\}$$

where $\boldsymbol{T} \xleftarrow{\$} \mathcal{D}_d, \boldsymbol{y} \xleftarrow{\$} \mathbb{Z}_p^{d\times 1}, \hat{y} \xleftarrow{\$} \mathbb{Z}_p$. A useful property of this assumption is that it has random self-reducibility: we can extend the column of $\binom{\boldsymbol{y}}{\hat{y}}$ from the original one column to any number of columns, with *tight reduction*. That is, in the extended problem, it becomes to distinguish

$$\left\{\bar{g}_1^{\boldsymbol{T}}, \bar{g}_1^{\boldsymbol{T}\left(\begin{smallmatrix}\boldsymbol{y}_1, & \ldots, & \boldsymbol{y}_\ell\\ 0, & \ldots, & 0\end{smallmatrix}\right)}\right\} \qquad \text{and} \qquad \left\{\bar{g}_1^{\boldsymbol{T}}, \bar{g}_1^{\boldsymbol{T}\left(\begin{smallmatrix}\boldsymbol{y}_1, & \ldots, & \boldsymbol{y}_\ell\\ \hat{y}_1, & \ldots, & \hat{y}_\ell\end{smallmatrix}\right)}\right\}$$

for any ℓ of polynomial size. Hence, we can use the j-th column for simulating the j-th randomness, *i.e.,* $\binom{\boldsymbol{y}_j}{0}$ or $\binom{\boldsymbol{y}_j}{\hat{y}_j}$ for $\binom{\boldsymbol{v}_{s_j}}{0}$ or $\binom{\boldsymbol{v}_{s_j}}{\hat{s}_j}$, and hence obtain the following indistinguishability:

$$\left\{\bar{g}_1^{\boldsymbol{H}_k\boldsymbol{B}\binom{\boldsymbol{v}_{s_j}}{0}}\right\}_{k,j} \qquad \text{and} \qquad \left\{\bar{g}_1^{\boldsymbol{H}_k\boldsymbol{B}\binom{\boldsymbol{v}_{s_j}}{\hat{s}_j}}\right\}_{k,j}$$

which is analogously to (1) as above, but now we can deal with *as many randomness variables as appear in* $\boldsymbol{s} = (s_0, \ldots, s_\ell)$, as required.[6] Now that we modify and extend dual system groups to be compatible with the framework of [1], some issues still remain in such a way that dual system groups formalization inherently cannot avoid so. To this end, we also

[5]Random self-reducibility was already used in [1] but for different reasons and schemes, it was for their (almost) tightly secure IBE scheme in the same paper.

[6]Note that here we implicitly set $\boldsymbol{B} = \boldsymbol{T}$, and in the proof, $\boldsymbol{H}_k$ will be known values to the reduction algorithm. We only give a very informal argument here just to grasp the intuition. The details are in, *e.g.,* Lemma 4.

restrict the *syntax* definition of pair encoding so as to resolve them. This will be done in a minimal manner that all the encoding schemes proposed so far [51, 1, 6] satisfy the additional restrictions. We thus call it *regular* encoding.

Restricting the Syntax Definition of Encodings. The encoding definition allows the multiplication of monomials $h_k s_j$ from a ciphertext encoding with $h_{k'} r_{j'}$ from a key encoding (when pairing). Since we translate the parameters $h_k, h_{k'}$ to matrices $\boldsymbol{H}_k, \boldsymbol{H}_{k'}$ and the matrix multiplication does not commute, such a multiplication procedure from composite-order settings would not be mimicked correctly (see Eq.(6)). To this end, we restrict the encoding scheme so that there will be no multiplication in the above manner. We additionally need three simple requirements which will be used in the security proof of the framework. We describe the intuition for them in §3.1.

1.3 Concurrent and Independent Work

Concurrently and independently, Chen, Gay, and Wee [14] recently propose a generic framework that abstracts dual system ABE for arbitrary predicates in prime-order bilinear groups. The main difference between their framework and ours is that ours can deal with *computationally secure encodings*, while their framework can deal only with information-theoretic ones. As motivated in [1], computational approaches have an advantage in that they are applicable to ABE for predicates where information-theoretic theoretic argument seems insufficient. These include ABE with some *unbounded* properties, or *constant-size* ciphertexts (or keys). We compare some instantiations of [14] (available in their full version) that are relevant to ours in Table 2.

Another difference is that the syntax of encoding in [14] seems more restricted in the sense that it can deal with only one element of randomness, while our syntax can deal with arbitrary many elements. On one hand, one unit of randomness is shown to suffice for all known information-theoretic encodings in [14]. On the other hand, multi-unit randomness seems essential in more esoteric predicates such as ABE for regular languages (of which information-theoretic encodings are not known).

In the conceptual view, their framework unifies both composite-order and prime-order groups into one generic construction via the dual system group syntax. Contrastingly, we focus only on the prime-order generic construction. Nevertheless, since we use the same notion of pair encoding as in the composite-order framework of [1], it can be said that our framework together with [1] provide a unified framework albeit with two generic constructions.

It is also worth noting that [14] extends their ABE framework to achieve weakly attribute-hiding predicate encryption.[7]

1.4 Related Work

Researches on ABE and its generalization, functional encryption (FE), stem from a large number of works [46, 28, 7, 44, 12, 48, 49, 3, 4, 34, 38, 41, 5, 35, 36, 13, 42, 43, 45], and many more, that progressively strengthen ABE in many aspects such as allowable predicate classes, security, underlying assumptions, added functionalities, efficiency, and so on.

[7]Note that, however, only (H)IBE and zero inner-product are currently the only predicates applicable in this extended framework [14].

Composite-order bilinear groups were first suggested by [11]. Freeman [18] proposed guidelines for translating schemes in composite-order groups to prime-order ones. Lewko [33] discussed why the techniques by Freeman was not sufficient for dual-system approaches, and then proposed new techniques, which are then applied to the IBE of [34] and the HIBE of [36]. The techniques of [18, 33] are versatile but do not work as generic compilers. Indeed, using their techniques to newly construct a scheme, a designer must still design a scheme and prove the security anew each time. On the other hand, our framework is a generic compiler which takes a formally defined object, namely, a pair encoding scheme, which is much simpler than ABE, as an input, and automatically outputs a fully-secure ABE scheme in prime-order groups via the generic construction, while its security proof will be guaranteed by the framework.

In this work, we allow only efficient tools, namely, bilinear groups. When basing on bilinear groups, the largest allowable predicate classes for ABE turn out to be Boolean formulae [28, 38] and deterministic finite Automata [50, 1], which are subclasses of log-depth circuits (NC1, or log-space computations). When basing on some seemingly stronger (and hence less efficient) tools, such as lattice-based cryptography and the LWE assumption, multilinear maps [19, 16], or cryptographic obfuscations [21], we can obtain ABE and FE for much larger classes such as poly-size circuits [20, 27, 23, 10], or Turing machines [25, 26]. We remark that, until recently, all known ABE systems for these general classes are only selectively secure (or fully secure but with exponential reductions). Fully secure ABE systems for circuits are recently proposed in [22, 2] using composite-order multi-linear maps [24] via dual system techniques. Constructing such ABEs in prime-order settings is still an interesting open problem. Our framework might be a suitable starting point for solving it.

2 Preliminaries

2.1 Definitions of Attribute Based Encryption

Predicate Family. We consider a predicate family $R = \{R_\kappa\}_{\kappa \in \mathbb{N}^c}$, for some constant $c \in \mathbb{N}$, where a relation $R_\kappa : \mathbb{X}_\kappa \times \mathbb{Y}_\kappa \to \{0,1\}$ is a predicate function that maps a pair of key attribute in a space $\mathbb{X}_\kappa$ and ciphertext attribute in a space $\mathbb{Y}_\kappa$ to $\{0,1\}$. The family index $\kappa = (n_1, n_2, \ldots)$ specifies the description of a predicate from the family. We will often neglect κ for simplicity of exposition.

Attribute Based Encryption Syntax. An attribute based encryption (ABE) scheme for predicate family R consists of the following algorithms.

- $\mathsf{Setup}(1^\lambda, \kappa) \to (\mathsf{PK}, \mathsf{MSK})$: takes as input a security parameter 1^λ and a family index κ of predicate family R, and outputs a master public key PK and a master secret key MSK.
- $\mathsf{Encrypt}(Y, M, \mathsf{PK}) \to \mathsf{CT}$: takes as input a ciphertext attribute $Y \in \mathbb{Y}_\kappa$, a message $M \in \mathcal{M}$, and public key PK. It outputs a ciphertext CT.
- $\mathsf{KeyGen}(X, \mathsf{MSK}, \mathsf{PK}) \to \mathsf{SK}$: takes as input a key attribute $X \in \mathbb{X}_\kappa$ and the master key MSK. It outputs a secret key SK.
- $\mathsf{Decrypt}(\mathsf{CT}, \mathsf{SK}) \to M$: given a ciphertext CT with its attribute Y and the decryption key SK with its attribute X, it outputs a message M or $\perp$.

Correctness. Consider all indexes κ, all $M \in \mathcal{M}$, $X \in \mathbb{X}_\kappa$, $Y \in \mathbb{Y}_\kappa$ such that $R_\kappa(X, Y) = 1$. If $\mathsf{Encrypt}(Y, M, \mathsf{PK}) \to \mathsf{CT}$ and $\mathsf{KeyGen}(X, \mathsf{MSK}, \mathsf{PK}) \to \mathsf{SK}$ where $(\mathsf{PK}, \mathsf{MSK})$ is generated from $\mathsf{Setup}(1^\lambda, \kappa)$, then $\mathsf{Decrypt}(\mathsf{CT}, \mathsf{SK}) \to M$.

We defer the security definition for ABE to §A.

2.2 Bilinear Groups and Assumptions

In our framework, for maximum generality and clarity, we consider asymmetric bilinear groups $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ of prime order p, with an efficiently computable bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. The symmetric version of our framework can be obtained by just setting $\mathbb{G}_1 = \mathbb{G}_2$.[8] We define a bilinear group generator $\mathcal{G}(\lambda)$ that takes as input a security parameter λ and outputs $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, p)$. We recall that e has the bilinear property: $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$ for any $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$, $a, b \in \mathbb{Z}$ and the non-degeneration property: $e(g_1, g_2) \neq 1 \in \mathbb{G}_T$ whenever $g_1 \neq 1 \in \mathbb{G}_1, g_2 \neq 1 \in \mathbb{G}_2$.

Notation for Matrix in the Exponents. Vectors will be treated as either row or column matrices. When unspecified, we shall let it be a row vector. Let $\mathbb{G}$ be a group. Let $\boldsymbol{a} = (a_1, \ldots, a_n)$ and $\boldsymbol{b} = (b_1, \ldots, b_n) \in \mathbb{G}^n$. We denote $\boldsymbol{a} \cdot \boldsymbol{b} = (a_1 \cdot b_1, \ldots, a_n \cdot b_n)$, where '$\cdot$' is the group operation of $\mathbb{G}$. For $g \in \mathbb{G}$ and $\boldsymbol{c} = (c_1, \ldots, c_n) \in \mathbb{Z}^n$, we denote $g^{\boldsymbol{c}} = (g^{c_1}, \ldots, g^{c_n})$. We denote by $\mathbb{GL}_{p,n}$ the group of invertible matrices (the general linear group) in $\mathbb{Z}_p^{n\times n}$. Consider $\boldsymbol{M} \in \mathbb{Z}_p^{d\times n}$ (the set of all $d \times n$ matrices in $\mathbb{Z}_p$). We denote the transpose of $\boldsymbol{M}$ as $\boldsymbol{M}^\top$. We denote by $g^{\boldsymbol{M}}$ the matrix in $\mathbb{G}^{d\times n}$ of which its (i, j) entry is $g^{\boldsymbol{M}_{i,j}}$, where $\boldsymbol{M}_{i,j}$ is the (i, j) entry of $\boldsymbol{M}$. For $\boldsymbol{Q} \in \mathbb{Z}_p^{\ell\times d}$, we denote $(g^{\boldsymbol{Q}})^{\boldsymbol{M}} = g^{\boldsymbol{QM}}$. Note that from $\boldsymbol{M}$ and $g^{\boldsymbol{Q}} \in \mathbb{G}^{\ell\times d}$, we can compute $g^{\boldsymbol{QM}}$ without knowing $\boldsymbol{Q}$, since its (i, j) entry is $\prod_{k=1}^d (g^{\boldsymbol{Q}_{i,k}})^{\boldsymbol{M}_{k,j}}$. The same can be said about $g^{\boldsymbol{M}}$ and $\boldsymbol{Q}$. For $\boldsymbol{X} \in \mathbb{Z}_p^{r\times c_1}$ and $\boldsymbol{Y} \in \mathbb{Z}_p^{r\times c_2}$, we denote its pairing as:

$$e(g_1^{\boldsymbol{X}}, g_2^{\boldsymbol{Y}}) = e(g_1, g_2)^{\boldsymbol{Y}^\top \boldsymbol{X}} \in \mathbb{G}_T^{c_2\times c_1}.$$

Matrix-DH Assumptions [17]**.** We call $\mathcal{D}_d$ a matrix distribution if it outputs (in poly time, with overwhelming probability) matrices in $\mathbb{Z}_p^{(d+1)\times(d+1)}$ of the form:

$$\boldsymbol{T} = \begin{array}{c c} & \begin{array}{cc} d & 1 \end{array} \\ \begin{array}{c} d \\ 1 \end{array} & \begin{pmatrix} \boldsymbol{M} & \boldsymbol{0} \\ \boldsymbol{c} & 1 \end{pmatrix} \end{array} \xleftarrow{\$} \mathcal{D}_d. \tag{2}$$

such that $\boldsymbol{M}$ is an invertible matrix in $\mathbb{Z}_p^{d\times d}$ (*i.e.,* $\boldsymbol{M} \in \mathbb{GL}_{p,d}$). We say that the *$\mathcal{D}_d$-Matrix Diffie-Hellman Assumption* holds relative to $\mathcal{G}$ if for all ppt adversaries $\mathcal{A}$, the advantage

$$\mathsf{Adv}_{\mathcal{A}}^{\mathcal{D}_d\text{-}\mathsf{MatDH}}(\lambda) := \left| \Pr\left[\mathcal{A}(\mathbb{G}, g_1^{\boldsymbol{T}}, g_1^{\boldsymbol{T}\binom{\boldsymbol{y}}{0}}) = 1\right] - \Pr\left[\mathcal{A}(\mathbb{G}, g_1^{\boldsymbol{T}}, g_1^{\boldsymbol{T}\binom{\boldsymbol{y}}{\hat{y}}}) = 1\right] \right|$$

is negligible in λ, where the probability is taken over $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, p) \xleftarrow{\$} \mathcal{G}(\lambda)$, $g_1 \xleftarrow{\$} \mathbb{G}_1$, $g_2 \xleftarrow{\$} \mathbb{G}_2$, $\boldsymbol{T} \xleftarrow{\$} \mathcal{D}_d$, $\boldsymbol{y} \xleftarrow{\$} \mathbb{Z}_p^{d\times 1}$, $y \xleftarrow{\$} \mathbb{Z}_p$, and the randomness of $\mathcal{A}$. Denote $\mathbb{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, p, g_1, g_2)$.

[8]This can be done since we will *not* assume, *e.g.,* the External DH assumption [9].

Remark 1. We remark that the assumption is progressively weaker as d increases. In particular, in any k-multi-linear groups, the $\mathcal{D}_d$-Matrix DH Assumption is false if $d < k$ ([17]). Hence, we can use the assumption with $d \geq 2$, since we work on bilinear groups. The most well-known special case of the $\mathcal{D}_d$-Matrix-DH Assumption is the Decision d-Linear Assumption, for which $\boldsymbol{M}$ are restricted to random diagonal matrices and $\boldsymbol{c}$ is fixed as the vector with all 1's.

Our scheme will use arbitrary $\mathcal{D}_d$ for maximal generality. One can directly tradeoff the weakness of assumption and the sizes of ciphertexts and keys by d.

Random Self Reducibility of Matrix-DH Assumptions. The $\mathcal{D}_d$-Matrix-DH Assumption is random self reducible, as shown in [17]: the problem instance defined by $(\boldsymbol{T}, \binom{\boldsymbol{y}}{\hat{y}})$ can be randomized to another instance defined by $(\boldsymbol{T}, \binom{\boldsymbol{y}'}{\hat{y}'})$. This is done by choosing $\boldsymbol{\delta} \xleftarrow{\$} \mathbb{Z}_p^{d\times 1}, \hat{\delta} \xleftarrow{\$} \mathbb{Z}_p$ and setting

$$g_1^{\boldsymbol{T}\binom{\boldsymbol{y}'}{\hat{y}'}} = g_1^{\boldsymbol{T}\binom{\boldsymbol{y}}{\hat{y}}\hat{\delta}} g_1^{\boldsymbol{T}\binom{\boldsymbol{\delta}}{0}},$$

and observe that $y = 0$ iff $y' = 0$. We can gather each new instance $\binom{\boldsymbol{y}'}{\hat{y}'}$ into columns of a matrix and consider the *m-fold* $\mathcal{D}_d$-Matrix-DH Assumption for which the advantage is defined as

$$\mathsf{Adv}_{\mathcal{A}}^{m,\mathcal{D}_d\text{-MatDH}}(\lambda) := \left| \Pr\left[\mathcal{A}(\mathbb{G}, g_1^{\boldsymbol{T}}, g_1^{\boldsymbol{T}\binom{\boldsymbol{Y}}{\boldsymbol{0}}}) = 1\right] - \Pr\left[\mathcal{A}(\mathbb{G}, g_1^{\boldsymbol{T}}, g_1^{\boldsymbol{T}\binom{\boldsymbol{Y}}{\hat{\boldsymbol{y}}}}) = 1\right] \right|,$$

where the probability is taken over $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, p) \xleftarrow{\$} \mathcal{G}(\lambda)$, $g_1 \xleftarrow{\$} \mathbb{G}_1$, $g_2 \xleftarrow{\$} \mathbb{G}_2$, $\boldsymbol{T} \xleftarrow{\$} \mathcal{D}_d$, $\boldsymbol{Y} \xleftarrow{\$} \mathbb{Z}_p^{d\times m}$, $\hat{\boldsymbol{y}} \xleftarrow{\$} \mathbb{Z}_p^{1\times m}$, and the randomness of $\mathcal{A}$. Again, we denote $\mathbb{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, p, g_1, g_2)$. Due to the random self-reducibility, the reduction to the m-fold variant is tight.

Proposition 1. ([17]) *For any integer m, for all ppt adversary $\mathcal{A}$, there exists a ppt algorithm $\mathcal{A}'$ such that* $\mathsf{Adv}_{\mathcal{A}'}^{m,\mathcal{D}_d\text{-MatDH}}(\lambda) = \mathsf{Adv}_{\mathcal{A}}^{\mathcal{D}_d\text{-MatDH}}(\lambda)$.

3 Definition of Pair Encoding

We recall the definition of pair encoding schemes as given in [1]. A pair encoding scheme for predicate family R consists of four deterministic algorithms given by $\mathsf{P} = (\mathsf{Param}, \mathsf{Enc1}, \mathsf{Enc2}, \mathsf{Pair})$ as follows:

- $\mathsf{Param}(\kappa) \to n$. It takes as input an index κ and outputs an integer n, which specifies the number of *common variables* in $\mathsf{Enc1}$, $\mathsf{Enc2}$. For the default notation, let $\boldsymbol{h} = (h_1, \dots, h_n)$ denote the the list of common variables.

- $\mathsf{Enc1}(X) \to (\boldsymbol{k} = (k_1, \dots, k_{m_1}); m_2)$. It takes as inputs $X \in \mathbb{X}_\kappa$, and outputs a sequence of polynomials $\{k_i\}_{i\in[1,m_1]}$ with coefficients in $\mathbb{Z}_p$, and $m_2 \in \mathbb{N}$. We require that each polynomial k_i is a *linear combination of monomials* $\alpha, r_j, h_k r_j$, where $\alpha, r_1, \dots, r_{m_2}, h_1, \dots, h_n$ are variables. More precisely, it outputs a set of coefficients $\{b_i\}_{i\in[1,m_1]}, \{b_{i,j}\}_{i\in[1,m_1],j\in[1,m_2]}$, $\{b_{i,j,k}\}_{i\in[1,m_1],j\in[1,m_2],k\in[1,n]}$ that defines the sequence of polynomials:

$$\boldsymbol{k}(\alpha,(r_1,\ldots,r_{m_2}),(h_1,\ldots,h_n)) = \left\{ b_i\alpha + \left(\sum_{j\in[1,m_2]} b_{i,j}r_j\right) + \left(\sum_{\substack{j\in[1,m_2]\\ k\in[1,n]}} b_{i,j,k}h_k r_j\right)\right\}_{i\in[1,m_1]} \tag{3}$$

- $\mathsf{Enc2}(Y) \to \big(\boldsymbol{c} = (c_1,\ldots,c_{w_1}); w_2\big)$. It takes as inputs $Y \in \mathbb{Y}_\kappa$, and outputs a sequence of polynomials $\{c_i\}_{i\in[1,w_1]}$ with coefficients in $\mathbb{Z}_p$, and $w_2 \in \mathbb{N}$. We require that each polynomial c_i is a *linear combination of monomials* $s_j, h_k s_j$, where $s_0, s_1,\ldots,s_{w_2}, h_1,\ldots,h_n$ are variables. More precisely, it outputs $\{a_{i,j}\}_{i\in[1,w_1],j\in[0,w_2]}, \{a_{i,j,k}\}_{i\in[1,w_1],j\in[0,w_2],k\in[1,n]}$ which is a set of coefficients that defines the sequence of polynomials:

$$\boldsymbol{c}((s_0,s_1,\ldots,s_{w_2}),(h_1,\ldots,h_n)) = \left\{ \left(\sum_{j\in[0,w_2]} a_{i,j}s_j\right) + \left(\sum_{\substack{j\in[0,w_2]\\ k\in[1,n]}} a_{i,j,k}h_k s_j\right)\right\}_{i\in[1,w_1]} \tag{4}$$

- $\mathsf{Pair}(X,Y) \to \boldsymbol{E}$. It takes as inputs X, Y, and output $\boldsymbol{E} \in \mathbb{Z}_p^{m_1\times w_1}$.

Correctness. The correctness requirement is defined as follows. Let $(\boldsymbol{k}; m_2) \leftarrow \mathsf{Enc1}(X)$, $(\boldsymbol{c}; w_2) \leftarrow \mathsf{Enc2}(Y)$, and $\boldsymbol{E} \leftarrow \mathsf{Pair}(X,Y)$. We have that if $R(X,Y) = 1$, then $\boldsymbol{kEc}^\top = \alpha s_0$, where the equality holds symbolically.

Note that since $\boldsymbol{kEc}^\top = \sum_{i\in[1,m_1],j\in[1,w_1]} E_{i,j}k_i c_j$, the correctness amounts to check if there is a linear combination of $k_i c_j$ terms summed up to αs_0. In what follows, we denote $\boldsymbol{h} = (h_1,\ldots,h_n), \boldsymbol{r} = (r_1,\ldots,r_{m_2}), \boldsymbol{s} = (s_0,s_1,\ldots,s_{w_2})$.

3.1 Regular Pair Encoding

Towards proving the security of our framework in prime-order groups, we require new properties for pair encoding. We formalize them as *regularity.* This would generally confine the class of encoding schemes that the new framework can deal with from the previous framework by [1]. Nonetheless, the confinement seems natural since all the pair encoding schemes proposed so far [1, 51, 6] turn out to be regular, and hence are not affected. The definition can be best described by using an example, we will thus illustrate an the regularity of an existing encoding from [1] in §F.

Definition 1 (Regular Pair Encoding)**.** We call a pair encoding *regular* if the following hold:

1. For $i \in [1,m_1], i' \in [1,w_1]$ such that there is $j \in [1,m_2], k \in [1,n], j' \in [1,w_2], k' \in [1,n]$ where $b_{i,j,k} \neq 0$ and $a_{i',j',k'} \neq 0$, we require that $E_{i,i'} = 0$.

2. For $j \in [1,m_2]$ such that there is no $i' \in [1,m_1]$ where $k_{i'} = r_j$, we require that for any $i \in [1,m_1], k \in [1,n]$, we have $b_{i,j,k} = 0$.

3. For $j \in [0, w_2]$ such that there is no $i' \in [1, w_1]$ where $c_{i'} = s_j$, we require that for any $i \in [1, m_1], k \in [1, n]$, we have $a_{i,j,k} = 0$.

4. There is $i' \in [1, w_1]$ where $c_{i'} = s_0$. Wlog, we always let such i' be 1, *i.e.*, the first polynomial in $\boldsymbol{c}$ is $c_1 = s_0$.

Explaining the Definition. The first restriction basically states that the multiplication of $(h_k r_j)$ and $(h_{k'} s_{j'})$ will not be allowed when pairing. The reason to do so is that the parameter $h_k, h_{k'}$ will be translated to matrices, and the matrix multiplication does not commute; hence, the multiplication procedure would not be mimicked correctly (from the composite-order setting) if it were to be allowed (see Eq. (6)). This restriction is quite natural since the product $r_j h_k, h_{k'} s_{j'}$ can be implemented by grouping $h_{k''} = h_k h_{k'}$, and just using associativity $(r_j h_{k''}) s_{j'} = r_j (h_{k''} s_{j'})$ instead; therefore, the multiplication of $(h_k r_j)$ and $(h_{k'} s_{j'})$ will not be needed in the first place.

The second restriction basically states that a term $h_k r_j$ is allowed in the key encoding only if r_j is given out explicitly in the key encoding. The third is similar but for the ciphertext encoding. These restrictions are also natural since intuitively to cancel out $h_k r_j$ (so that the bilinear combination would give only the term αs_0 and no others), one would need r_j to multiply with, say $h_k s_{j'}$ (since we cannot do the multiplication concerning two parameters, as depicted above). The meaning of the fourth is clear: s_0 must be given out in the encoding. These latter three restrictions will be used for the security proofs of game transitions that are based on the security of encodings (Lemma 7,10).

3.2 Security Definitions for Pair Encodings

We will use (almost) the same definitions for security notions of pair encoding schemes as given in [1], with a refinement regarding the number of queries in [6]. We therefore defer it to §B. The definitions comprise an information-theoretic flavor called *perfectly master-key hiding* (PMH) and a computational flavor called *doubly selectively master-key hiding*, which consists of two sub-notions called *selectively master-key hiding* (SMH) and *co-selectively master-key hiding* (CMH). We use a new refinement proposed in [6] that parameterizes the notions with the number of queries for ciphertext and key. The notions in [1] can then be rephrased as $(1, \mathsf{poly})$-SMH and $(1, 1)$-CMH. An advantage of this refinement is that we can have a "dual" conversion that converts between $(1, 1)$-CMH and $(1, 1)$-SMH for dual predicate [6]. We remark a slight difference from those in [1, 6]: here we define it in *asymmetric* and *prime-order* groups, while it was defined in *symmetric* and *prime-order subgroup of composite-order* groups in [1, 6]. We argue that these are merely syntactical and the security of all concrete pair encoding schemes proposed in [1, 6] will preserve. We refer to §B.

4 Our Framework in Prime-Order Groups

4.1 Intuition for Translation to Prime-Order Groups

Before describing our prime-order framework, we describe how we translate elements, procedures, and properties from the composite-order group setting to the prime-order group setting. For self-containment, we also describe the composite-order framework of [1] in §C.

- **Generators.** In composite-order groups, we have $g_1 \in \mathbb{G}_{1,p_1}$, $\hat{g}_1 \in \mathbb{G}_{1,p_2}$, $g_2 \in \mathbb{G}_{2,p_1}$, $\hat{g}_2 \in \mathbb{G}_{2,p_2}$. In prime-order groups, we use the following elements respectively:

$$g_1^{\boldsymbol{B}\binom{\boldsymbol{I}_d}{0}} \in \mathbb{G}_1^{(d+1)\times d}, \qquad g_1^{\boldsymbol{B}\binom{\boldsymbol{0}}{1}} \in \mathbb{G}_1^{(d+1)\times 1},$$
$$g_2^{\boldsymbol{Z}\binom{\boldsymbol{I}_d}{0}} \in \mathbb{G}_1^{(d+1)\times d}, \qquad g_2^{\boldsymbol{Z}\binom{\boldsymbol{0}}{1}} \in \mathbb{G}_1^{(d+1)\times 1}.$$

where $\boldsymbol{B} \xleftarrow{\$} \mathbb{GL}_{p,d+1}$, $\boldsymbol{Z} := \boldsymbol{B}^{-\top}\boldsymbol{D}$ where $\boldsymbol{D} := \left(\begin{smallmatrix} \tilde{\boldsymbol{D}} & \boldsymbol{0} \\ \boldsymbol{0} & 1 \end{smallmatrix}\right) \in \mathbb{GL}_{p,d+1}$ with $\tilde{\boldsymbol{D}} \xleftarrow{\$} \mathbb{GL}_{p,d}$.

- **Variables.** The role of parameter h_k (in $\boldsymbol{h}$) in the composite-order setting will be played by a matrix $\boldsymbol{H}_k \in \mathbb{Z}_p^{(d+1)\times(d+1)}$. The role of randomness s_j, r_j (in $\boldsymbol{s}, \boldsymbol{r}$) to be exponentiated over g_1, g_2 in the composite-order setting for a ciphertext and a key will be played by vectors $\boldsymbol{s}_j, \boldsymbol{r}_j \in \mathbb{Z}_p^{d\times 1}$, respectively, in the prime-order setting. The role of randomness $\hat{s}_j, \hat{r}_j$ (in $\hat{\boldsymbol{s}}, \hat{\boldsymbol{r}}$) to be exponentiated over $\hat{g}_1, \hat{g}_2$ will be used as it is (a scalar in $\mathbb{Z}_p$) in the prime-order setting.

- **Exponentiation by parameter.** To mimic exponentiation $g_1^{h_k}$, $\hat{g}_1^{\hat{h}_k}$, $g_2^{h_k}$, $\hat{g}_2^{\hat{h}_k}$ in the composite-order setting, we do the following in the prime-order setting:

$$g_1^{\boldsymbol{H}_k\boldsymbol{B}\binom{\boldsymbol{I}_d}{0}} \in \mathbb{G}_1^{(d+1)\times d}, \qquad g_1^{\boldsymbol{B}\binom{\boldsymbol{0}}{1}\hat{h}_k} \in \mathbb{G}_1^{(d+1)\times 1},$$
$$g_2^{\boldsymbol{H}_k^\top\boldsymbol{Z}\binom{\boldsymbol{I}_d}{0}} \in \mathbb{G}_1^{(d+1)\times d}, \qquad g_2^{\boldsymbol{Z}\binom{\boldsymbol{0}}{1}\hat{h}_k} \in \mathbb{G}_1^{(d+1)\times 1}$$

- **Exponentiation by randomness.** To mimic exponentiation $g_1^{s_j}$, $\hat{g}_1^{\hat{s}_j}$, $g_2^{r_j}$, $\hat{g}_2^{\hat{r}_j}$, in the composite-order setting, we do the following in the prime-order setting:

$$g_1^{\boldsymbol{B}\binom{\boldsymbol{I}_d}{0}\boldsymbol{s}_j} = g_1^{\boldsymbol{B}\binom{\boldsymbol{s}_j}{0}} \in \mathbb{G}_1^{(d+1)\times 1}, \qquad g_1^{\boldsymbol{B}\binom{\boldsymbol{0}}{1}\hat{s}_j} = g_1^{\boldsymbol{B}\binom{\boldsymbol{0}}{\hat{s}_j}} \in \mathbb{G}_1^{(d+1)\times 1},$$
$$g_2^{\boldsymbol{Z}\binom{\boldsymbol{I}_d}{0}\boldsymbol{r}_j} = g_2^{\boldsymbol{Z}\binom{\boldsymbol{r}_j}{0}} \in \mathbb{G}_1^{(d+1)\times 1}, \qquad g_2^{\boldsymbol{Z}\binom{\boldsymbol{0}}{1}\hat{r}_j} = g_2^{\boldsymbol{Z}\binom{\boldsymbol{0}}{\hat{r}_j}} \in \mathbb{G}_1^{(d+1)\times 1},$$

- **Exponentiation by randomness over parameter.** To mimic $(g_1^{h_k})^{s_j}$, $(\hat{g}_1^{\hat{h}_k})^{\hat{s}_j}$, $(g_2^{h_k})^{r_j}$, $(\hat{g}_2^{\hat{h}_k})^{\hat{r}_j}$, in the composite-order setting, we do the following in the prime-order setting:

$$g_1^{\boldsymbol{H}_k\boldsymbol{B}\binom{\boldsymbol{I}_d}{0}\boldsymbol{s}_j} = g_1^{\boldsymbol{H}_k\boldsymbol{B}\binom{\boldsymbol{s}_j}{0}} \in \mathbb{G}_1^{(d+1)\times 1}, \qquad g_1^{\boldsymbol{B}\binom{\boldsymbol{0}}{1}\hat{h}_k\hat{s}_j} = g_1^{\boldsymbol{B}\binom{\boldsymbol{0}}{\hat{h}_k\hat{s}_j}} \in \mathbb{G}_1^{(d+1)\times 1},$$
$$g_2^{\boldsymbol{H}_k^\top\boldsymbol{Z}\binom{\boldsymbol{I}_d}{0}\boldsymbol{r}_j} = g_2^{\boldsymbol{H}_k^\top\boldsymbol{Z}\binom{\boldsymbol{r}_j}{0}} \in \mathbb{G}_1^{(d+1)\times 1}, \qquad g_2^{\boldsymbol{Z}\binom{\boldsymbol{0}}{1}\hat{h}_k\hat{r}_j} = g_2^{\boldsymbol{Z}\binom{\boldsymbol{0}}{\hat{h}_k\hat{r}_j}} \in \mathbb{G}_1^{(d+1)\times 1}$$

- **Pair Encoding.** From the ciphertext attribute encoding $\boldsymbol{c}(\boldsymbol{s}, \boldsymbol{h})$ defined in Eq.(4), we define an augmented encoding with variables $\boldsymbol{x}_j$ replacing s_j and $\boldsymbol{H}_k$ replacing h_k as

$$\boldsymbol{c}_{\boldsymbol{B}}((\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{w_2}), (\boldsymbol{H}_1, \ldots, \boldsymbol{H}_n)) := \left\{ \left(\sum_{j\in[0,w_2]} a_{i,j}\boldsymbol{B}\boldsymbol{x}_j \right) + \left(\sum_{\substack{j\in[0,w_2] \\ k\in[1,n]}} a_{i,j,k}\boldsymbol{H}_k\boldsymbol{B}\boldsymbol{x}_j \right) \right\}_{i\in[1,w_1]}.$$

Similarly, from the key attribute encoding $\boldsymbol{k}(\alpha, \boldsymbol{s}, \boldsymbol{h})$ defined in Eq.(3), we define an augmented encoding with variables $\boldsymbol{y}_j$ replacing s_j, $\boldsymbol{H}_k$ replacing h_k, and $\boldsymbol{\alpha} \in \mathbb{Z}_p^{(d+1)\times 1}$ replacing α as

$$\boldsymbol{k}_{\boldsymbol{Z}}(\boldsymbol{\alpha}, (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{m_2}), (\boldsymbol{H}_1, \ldots, \boldsymbol{H}_n)) = \left\{ b_i\boldsymbol{\alpha} + \left(\sum_{j\in[1,m_2]} b_{i,j}\boldsymbol{Z}\boldsymbol{y}_j \right) + \left(\sum_{\substack{j\in[1,m_2]\\ k\in[1,n]}} b_{i,j,k}\boldsymbol{H}_k^\top \boldsymbol{Z}\boldsymbol{y}_j \right) \right\}_{i\in[1,m_1]}.$$

Note that we will use $\boldsymbol{x}_j$ as either $\left(\begin{smallmatrix}\boldsymbol{s}_j\\0\end{smallmatrix}\right)$ or $\left(\begin{smallmatrix}\boldsymbol{s}_j\\\hat{s}_j\end{smallmatrix}\right)$ and $\boldsymbol{y}_j$ as either $\left(\begin{smallmatrix}\boldsymbol{r}_j\\0\end{smallmatrix}\right)$ or $\left(\begin{smallmatrix}\boldsymbol{r}_j\\\hat{r}_j\end{smallmatrix}\right)$.

- **Associativity.** In the composite-order setting, we have $e(t_1^{h_k s_j}, t_2^{r_i}) = e(t_1^{s_j}, t_2^{h_k r_i})$, for any $t_1 \in \mathbb{G}_1, t_2 \in \mathbb{G}_2$. In the prime-order setting, we have

$$e(g_1^{\boldsymbol{H}_k\boldsymbol{B}\left(\begin{smallmatrix}\boldsymbol{s}_j\\\hat{s}_j\end{smallmatrix}\right)}, g_2^{\boldsymbol{Z}\left(\begin{smallmatrix}\boldsymbol{r}_i\\\hat{r}_i\end{smallmatrix}\right)}) = e(g_1^{\boldsymbol{B}\left(\begin{smallmatrix}\boldsymbol{s}_j\\\hat{s}_j\end{smallmatrix}\right)}, g_2^{\boldsymbol{H}_k^\top\boldsymbol{Z}\left(\begin{smallmatrix}\boldsymbol{r}_i\\\hat{r}_i\end{smallmatrix}\right)}). \tag{5}$$

as $e(g_1, g_2)^{\left((\boldsymbol{r}_i^\top\ \hat{r}_i)\boldsymbol{Z}^\top\right)\left(\boldsymbol{H}_k\boldsymbol{B}\left(\begin{smallmatrix}\boldsymbol{s}_j\\\hat{s}_j\end{smallmatrix}\right)\right)} = e(g_1, g_2)^{\left((\boldsymbol{r}_i^\top\ \hat{r}_i)\boldsymbol{Z}^\top\boldsymbol{H}_k\right)\left(\boldsymbol{B}\left(\begin{smallmatrix}\boldsymbol{s}_j\\\hat{s}_j\end{smallmatrix}\right)\right)}$.

- **Parameter-Hiding.** In composite-order groups, we have that: given $g_1^{h_k}, g_2^{h_k}, g_1, \hat{g}_1, g_2, \hat{g}_2, p_1, p_2$; $h_k \bmod p_2$ is information-theoretically hidden. In the prime-order setting, we have Lemma 2.

Lemma 2. *Given* $g_1^{\boldsymbol{H}_k\boldsymbol{B}\left(\begin{smallmatrix}\boldsymbol{I}_d\\0\end{smallmatrix}\right)} \in \mathbb{G}_1^{(d+1)\times d}, g_2^{\boldsymbol{H}_k^\top\boldsymbol{Z}\left(\begin{smallmatrix}\boldsymbol{I}_d\\0\end{smallmatrix}\right)} \in \mathbb{G}_2^{(d+1)\times d}$*, along with* $g_1, g_2, \boldsymbol{B}, \boldsymbol{Z}$*, the quantity of the* $(d+1, d+1)$ *entry of* $\boldsymbol{B}^{-1}\boldsymbol{H}_k\boldsymbol{B}$ *is information-theoretically hidden.*

Proof. Write $\boldsymbol{B}^{-1}\boldsymbol{H}_k\boldsymbol{B} = \left(\begin{smallmatrix}M_1 & M_2\\M_3 & \delta\end{smallmatrix}\right)$, where $M_1 \in \mathbb{Z}_p^{d\times d}, M_2 \in \mathbb{Z}_p^{d\times 1}, M_3 \in \mathbb{Z}_p^{1\times d}, \delta \in \mathbb{Z}_p$. We have

$$\begin{aligned}
\boldsymbol{H}_k\boldsymbol{B}\left(\begin{smallmatrix}\boldsymbol{I}_d\\0\end{smallmatrix}\right) &= \boldsymbol{B}\left(\begin{smallmatrix}M_1 & M_2\\M_3 & \delta\end{smallmatrix}\right)\left(\begin{smallmatrix}\boldsymbol{I}_d\\0\end{smallmatrix}\right) = \boldsymbol{B}\left(\begin{smallmatrix}M_1\\M_3\end{smallmatrix}\right),\\
\boldsymbol{H}_k^\top\boldsymbol{Z}\left(\begin{smallmatrix}\boldsymbol{I}_d\\0\end{smallmatrix}\right) &= \boldsymbol{H}_k^\top\boldsymbol{B}^{-\top}\left(\begin{smallmatrix}\tilde{\boldsymbol{D}} & \boldsymbol{0}\\\boldsymbol{0} & 1\end{smallmatrix}\right)\left(\begin{smallmatrix}\boldsymbol{I}_d\\0\end{smallmatrix}\right) = \boldsymbol{B}^{-\top}\left(\begin{smallmatrix}M_1^\top & M_3^\top\\M_2^\top & \delta\end{smallmatrix}\right)\left(\begin{smallmatrix}\tilde{\boldsymbol{D}}\\\boldsymbol{0}\end{smallmatrix}\right) = \boldsymbol{B}^{-\top}\left(\begin{smallmatrix}M_1^\top\tilde{\boldsymbol{D}}\\M_2^\top\tilde{\boldsymbol{D}}\end{smallmatrix}\right),
\end{aligned}$$

where in the second line, we use the fact that $\boldsymbol{B}^\top\boldsymbol{H}^\top\boldsymbol{B}^{-\top} = \left(\begin{smallmatrix}M_1^\top & M_3^\top\\M_2^\top & \delta\end{smallmatrix}\right)$. We can see that both $\boldsymbol{H}_k\boldsymbol{B}\left(\begin{smallmatrix}\boldsymbol{I}_d\\0\end{smallmatrix}\right), \boldsymbol{H}_k^\top\boldsymbol{Z}\left(\begin{smallmatrix}\boldsymbol{I}_d\\0\end{smallmatrix}\right)$ do not contain information on δ. □

We also give an intuition why commutativity does not preserve to prime-order settings as follows.

- **Unavailable Commutativity.** In the composite-order setting, we allow for any $t_1 \in \mathbb{G}_1, t_2 \in \mathbb{G}_2$, $e(t_1^{h_k s_j}, t_2^{h_{k'} r_i}) = e(t_1^{h_{k'} s_j}, t_2^{h_k r_i})$. However, when translating to our prime-order setting using our rules so far, an analogous mechanism would not hold as we can see that:

$$e(g_1^{\boldsymbol{H}_k\boldsymbol{B}\left(\begin{smallmatrix}\boldsymbol{s}_j\\\hat{s}_j\end{smallmatrix}\right)}, g_2^{\boldsymbol{H}_{k'}^\top\boldsymbol{Z}\left(\begin{smallmatrix}\boldsymbol{r}_i\\\hat{r}_i\end{smallmatrix}\right)}) \neq e(g_1^{\boldsymbol{H}_{k'}\boldsymbol{B}\left(\begin{smallmatrix}\boldsymbol{s}_j\\\hat{s}_j\end{smallmatrix}\right)}, g_2^{\boldsymbol{H}_k^\top\boldsymbol{Z}\left(\begin{smallmatrix}\boldsymbol{r}_i\\\hat{r}_i\end{smallmatrix}\right)}), \tag{6}$$

as $e(g_1,g_2)^{\left((\boldsymbol{r}_i^\top\ \hat{r}_i)\boldsymbol{Z}^\top\boldsymbol{H}_{k'}\right)\left(\boldsymbol{H}_k\boldsymbol{B}\binom{\boldsymbol{s}_j}{\hat{s}_j}\right)} \neq e(g_1,g_2)^{\left((\boldsymbol{r}_i^\top\ \hat{r}_i)\boldsymbol{Z}^\top\boldsymbol{H}_k\right)\left(\boldsymbol{H}_{k'}\boldsymbol{B}\binom{\boldsymbol{s}_j}{\hat{s}_j}\right)}$, due to the fact that the matrix multiplication is not commutative. However, we will *not* use this commutati-vity-based computation in our prime-order framework by disallowing exactly this kind of multiplication to occur. We enable this by the definition of *regular encoding* (the first rule), which exactly prevents multiplication of term $h_k s_j$ with $h_{k'} r_{j'}$.

4.2 Our Generic Construction for Fully Secure ABE

We are now ready to describe our generic construction in prime-order groups. It is obtained by translating the composite-order scheme of [1], recapped in §C, to the prime-order setting using the above rules. As a caveat, we actually use a variant of [1], see Remark 2. From a pair encoding scheme P for a predicate R, we construct an ABE scheme for the predicate R, denoted ABE(P), as follows.

- Setup$(1^\lambda,\kappa)$: Run $(\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T,e,p)\xleftarrow{\$}\mathcal{G}(\lambda)$. Pick generators $g_1\xleftarrow{\$}\mathbb{G}_1$ and $g_2\xleftarrow{\$}\mathbb{G}_2$. Run $n\leftarrow\mathsf{Param}(\kappa)$. Pick $\boldsymbol{H}_1,\ldots,\boldsymbol{H}_n\xleftarrow{\$}\mathbb{Z}_p^{(d+1)\times(d+1)}$ and $\boldsymbol{B}\xleftarrow{\$}\mathbb{GL}_{p,d+1}\subset\mathbb{Z}_p^{(d+1)\times(d+1)}$. Choose $\tilde{\boldsymbol{D}}\xleftarrow{\$}\mathbb{GL}_{p,d}$, define $\boldsymbol{D}:=\begin{pmatrix}\tilde{\boldsymbol{D}} & \boldsymbol{0}\\ \boldsymbol{0} & 1\end{pmatrix}\in\mathbb{GL}_{p,d+1}$ and $\boldsymbol{Z}:=\boldsymbol{B}^{-\top}\boldsymbol{D}$. Choose $\boldsymbol{\alpha}\xleftarrow{\$}\mathbb{Z}_p^{(d+1)\times 1}$. Output

$$\begin{aligned}\mathsf{PK}&=\left(\mathbb{G},e(g_1,g_2)^{\boldsymbol{\alpha}\boldsymbol{B}\binom{\boldsymbol{I}_d}{0}},g_1^{\boldsymbol{B}\binom{\boldsymbol{I}_d}{0}},g_1^{\boldsymbol{H}_1\boldsymbol{B}\binom{\boldsymbol{I}_d}{0}},\ldots,g_1^{\boldsymbol{H}_n\boldsymbol{B}\binom{\boldsymbol{I}_d}{0}}\right),\\ \mathsf{MSK}&=\left(g_2^{\boldsymbol{\alpha}},\quad g_2^{\boldsymbol{Z}\binom{\boldsymbol{I}_d}{0}},g_2^{\boldsymbol{H}_1^\top\boldsymbol{Z}\binom{\boldsymbol{I}_d}{0}},\ldots,g_2^{\boldsymbol{H}_n^\top\boldsymbol{Z}\binom{\boldsymbol{I}_d}{0}}\right).\end{aligned}\tag{7}$$

- Encrypt(Y,M,PK): Upon input $Y\in\mathbb{Y}$, run $(\boldsymbol{c};w_2)\leftarrow\mathsf{Enc2}(Y)$. Pick $\boldsymbol{s}_0,\boldsymbol{s}_1,\ldots,\boldsymbol{s}_{w_2}\xleftarrow{\$}\mathbb{Z}_p^{d\times1}$. Let $\boldsymbol{S}:=\left(\binom{\boldsymbol{s}_0}{0},\binom{\boldsymbol{s}_1}{0},\ldots,\binom{\boldsymbol{s}_{w_2}}{0}\right)\in(\mathbb{Z}_p^{(d+1)\times1})^{w_2+1}$. Denote $\mathbb{H}:=(\boldsymbol{H}_1,\ldots,\boldsymbol{H}_n)$. Output the ciphertext as $\mathsf{CT}=(\boldsymbol{C},C_0)$:

$$\boldsymbol{C}=g_1^{\boldsymbol{c}_{\boldsymbol{B}}(\boldsymbol{S},\mathbb{H})}\in(\mathbb{G}_1^{(d+1)\times1})^{w_1},\qquad C_0=e(g_1,g_2)^{\boldsymbol{\alpha}^\top\boldsymbol{B}\binom{\boldsymbol{s}_0}{0}}\cdot M\in\mathbb{G}_T.\tag{8}$$

  Note that $\boldsymbol{C}$ can be computed from PK since

$$\boldsymbol{c}_{\boldsymbol{B}}(\boldsymbol{S},\mathbb{H})=\left\{\left(\sum_{j\in[0,w_2]}a_{i,j}\boldsymbol{B}\binom{\boldsymbol{s}_j}{0}\right)+\left(\sum_{\substack{j\in[0,w_2]\\k\in[1,n]}}a_{i,j,k}\boldsymbol{H}_k\boldsymbol{B}\binom{\boldsymbol{s}_j}{0}\right)\right\}_{i\in[1,w_1]}\tag{9}$$

  and thanks to the identity relation $\left(\boldsymbol{X}\binom{\boldsymbol{I}_d}{0}\right)\boldsymbol{y}=\boldsymbol{X}\binom{\boldsymbol{y}}{0}$ for any $\boldsymbol{X}\in\mathbb{Z}_p^{(d+1)\times(d+1)}$, $\boldsymbol{y}\in\mathbb{Z}_p^{d\times1}$.

- KeyGen(X,MSK): Upon input $X\in\mathbb{X}$, run $(\boldsymbol{k};m_2)\leftarrow\mathsf{Enc1}(X)$. Randomly pick $\boldsymbol{r}_1,\ldots,\boldsymbol{r}_{m_2}\xleftarrow{\$}\mathbb{Z}_p^{d\times1}$. Let $\boldsymbol{R}:=\left(\binom{\boldsymbol{r}_1}{0},\ldots,\binom{\boldsymbol{r}_{m_2}}{0}\right)\in(\mathbb{Z}_p^{(d+1)\times1})^{m_2}$. Recall the notation $\mathbb{H}=(\boldsymbol{H}_1,\ldots,\boldsymbol{H}_n)$. Output

$$\mathsf{SK}=g_2^{\boldsymbol{k}_{\boldsymbol{Z}}(\boldsymbol{\alpha},\boldsymbol{R},\mathbb{H})}\in(\mathbb{G}_2^{(d+1)\times1})^{m_1}.\tag{10}$$

Note that SK can be computed from MSK since

$$\boldsymbol{k}_{\boldsymbol{Z}}(\boldsymbol{\alpha}, \boldsymbol{R}, \mathbb{H}) = \left\{ b_i \boldsymbol{\alpha} + \left(\sum_{j \in [1,m_2]} b_{i,j} \boldsymbol{Z} \left(\begin{smallmatrix} \boldsymbol{r}_j \\ 0 \end{smallmatrix} \right) \right) + \left(\sum_{\substack{j \in [1,m_2] \\ k \in [1,n]}} b_{i,j,k} \boldsymbol{H}_k^\top \boldsymbol{Z} \left(\begin{smallmatrix} \boldsymbol{r}_j \\ 0 \end{smallmatrix} \right) \right) \right\}_{i \in [1,m_1]} \tag{11}$$

and thanks again to the identity relation above.

- $\mathsf{Decrypt}(\mathsf{CT}, \mathsf{SK})$: Obtain Y, X from CT, SK. Suppose $R(X,Y) = 1$. Run $\boldsymbol{E} \leftarrow \mathsf{Pair}(X,Y)$. Denote by $\boldsymbol{C}[i']$ the i'-th element in $\boldsymbol{C}$, and $\mathsf{SK}[i]$ the i-th element in SK. Compute

$$e(g_1, g_2)^{\boldsymbol{\alpha}^\top \boldsymbol{B} \left(\begin{smallmatrix} \boldsymbol{s}_0 \\ 0 \end{smallmatrix} \right)} \in \mathbb{G}_T \leftarrow \prod_{i \in [1,m_1], i' \in [1,w_1]} e(\boldsymbol{C}[i'], \mathsf{SK}[i])^{E_{i,i'}},$$

  and obtain $M \leftarrow C_0 / e(g_1, g_2)^{\boldsymbol{\alpha}^\top \boldsymbol{B} \left(\begin{smallmatrix} \boldsymbol{s}_0 \\ 0 \end{smallmatrix} \right)}$.

Correctness. We would like to prove that if $R(X,Y) = 1$ then

$$\boldsymbol{\alpha}^\top \boldsymbol{B} \left(\begin{smallmatrix} \boldsymbol{s}_0 \\ 0 \end{smallmatrix} \right) = \sum_{i \in [1,m_1], i' \in [1,w_1]} E_{i,i'} \cdot (\boldsymbol{k}_{\boldsymbol{Z}}(\boldsymbol{\alpha}, \boldsymbol{R}, \mathbb{H})[i])^\top \cdot \boldsymbol{c}_{\boldsymbol{B}}(\boldsymbol{S}, \mathbb{H})[i'].$$

This is implied from the correctness of the pair encoding which states that: if $R(X,Y) = 1$, then $\alpha s_0 = \sum_{i \in [1,m_1], i' \in [1,w_1]} E_{i,i'} \cdot \boldsymbol{k}(\alpha, \boldsymbol{r}, \boldsymbol{h})[i] \cdot \boldsymbol{c}(\boldsymbol{s}, \boldsymbol{h})[i']$. Intuitively, since we translate to the prime-order setting by substituting variables and procedures while preserving their properties as in §4.1, this relation should also translate to the above equation. In particular, we use associativity but not use commutativity, as clarified in §4.1. We verify the correctness more formally in §D.

5 Security Theorems and Proofs

We obtain three security theorems for the generic construction. The first one is the main theorem and is for the case when the pair encoding is $(1, \mathsf{poly})$-SMH and $(1,1)$-CMH, where we achieve tighter reduction cost, $O(q_1)$. The other two are for the case of PMH and the pair of $(1,1)$-SMH, $(1,1)$-CMH, where we obtain normal reduction cost, $O(q_{\text{all}})$. We postpone the latter two to §E.

Theorem 3. *Suppose that a pair encoding scheme* P *for predicate* R *is* $(1, \mathsf{poly})$*-selectively and* $(1,1)$*-co-selectively master-key hiding in* $\mathcal{G}$*, and the Matrix-DH Assumption holds in* $\mathcal{G}$*. Then the construction* $\mathsf{ABE}(\mathsf{P})$ *in* $\mathcal{G}$ *is fully secure. More precisely, for any PPT adversary* $\mathcal{A}$*, let* q_1 *denote the number of queries in phase 1, there exist PPT algorithms* $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3$*, whose running times are the same as* $\mathcal{A}$ *plus some polynomial times, such that for any* λ*,*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ABE}}(\lambda) \leq (2q_1 + 3)\mathsf{Adv}_{\mathcal{B}_1}^{\mathcal{D}_d\text{-}\mathsf{MatDH}}(\lambda) + q_1 \mathsf{Adv}_{\mathcal{B}_2}^{(1,1)\text{-}\mathsf{CMH}}(\lambda) + \mathsf{Adv}_{\mathcal{B}_3}^{(1,\mathsf{poly})\text{-}\mathsf{SMH}}(\lambda),$$

Semi-functional Algorithms. We define semi-functional algorithms which will be used in the security proof. These are also translated from semi-functional algorithms from the framework of [1], recapped in §C.

- $\mathsf{SFSetup}(1^\lambda, \kappa) \to (\mathsf{PK}, \mathsf{MSK}, \widehat{\mathsf{PK}}, \widehat{\mathsf{MSK}}_{\mathsf{base}}, \widehat{\mathsf{MSK}}_{\mathsf{aux}})$: This is exactly the same as the Setup algorithm albeit it additionally outputs also $\widehat{\mathsf{PK}}$, $\widehat{\mathsf{MSK}}_{\mathsf{base}}$, $\widehat{\mathsf{MSK}}_{\mathsf{aux}}$ defined as

$$\widehat{\mathsf{PK}} = \left(e(g_1, g_2)^{\boldsymbol{\alpha}^\top \boldsymbol{B}\binom{\mathbf{0}}{1}}, g_1^{\boldsymbol{B}\binom{\mathbf{0}}{1}}, g_1^{\boldsymbol{H}_1\boldsymbol{B}\binom{\mathbf{0}}{1}}, \dots, g_1^{\boldsymbol{H}_n\boldsymbol{B}\binom{\mathbf{0}}{1}} \right), \tag{12}$$

$$\widehat{\mathsf{MSK}}_{\mathsf{base}} = g_2^{\boldsymbol{Z}\binom{\mathbf{0}}{1}}, \qquad \widehat{\mathsf{MSK}}_{\mathsf{aux}} = \left(g_2^{\boldsymbol{H}_1^\top \boldsymbol{Z}\binom{\mathbf{0}}{1}}, \dots, g_2^{\boldsymbol{H}_n^\top \boldsymbol{Z}\binom{\mathbf{0}}{1}} \right). \tag{13}$$

- $\mathsf{SFEncrypt}(Y, M, \mathsf{PK}, \widehat{\mathsf{PK}}) \to \mathsf{CT}$: Upon inputs Y, M, PK and $\widehat{\mathsf{PK}}$, first run $(\boldsymbol{c}; w_2) \leftarrow \mathsf{Enc2}(Y)$. Pick $\boldsymbol{s}_0, \boldsymbol{s}_1, \dots, \boldsymbol{s}_{w_2} \xleftarrow{\$} \mathbb{Z}_p^{d\times 1}$, $\hat{s}_0, \hat{s}_1, \dots, \hat{s}_{w_2} \xleftarrow{\$} \mathbb{Z}_p$. Let

$$\boldsymbol{S} := \left(\binom{\boldsymbol{s}_0}{0}, \binom{\boldsymbol{s}_1}{0}, \dots, \binom{\boldsymbol{s}_{w_2}}{0} \right), \hat{\boldsymbol{S}} := \left(\binom{\mathbf{0}}{\hat{s}_0}, \binom{\mathbf{0}}{\hat{s}_1}, \dots, \binom{\mathbf{0}}{\hat{s}_{w_2}} \right) \in (\mathbb{Z}_p^{(d+1)\times 1})^{w_2+1}$$

Output the ciphertext as $\mathsf{CT} = (\boldsymbol{C}, C_0)$:

$$\boldsymbol{C} = g_1^{\boldsymbol{c}_{\boldsymbol{B}}(\boldsymbol{S},\mathbb{H}) + \boldsymbol{c}_{\boldsymbol{B}}(\hat{\boldsymbol{S}},\mathbb{H})} \in (\mathbb{G}_1^{(d+1)\times 1})^{w_1}, \qquad C_0 = e(g_1, g_2)^{\boldsymbol{\alpha}^\top \boldsymbol{B}\binom{\boldsymbol{s}_0}{\hat{s}_0}} \cdot M \in \mathbb{G}_T. \tag{14}$$

Note that $\boldsymbol{C}$ can be computed from PK and $\widehat{\mathsf{PK}}$ since

$$\boldsymbol{c}_{\boldsymbol{B}}(\boldsymbol{S}, \mathbb{H}) + \boldsymbol{c}_{\boldsymbol{B}}(\hat{\boldsymbol{S}}, \mathbb{H}) = \left\{ \left(\sum_{j\in[0,w_2]} a_{i,j}\boldsymbol{B}\binom{\boldsymbol{s}_j}{\hat{s}_j} \right) + \left(\sum_{\substack{j\in[0,w_2]\\ k\in[1,n]}} a_{i,j,k}\boldsymbol{H}_k\boldsymbol{B}\binom{\boldsymbol{s}_j}{\hat{s}_j} \right) \right\}_{i\in[1,w_1]}$$

and thanks to the identity relation $\left(\boldsymbol{X}\binom{\boldsymbol{I}_d}{0}\right)\boldsymbol{y} + \left(\boldsymbol{X}\binom{\mathbf{0}}{1}\right)\hat{y} = \boldsymbol{X}\binom{\boldsymbol{y}}{\hat{y}}$.

- $\mathsf{SFKeyGen}\big(X, \mathsf{MSK}, \widehat{\mathsf{MSK}}_{\mathsf{base}}, \widehat{\mathsf{MSK}}_{\mathsf{aux}}, \mathsf{type} \in \{0,1,2,3\}, \beta \in \mathbb{Z}_p\big) \to \mathsf{SK}$: Upon inputs X, MSK, $\widehat{\mathsf{MSK}}_{\mathsf{base}}, \widehat{\mathsf{MSK}}_{\mathsf{aux}}, \mathsf{type} \in \{0,1,2,3\}, \beta \in \mathbb{Z}_p$, first run $(\boldsymbol{k}; m_2) \leftarrow \mathsf{Enc1}(X)$. Pick $\boldsymbol{r}_1, \dots, \boldsymbol{r}_{m_2} \xleftarrow{\$} \mathbb{Z}_p^{d\times 1}$ and $\hat{r}_1, \dots, \hat{r}_{m_2} \xleftarrow{\$} \mathbb{Z}_p$. Let

$$\boldsymbol{R} := \left(\binom{\boldsymbol{r}_1}{0}, \dots, \binom{\boldsymbol{r}_{m_2}}{0} \right), \hat{\boldsymbol{R}} := \left(\binom{\mathbf{0}}{\hat{r}_1}, \dots, \binom{\mathbf{0}}{\hat{r}_{m_2}} \right) \in (\mathbb{Z}_p^{(d+1)\times 1})^{m_2}$$

Output the secret key SK:

$$\mathsf{SK} = \begin{cases} g_2^{\boldsymbol{k}_{\boldsymbol{Z}}(\boldsymbol{\alpha},\boldsymbol{R},\mathbb{H})} & \text{if } \mathsf{type} = 0 \quad (15) \\ g_2^{\boldsymbol{k}_{\boldsymbol{Z}}(\boldsymbol{\alpha},\boldsymbol{R},\mathbb{H}) + \boldsymbol{k}_{\boldsymbol{Z}}(\ \mathbf{0}\ ,\hat{\boldsymbol{R}},\mathbb{H})} & \text{if } \mathsf{type} = 1 \quad (16) \\ g_2^{\boldsymbol{k}_{\boldsymbol{Z}}(\boldsymbol{\alpha},\boldsymbol{R},\mathbb{H}) + \boldsymbol{k}_{\boldsymbol{Z}}(\boldsymbol{Z}\binom{\mathbf{0}}{\beta},\hat{\boldsymbol{R}},\mathbb{H})} & \text{if } \mathsf{type} = 2 \quad (17) \\ g_2^{\boldsymbol{k}_{\boldsymbol{Z}}(\boldsymbol{\alpha},\boldsymbol{R},\mathbb{H}) + \boldsymbol{k}_{\boldsymbol{Z}}(\boldsymbol{Z}\binom{\mathbf{0}}{\beta},\mathbf{0},\mathbb{H})} & \text{if } \mathsf{type} = 3 \quad (18) \end{cases}$$

Note that SK of each type can be computed from $\mathsf{MSK}, \widehat{\mathsf{MSK}}_{\mathsf{base}}, \widehat{\mathsf{MSK}}_{\mathsf{aux}}$ since $\boldsymbol{k}_{\boldsymbol{Z}}(\boldsymbol{\alpha}, \boldsymbol{R}, \mathbb{H}) + \boldsymbol{k}_{\boldsymbol{Z}}(\boldsymbol{Z}\binom{\mathbf{0}}{\beta}, \hat{\boldsymbol{R}}, \mathbb{H}) =$

$$\left\{ b_i\boldsymbol{\alpha} + b_i\boldsymbol{Z}\binom{\mathbf{0}}{\beta} + \left(\sum_{j\in[1,m_2]} b_{i,j}\boldsymbol{Z}\binom{\boldsymbol{r}_j}{\hat{r}_j} \right) + \left(\sum_{\substack{j\in[1,m_2]\\ k\in[1,n]}} b_{i,j,k}\boldsymbol{H}_k^\top\boldsymbol{Z}\binom{\boldsymbol{r}_j}{\hat{r}_j} \right) \right\}_{i\in[1,m_1]}$$

and thanks again to the identity relation above. Furthermore, we note that:

Figure 1: The sequence of games in the security proof.

G_0	:Modify $^{(0)}$	$\boxed{\mathsf{SFsetup}(1^\lambda,\kappa) \to (\mathsf{PK},\mathsf{MSK},\widehat{\mathsf{PK}},\widehat{\mathsf{MSK}}_{\mathsf{base}},\widehat{\mathsf{MSK}}_{\mathsf{aux}})}$.
	Modify $^{(2)}$	$\boxed{\boldsymbol{C}^\star \leftarrow \mathsf{SFEncrypt}(Y^\star, M_b, \mathsf{PK}, \widehat{\mathsf{PK}})}$.
$\mathsf{G}_{i,1}$	:Modify $^{(1)}$	$\boxed{\beta_j \xleftarrow{\$} \mathbb{Z}_p,\ \mathsf{SK}_j \leftarrow \begin{cases} \mathsf{SFKeyGen}(X_j,\mathsf{MSK},\widehat{\mathsf{MSK}}_{\mathsf{base}}, \quad - \quad ,3,\beta_j) & \text{if } j < i \\ \mathsf{SFKeyGen}(X_j,\mathsf{MSK},\widehat{\mathsf{MSK}}_{\mathsf{base}},\widehat{\mathsf{MSK}}_{\mathsf{aux}},1,-) & \text{if } j = i \\ \quad \mathsf{KeyGen}(X_j,\mathsf{MSK}) & \text{if } j > i \end{cases}}$
$\mathsf{G}_{i,2}$	:Modify $^{(1)}$	$\boxed{\beta_j \xleftarrow{\$} \mathbb{Z}_p,\ \mathsf{SK}_j \leftarrow \begin{cases} \mathsf{SFKeyGen}(X_j,\mathsf{MSK},\widehat{\mathsf{MSK}}_{\mathsf{base}}, \quad - \quad ,3,\beta_j) & \text{if } j < i \\ \mathsf{SFKeyGen}(X_j,\mathsf{MSK},\widehat{\mathsf{MSK}}_{\mathsf{base}},\widehat{\mathsf{MSK}}_{\mathsf{aux}},2,\beta_j) & \text{if } j = i \\ \quad \mathsf{KeyGen}(X_j,\mathsf{MSK}) & \text{if } j > i \end{cases}}$
$\mathsf{G}_{i,3}$	:Modify $^{(1)}$	$\boxed{\beta_j \xleftarrow{\$} \mathbb{Z}_p,\ \mathsf{SK}_j \leftarrow \begin{cases} \mathsf{SFKeyGen}(X_j,\mathsf{MSK},\widehat{\mathsf{MSK}}_{\mathsf{base}}, \quad - \quad ,3,\beta_j) & \text{if } j \le i \\ \quad \mathsf{KeyGen}(X_j,\mathsf{MSK}) & \text{if } j > i \end{cases}}$
G_{q_1+1}	:Modify $^{(3)}$	$\boxed{\mathsf{SK}_j \leftarrow \quad \mathsf{SFKeyGen}(X_j,\mathsf{MSK},\widehat{\mathsf{MSK}}_{\mathsf{base}},\widehat{\mathsf{MSK}}_{\mathsf{aux}},1,-)}$
G_{q_1+2}	:Insert	$\boxed{\beta \xleftarrow{\$} \mathbb{Z}_p}$ at the begin of Phase 2.
	Modify $^{(3)}$	$\boxed{\mathsf{SK}_j \leftarrow \quad \mathsf{SFKeyGen}(X_j,\mathsf{MSK},\widehat{\mathsf{MSK}}_{\mathsf{base}},\widehat{\mathsf{MSK}}_{\mathsf{aux}},2,\beta)}$
G_{q_1+3}	:Modify $^{(3)}$	$\boxed{\mathsf{SK}_j \leftarrow \quad \mathsf{SFKeyGen}(X_j,\mathsf{MSK},\widehat{\mathsf{MSK}}_{\mathsf{base}}, \quad - \quad ,3,\beta)}$
$\mathsf{G}_{\text{final}}$	:Modify $^{(2)}$	$\boxed{M \xleftarrow{\$} \mathcal{M},\ \boldsymbol{C}^\star \leftarrow \mathsf{SFEncrypt}(Y^\star, M, \mathsf{PK}, \widehat{\mathsf{PK}})}$.

– In computing type $0, 3$, $\widehat{\mathsf{MSK}}_{\mathsf{aux}}$ is not required as input (and no $\hat{\boldsymbol{R}}$ needed).

– In computing type $0, 1$, β is not required as input.

Proof of Theorem 3. We use a sequence of games in the following order:

$$\begin{tikzcd}
\mathsf{G}_{\text{real}} \arrow[r, "\begin{matrix}\mathsf{MatDH}\\ 4\end{matrix}"'] & \mathsf{G}_0 \arrow[r] & \mathsf{G}_{1,1} \arrow[r] & \cdots \arrow[r] & \mathsf{G}_{i-1,3} \arrow[r, "\begin{matrix}\mathsf{MatDH}\\ 5\end{matrix}"'] & \mathsf{G}_{i,1} \arrow[r, "\begin{matrix}\mathsf{CMH}\\ 7\end{matrix}"'] & \mathsf{G}_{i,2} \arrow[r, "\begin{matrix}\mathsf{MatDH}\\ 8\end{matrix}"'] & \mathsf{G}_{i,3} \arrow[r] & \cdots \arrow[r] & \mathsf{G}_{q_1,3} \arrow[r, "\begin{matrix}\mathsf{MatDH}\\ 9\end{matrix}"'] & \mathsf{G}_{q_1+1} \arrow[r, "\begin{matrix}\mathsf{SMH}\\ 10\end{matrix}"'] & \mathsf{G}_{q_1+2} \arrow[r, "\begin{matrix}\mathsf{MatDH}\\ 11\end{matrix}"'] & \mathsf{G}_{q_1+3} \arrow[r, "\begin{matrix}=\\ 12\end{matrix}"'] & \mathsf{G}_{\text{final}}
\end{tikzcd}$$

where each game is defined as follows. G_{real} is the actual security game, as defined in §A. Each of the following game is defined exactly as *its previous game* in the sequence except the specified modification as follows. For notational purpose, let $\mathsf{G}_{0,3} := \mathsf{G}_0$.

– G_0: We modify the challenge ciphertext to be semi-functional type.

– $\mathsf{G}_{i,t}$ where $i \in [1, q_1]$, $t \in \{1, 2, 3\}$: We modify the i-th queried key to be semi-functional of type-t. We use fresh β for each key (for type $t = 2, 3$).

– G_{q_1+t} where $t \in \{1, 2, 3\}$: We modify all keys in phase 2 to be semi-functional of type-t at once. We use the same β for all these keys (for type $t = 2, 3$).

– $\mathsf{G}_{\text{final}}$: We modify the challenge to encrypt a random message.

More formal definitions of games are depicted in Fig. 1, where each box shows each game modification and the box number shows the place of modification in the security game in §A.

In the final game, the advantage of $\mathcal{A}$ is trivially 0. We prove the indistinguishability between all these adjacent games (under the underlying assumptions as written in the diagram). These comprise Lemma 4,5,7,8,9,10,11,12 (also depicted in the diagram). The proofs of these are shown below. From these, we obtain Theorem 3. $\square$

The following remark describes how our framework deviates from the framework of [1].

Remark 2. In translating to the prime-order setting, we actually start from a *variant* of [1], where we also incorporate a technique from [51] related to the information-theoretic argument for the final transition. This has an advantage over [1] in that we can eliminate a computational assumption for this transition. To enable this, we have used $\boldsymbol{\alpha} \in \mathbb{Z}_p^{(d+1)\times 1}$, instead of an element from the left subspace, say $\boldsymbol{Z}\binom{\boldsymbol{\alpha}'}{0}$, for some $\boldsymbol{\alpha}' \in \mathbb{Z}_p^{d\times 1}$ (as would be used if [1] were used), to define $\boldsymbol{k_Z}(\boldsymbol{\alpha}, \boldsymbol{R}, \mathbb{H})$ in keys as in Eq. (11) and C_0 in ciphertexts as in Eq. (8), (14). Also, we have defined the message mask element of semi-functional ciphertext in Eq. (14) to contain $\binom{\boldsymbol{s}_0}{\hat{s}_0}$, instead of $\binom{\boldsymbol{s}_0}{0}$.

In the following subsections, we prove the distinguishability between the consecutive games. We define $\mathsf{G}_j\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ABE}}(\lambda)$ to be the advantage of $\mathcal{A}$ in the game G_j.

5.1 Normal to Semi-functional Ciphertext

Lemma 4 ($\mathsf{G}_{\mathrm{real}}$ to G_0)**.** *For any adversary $\mathcal{A}$ against* ABE*, there exists an algorithm $\mathcal{B}$ that breaks the $\mathcal{D}_d$-Matrix-DH Assumption with* $|\mathsf{G}_{\mathrm{real}}\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ABE}}(\lambda) - \mathsf{G}_0\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ABE}}(\lambda)| \le \mathsf{Adv}_{\mathcal{B}}^{\mathcal{D}_d\text{-}\mathsf{MatDH}}(\lambda)$.

Proof. The algorithm $\mathcal{B}$ obtains an input $(\mathbb{G}, g_1^{\boldsymbol{T}}, g_1^{\boldsymbol{T}\binom{\boldsymbol{y}}{\hat{y}}})$ from the $\mathcal{D}_d$-Matrix DH Assumption where either $\hat{y} = 0$ or $\hat{y} \xleftarrow{\$} \mathbb{Z}_p$, and $\boldsymbol{T} \xleftarrow{\$} \mathcal{D}_d$, $\boldsymbol{y} \xleftarrow{\$} \mathbb{Z}_p^{d\times 1}$.

Setup. The algorithm $\mathcal{B}$ does exactly the same as $\mathsf{SFSetup}(1^\lambda, \kappa)$ except that it uses $\mathbb{G}$ from its input, and that it will set $\boldsymbol{B}$ and $\boldsymbol{D}$ in an implicit way. $\mathsf{PK}, \mathsf{MSK}, \widehat{\mathsf{PK}}$ will be determined from this implicit programming. Note that $\widehat{\mathsf{MSK}}_{\mathsf{base}}, \widehat{\mathsf{MSK}}_{\mathsf{aux}}$ are also determined from this but not computable; nevertheless, they are not used in $\mathsf{G}_{\mathrm{real}}$ or G_0. $\mathcal{B}$ begins by choosing $\tilde{\boldsymbol{B}} \xleftarrow{\$} \mathbb{GL}_{p,d+1}$, $\boldsymbol{J} \xleftarrow{\$} \mathbb{GL}_{p,d}$ and *implicitly* setting [9]

$$\boldsymbol{B} = \tilde{\boldsymbol{B}}\boldsymbol{T}, \qquad\qquad \boldsymbol{Z} = \tilde{\boldsymbol{B}}^{-\top}\tilde{\boldsymbol{Z}} := (\tilde{\boldsymbol{B}}^{-\top}) \begin{array}{c c} & \begin{array}{cc} d & 1 \end{array} \\ \begin{array}{c} d \\ 1 \end{array} & \begin{pmatrix} \boldsymbol{J} & -\boldsymbol{M}^{-\top}\boldsymbol{c}^\top \\ \boldsymbol{0} & 1 \end{pmatrix} \end{array},$$

where we recall that $\boldsymbol{T} = \begin{pmatrix} \boldsymbol{M} & \boldsymbol{0} \\ \boldsymbol{c} & 1 \end{pmatrix}$ from Eq.(2). From this, we have

$$\boldsymbol{D} = \boldsymbol{B}^\top\boldsymbol{Z} = (\boldsymbol{T}^\top\tilde{\boldsymbol{B}}^\top)(\tilde{\boldsymbol{B}}^{-\top}\tilde{\boldsymbol{Z}}) = \boldsymbol{T}^\top\tilde{\boldsymbol{Z}}$$

$$= \begin{array}{c c} & \begin{array}{cc} d & 1 \end{array} \\ \begin{array}{c} d \\ 1 \end{array} & \begin{pmatrix} \boldsymbol{M}^\top & \boldsymbol{c}^\top \\ \boldsymbol{0} & 1 \end{pmatrix} \end{array} \begin{array}{c} \begin{array}{cc} d & 1 \end{array} \\ \begin{pmatrix} \boldsymbol{J} & -\boldsymbol{M}^{-\top}\boldsymbol{c}^\top \\ \boldsymbol{0} & 1 \end{pmatrix} \end{array} = \begin{array}{c c} & \begin{array}{cc} d & 1 \end{array} \\ \begin{array}{c} d \\ 1 \end{array} & \begin{pmatrix} \boldsymbol{M}^\top\boldsymbol{J} & \boldsymbol{0} \\ \boldsymbol{0} & 1 \end{pmatrix} \end{array},$$

[9]Here, we write dimensions of sub-matrices for ease of viewing.

where we can verify that the upper right block is $\mathbf{0}$ by seeing that $(\boldsymbol{M}^\top)(-\boldsymbol{M}^{-\top}\boldsymbol{c}^\top) + (\boldsymbol{c}^\top)(1) = \mathbf{0}$. $\boldsymbol{B}$ is properly distributed due to the randomness of $\tilde{\boldsymbol{B}}$ and that $\tilde{\boldsymbol{B}}, \boldsymbol{T} \in \mathbb{GL}_{p,d+1}$. $\boldsymbol{D}$ is properly distributed due to the randomness of $\boldsymbol{J}$ and that $\boldsymbol{M}^\top, \boldsymbol{J} \in \mathbb{GL}_{p,d}$. $\mathcal{B}$ can then compute

$$g_1^{\boldsymbol{B}} = g_1^{\tilde{\boldsymbol{B}}\boldsymbol{T}}, \qquad\qquad g_2^{\boldsymbol{Z}\binom{\boldsymbol{I}_d}{0}} = g_2^{\tilde{\boldsymbol{B}}^{-\top}\binom{\boldsymbol{J}}{\mathbf{0}}},$$

where the first term is computable from $g_1^{\boldsymbol{T}}$, while in the second term, the unknown last column of $\boldsymbol{Z}$ vanishes through the left projection map, $\binom{\boldsymbol{I}_d}{0}$. From these two terms, $\mathcal{B}$ can compute $\mathsf{PK}, \mathsf{MSK}, \widehat{\mathsf{PK}}$; in particular, $\mathcal{B}$ chooses $\boldsymbol{\alpha} \xleftarrow{\$} \mathbb{Z}_p^{(d+1)\times 1}$, $\boldsymbol{H}_1, \ldots, \boldsymbol{H}_n \xleftarrow{\$} \mathbb{Z}_p^{(d+1)\times(d+1)}$, and computes Eq.(7),(12). (We note that $\widehat{\mathsf{PK}}$ is not used in the simulation though).

Phase 1. When $\mathcal{A}$ makes the j-th key query for X_j, $\mathcal{B}$ generates a key $\mathsf{SK} \leftarrow \mathsf{KeyGen}(X_j, \mathsf{MSK})$.

Challenge. The adversary $\mathcal{A}$ outputs messages $M_0, M_1 \in \mathbb{G}_T$ along with a target $Y^\star$. $\mathcal{B}$ chooses $b \xleftarrow{\$} \{0,1\}$. $\mathcal{B}$ extends the Matrix-DH Assumption to (w_2+1)-fold by random self reducibility and obtains $(g_1^{\boldsymbol{T}}, g_1^{\boldsymbol{T}\binom{\boldsymbol{Y}}{\hat{\boldsymbol{y}}}})$ where either $\hat{\boldsymbol{y}} = \mathbf{0}$ or $\hat{\boldsymbol{y}} \xleftarrow{\$} \mathbb{Z}_p^{1\times(w_2+1)}$ with $\boldsymbol{T} \xleftarrow{\$} \mathcal{D}_d$, $\boldsymbol{Y} \xleftarrow{\$} \mathbb{Z}_p^{d\times(w_2+1)}$. $\mathcal{B}$ implicitly sets $\boldsymbol{S} + \hat{\boldsymbol{S}} = \binom{\boldsymbol{Y}}{\hat{\boldsymbol{y}}}$; that is,

$$\boldsymbol{S} + \hat{\boldsymbol{S}} = \begin{array}{c} \\ d \\ 1 \end{array}\!\!\begin{array}{c} \begin{array}{cccc} 1 & 1 & & 1 \end{array} \\ \begin{pmatrix} \boldsymbol{s}_0 & \boldsymbol{s}_1 & \cdots & \boldsymbol{s}_{w_2} \\ \hat{s}_0 & \hat{s}_1 & \cdots & \hat{s}_{w_2} \end{pmatrix} \end{array} = \begin{pmatrix} \boldsymbol{Y} \\ \hat{\boldsymbol{y}} \end{pmatrix} = \begin{array}{c} \\ d \\ 1 \end{array}\!\!\begin{array}{c} \begin{array}{cccc} 1 & 1 & & 1 \end{array} \\ \begin{pmatrix} \boldsymbol{y}_0 & \boldsymbol{y}_1 & \cdots & \boldsymbol{y}_{w_2} \\ \hat{y}_0 & \hat{y}_1 & \cdots & \hat{y}_{w_2} \end{pmatrix} \end{array},$$

where we denote $\binom{\boldsymbol{y}_j}{\hat{y}_j}$ as the j-th column of $\binom{\boldsymbol{Y}}{\hat{\boldsymbol{y}}}$. Thus, $\binom{\boldsymbol{s}_j}{\hat{s}_j} = \binom{\boldsymbol{y}_j}{\hat{y}_j}$. $\mathcal{B}$ then can compute for each $j \in [0, w_2]$,

$$g_1^{\boldsymbol{B}\binom{\boldsymbol{s}_j}{\hat{s}_j}} = g_1^{\tilde{\boldsymbol{B}}\boldsymbol{T}\binom{\boldsymbol{s}_j}{\hat{s}_j}} = g_1^{\tilde{\boldsymbol{B}}\boldsymbol{T}\binom{\boldsymbol{y}_j}{\hat{y}_j}}.$$

since $\mathcal{B}$ possesses $g_1^{\boldsymbol{T}\binom{\boldsymbol{Y}}{\hat{\boldsymbol{y}}}}$. From these terms, $\mathcal{B}$ can compute the ciphertext in Eq.(14) since $\mathcal{B}$ possesses $\boldsymbol{\alpha}, \boldsymbol{H}_1, \ldots, \boldsymbol{H}_n$. In particular, for C_0, $\mathcal{B}$ computes $C_0 = e(g_1^{\boldsymbol{\alpha}^\top \boldsymbol{B}\binom{\boldsymbol{s}_0}{\hat{s}_0}}, g_2)$.

Phase 2. $\mathcal{B}$ does the same as in Phase 1.

Guess. The algorithm $\mathcal{B}$ has properly simulated $\mathsf{G}_{\mathrm{real}}$ if $\hat{y} = 0$ and G_0 if $\hat{y} \xleftarrow{\$} \mathbb{Z}_p$. Hence, $\mathcal{B}$ can use the output of $\mathcal{A}$ to break the Matrix DH Assumption. □

5.2 Normal to Type-1 Semi-functional Key in Phase 1

Lemma 5 ($\mathsf{G}_{i-1,3}$ to $\mathsf{G}_{i,1}$)**.** *For any adversary $\mathcal{A}$ against* ABE*, there exists an algorithm $\mathcal{B}$ that breaks the $\mathcal{D}_d$-Matrix-DH Assumption with* $|\mathsf{G}_{i-1,3}\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ABE}}(\lambda) - \mathsf{G}_{i,1}\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ABE}}(\lambda)| \le \mathsf{Adv}_{\mathcal{B}}^{\mathcal{D}_d\text{-}\mathsf{MatDH}}(\lambda)$.

Proof. The algorithm $\mathcal{B}$ obtains an input $(\mathbb{G}, g_2^{\boldsymbol{T}}, g_2^{\boldsymbol{T}\binom{\boldsymbol{y}}{\hat{y}}})$ from the $\mathcal{D}_d$-Matrix DH Assumption where either $\hat{y} = 0$ or $\hat{y} \xleftarrow{\$} \mathbb{Z}_p$, and $\boldsymbol{T} \xleftarrow{\$} \mathcal{D}_d$, $\boldsymbol{y} \xleftarrow{\$} \mathbb{Z}_p^{d\times 1}$. Note that it is wlog that we let the problem instance term be defined over g_2 (instead of g_1).

Setup. The algorithm $\mathcal{B}$ does exactly the same as $\mathsf{SFSetup}(1^\lambda, \kappa)$ except that it uses $\mathbb{G}$ from its input, and that it will set $\boldsymbol{B}$ and $\boldsymbol{D}$ in an implicit way. $\mathsf{PK}, \mathsf{MSK}, \widehat{\mathsf{PK}}, \widehat{\mathsf{MSK}}_{\mathsf{base}}, \widehat{\mathsf{MSK}}_{\mathsf{aux}}$ will be determined from this implicit programming, but only $\widehat{\mathsf{PK}}$ will not be computable. $\mathcal{B}$ begins by choosing $\tilde{\boldsymbol{B}} \xleftarrow{\$} \mathbb{GL}_{p,d+1}$, $\boldsymbol{J} \xleftarrow{\$} \mathbb{GL}_{p,d}$ and implicitly setting

$$\boldsymbol{B} = \tilde{\boldsymbol{B}} \begin{array}{c|cc} & d & 1 \\ \hline d & \boldsymbol{I} & \boldsymbol{M}^{-\top}\boldsymbol{c}^\top \\ 1 & \boldsymbol{0} & -1 \end{array}, \qquad \boldsymbol{D} = \begin{array}{c|cc} & d & 1 \\ \hline d & \boldsymbol{MJ} & \boldsymbol{0} \\ 1 & \boldsymbol{0} & 1 \end{array},$$

where we recall that $\boldsymbol{T} = \left(\begin{smallmatrix} \boldsymbol{M} & \boldsymbol{0} \\ \boldsymbol{c} & 1 \end{smallmatrix}\right)$ from Eq.(2). $\boldsymbol{B}$ is properly distributed due to the uniform randomness of $\tilde{\boldsymbol{B}}$ in $\mathbb{GL}_{p,d+1}$. $\boldsymbol{D}$ is properly distributed due to the randomness of $\boldsymbol{J}$ and that $\boldsymbol{M}, \boldsymbol{J} \in \mathbb{GL}_{p,d}$. From this we have

$$\boldsymbol{B}^{-1} = \begin{array}{c|cc} & d & 1 \\ \hline d & \boldsymbol{I} & \boldsymbol{M}^{-\top}\boldsymbol{c}^\top \\ 1 & \boldsymbol{0} & -1 \end{array} \tilde{\boldsymbol{B}}^{-1},$$

$$\begin{aligned} \boldsymbol{Z} = \boldsymbol{B}^{-\top}\boldsymbol{D} &= \tilde{\boldsymbol{B}}^{-\top} \begin{array}{c|cc} & d & 1 \\ \hline d & \boldsymbol{I} & \boldsymbol{0} \\ 1 & \boldsymbol{c}\boldsymbol{M}^{-1} & -1 \end{array} \begin{pmatrix} \boldsymbol{MJ} & \boldsymbol{0} \\ \boldsymbol{0} & 1 \end{pmatrix} \\ &= \tilde{\boldsymbol{B}}^{-\top} \begin{array}{c|cc} & d & 1 \\ \hline d & \boldsymbol{MJ} & \boldsymbol{0} \\ 1 & \boldsymbol{cJ} & -1 \end{array} = \tilde{\boldsymbol{B}}^{-\top}\boldsymbol{T} \begin{array}{c|cc} & d & 1 \\ \hline d & \boldsymbol{J} & \boldsymbol{0} \\ 1 & \boldsymbol{0} & -1 \end{array}. \end{aligned}$$

We denote $\tilde{\boldsymbol{Z}} := \left(\begin{smallmatrix} \boldsymbol{J} & \boldsymbol{0} \\ \boldsymbol{0} & -1 \end{smallmatrix}\right)$, hence $\boldsymbol{Z} = \tilde{\boldsymbol{B}}^{-\top}\boldsymbol{T}\tilde{\boldsymbol{Z}}$. We also have $\tilde{\boldsymbol{Z}}^{-1} = \left(\begin{smallmatrix} \boldsymbol{J}^{-1} & \boldsymbol{0} \\ \boldsymbol{0} & -1 \end{smallmatrix}\right)$. $\mathcal{B}$ then can compute

$$g_1^{\boldsymbol{B}\binom{\boldsymbol{I}_d}{0}} = g_1^{\tilde{\boldsymbol{B}}\binom{\boldsymbol{I}_d}{0}}, \qquad g_2^{\boldsymbol{Z}} = g_2^{\tilde{\boldsymbol{B}}^{-\top}\boldsymbol{T}\tilde{\boldsymbol{Z}}},$$

where in the first term, the unknown last column of $\boldsymbol{B}$ vanishes through the left projection map $\binom{\boldsymbol{I}_d}{0}$, while the second term is computable from $g_2^{\boldsymbol{T}}$. From these two terms, $\mathcal{B}$ can compute $\mathsf{PK}, \mathsf{MSK}, \widehat{\mathsf{MSK}}_{\mathsf{base}}, \widehat{\mathsf{MSK}}_{\mathsf{aux}}$; in particular, $\mathcal{B}$ chooses $\boldsymbol{\alpha} \xleftarrow{\$} \mathbb{Z}_p^{(d+1)\times 1}$, $\boldsymbol{H}_1, \ldots, \boldsymbol{H}_n \xleftarrow{\$} \mathbb{Z}_p^{(d+1)\times(d+1)}$ and computes Eq.(7),(13).

Phase 1. When $\mathcal{A}$ makes the j-th key query for X_j, $\mathcal{B}$ generates a key as follows

- **Case** $j > i$**.** $\mathcal{B}$ generates a normal key $\mathsf{SK} \leftarrow \mathsf{KeyGen}(X_j, \mathsf{MSK})$.
- **Case** $j < i$**.** $\mathcal{B}$ chooses $\beta_j \xleftarrow{\$} \mathbb{Z}_p$ and generates a type-3 semi-functional key as $\mathsf{SK} \leftarrow \mathsf{SFKeyGen}(X_j, \mathsf{MSK}, \widehat{\mathsf{MSK}}_{\mathsf{base}}, -, 3, \beta_j)$.
- **Case** $j = i$**.** Let $(\boldsymbol{k}; m_2) \leftarrow \mathsf{Enc1}(X_j)$. $\mathcal{B}$ extends the Matrix-DH Assumption to m_2-fold by random self reducibility and obtains $(g_2^{\boldsymbol{T}}, g_2^{\boldsymbol{T}\binom{\boldsymbol{Y}}{\hat{\boldsymbol{y}}}})$ where either $\hat{\boldsymbol{y}} = \boldsymbol{0}$ or $\hat{\boldsymbol{y}} \xleftarrow{\$} \mathbb{Z}_p^{1\times m_2}$ with $\boldsymbol{T} \xleftarrow{\$} \mathcal{D}_d$, $\boldsymbol{Y} \xleftarrow{\$} \mathbb{Z}_p^{d\times m_2}$. $\mathcal{B}$ implicitly sets $\boldsymbol{R} + \hat{\boldsymbol{R}} = \tilde{\boldsymbol{Z}}^{-1}\binom{\boldsymbol{Y}}{\hat{\boldsymbol{y}}}$; that is,

$$\boldsymbol{R} + \hat{\boldsymbol{R}} = \begin{array}{c|ccc} & 1 & & 1 \\ \hline d & \boldsymbol{r}_1 & \cdots & \boldsymbol{r}_{m_2} \\ 1 & \hat{r}_1 & \cdots & \hat{r}_{m_2} \end{array} = \tilde{\boldsymbol{Z}}^{-1} \begin{array}{c|c} & m_2 \\ \hline d & \boldsymbol{Y} \\ 1 & \hat{\boldsymbol{y}} \end{array} = \tilde{\boldsymbol{Z}}^{-1} \begin{array}{c|ccc} & 1 & & 1 \\ \hline d & \boldsymbol{y}_1 & \cdots & \boldsymbol{y}_{m_2} \\ 1 & \hat{y}_1 & \cdots & \hat{y}_{m_2} \end{array},$$

where we denote $\begin{pmatrix} \boldsymbol{y}_j \\ \hat{y}_j \end{pmatrix}$ as the j-th column of $\begin{pmatrix} \boldsymbol{Y} \\ \hat{\boldsymbol{y}} \end{pmatrix}$. Thus, $\begin{pmatrix} \boldsymbol{r}_j \\ \hat{r}_j \end{pmatrix} = \tilde{\boldsymbol{Z}}^{-1} \begin{pmatrix} \boldsymbol{y}_j \\ \hat{y}_j \end{pmatrix}$. $\mathcal{B}$ then can compute for each $j \in [1, m_2]$,

$$g_2^{\boldsymbol{Z}\begin{pmatrix} \boldsymbol{r}_j \\ \hat{r}_j \end{pmatrix}} = g_2^{\left(\tilde{\boldsymbol{B}}^{-\top}\boldsymbol{T}\tilde{\boldsymbol{Z}}\right)\left(\tilde{\boldsymbol{Z}}^{-1}\begin{pmatrix} \boldsymbol{y}_j \\ \hat{y}_j \end{pmatrix}\right)} = g_2^{\tilde{\boldsymbol{B}}^{-\top}\boldsymbol{T}\begin{pmatrix} \boldsymbol{y}_j \\ \hat{y}_j \end{pmatrix}},$$

since $\mathcal{B}$ possesses $g_2^{\boldsymbol{T}\begin{pmatrix} \boldsymbol{Y} \\ \hat{\boldsymbol{y}} \end{pmatrix}}$. From these terms, $\mathcal{B}$ can compute all the elements of the key since $\mathcal{B}$ possesses $\boldsymbol{\alpha}, \boldsymbol{H}_1, \ldots, \boldsymbol{H}_n$. We can see that if $\hat{\boldsymbol{y}} = \boldsymbol{0}$ then the key is a normal key, and if $\hat{\boldsymbol{y}} \xleftarrow{\$} \mathbb{Z}_p^{1\times m_2}$ then the key is type-1 semi-functional.

Challenge. The adversary $\mathcal{A}$ outputs messages $M_0, M_1 \in \mathbb{G}_T$ along with a target $Y^\star$. $\mathcal{B}$ chooses $b \xleftarrow{\$} \{0,1\}$. Let $(\boldsymbol{c}; w_2) \leftarrow \mathsf{Enc2}(Y)$. For $j \in [0, w_2]$, $\mathcal{B}$ chooses $\begin{pmatrix} \boldsymbol{s}'_j \\ \hat{s}'_j \end{pmatrix} \xleftarrow{\$} \mathbb{Z}_p^{(d+1)\times 1}$ and implicitly sets

$$\begin{pmatrix} \boldsymbol{s}_j \\ \hat{s}_j \end{pmatrix} = \boldsymbol{B}^{-1} \begin{pmatrix} \boldsymbol{s}'_j \\ \hat{s}'_j \end{pmatrix}.$$

$\mathcal{B}$ then can compute

$$g_1^{\boldsymbol{B}\begin{pmatrix} \boldsymbol{s}_j \\ \hat{s}_j \end{pmatrix}} = g_1^{\boldsymbol{B}\left(\boldsymbol{B}^{-1}\begin{pmatrix} \boldsymbol{s}'_j \\ \hat{s}'_j \end{pmatrix}\right)} = g_1^{\begin{pmatrix} \boldsymbol{s}'_j \\ \hat{s}'_j \end{pmatrix}}.$$

From these terms, $\mathcal{B}$ can compute all the elements of the semi-functional ciphertext for M_b (since, again, $\mathcal{B}$ possesses $\boldsymbol{\alpha}, \boldsymbol{H}_1, \ldots, \boldsymbol{H}_n$). In particular, for C_0, $\mathcal{B}$ computes

$$C_0 = e(g_1^{\boldsymbol{\alpha}^\top \boldsymbol{B}\begin{pmatrix} \boldsymbol{s}_0 \\ \hat{s}_0 \end{pmatrix}}, g_2) = e(g_1^{\boldsymbol{\alpha}^\top\begin{pmatrix} \boldsymbol{s}'_j \\ \hat{s}'_j \end{pmatrix}}, g_2).$$

Phase 2. When $\mathcal{A}$ makes the j-th key query for X_j, since $j > i$ in this phase, $\mathcal{B}$ generates a normal key $\mathsf{SK} \leftarrow \mathsf{KeyGen}(X_j, \mathsf{MSK})$.

Guess. The algorithm $\mathcal{B}$ has properly simulated $\mathsf{G}_{i-1,3}$ if $\hat{y} = 0$ and $\mathsf{G}_{i,1}$ if $\hat{y} \xleftarrow{\$} \mathbb{Z}_p$. Hence, $\mathcal{B}$ can use the output of $\mathcal{A}$ to break the Matrix DH Assumption. □

5.3 Applying the Parameter-Hiding Lemma

Equivalent Semi-Functional Algorithms. Before describing the proof for the next transition (switching type-1 to type-2 semi-functional keys), we prepare another setup algorithm, $\mathsf{SFSetup}'$, and its consequences to the other algorithms. The difference from $\mathsf{SFSetup}$ is shown in red color. We will then prove that $\mathsf{SFSetup}'$ is indeed equivalent to $\mathsf{SFSetup}$.

- $\mathsf{SFSetup}'(1^\lambda, \kappa) \to (\mathsf{PK}, \mathsf{MSK}, \widehat{\mathsf{PK}}', \widehat{\mathsf{MSK}}_{\mathsf{base}}, \widehat{\mathsf{MSK}}'_{\mathsf{aux}})$: This is modifed from $\mathsf{SFSetup}$, where it outputs $\mathsf{PK}, \mathsf{MSK}, \widehat{\mathsf{MSK}}_{\mathsf{base}}$ in exactly the same way. It additionally chooses $\hat{h}_1, \ldots, \hat{h}_n \xleftarrow{\$} \mathbb{Z}_p$ and outputs $\widehat{\mathsf{PK}}'$, $\widehat{\mathsf{MSK}}'_{\mathsf{aux}}$ defined as

$$\widehat{\mathsf{PK}}' = \left(e(g_1, g_2)^{\boldsymbol{\alpha}^\top \boldsymbol{B}\begin{pmatrix} \boldsymbol{0} \\ 1 \end{pmatrix}}, g_1^{\boldsymbol{B}\begin{pmatrix} \boldsymbol{0} \\ 1 \end{pmatrix}}, g_1^{\boldsymbol{H}_1\boldsymbol{B}\begin{pmatrix} \boldsymbol{0} \\ 1 \end{pmatrix} + \boldsymbol{B}\begin{pmatrix} \boldsymbol{0} \\ \hat{h}_1 \end{pmatrix}}, \ldots, g_1^{\boldsymbol{H}_n\boldsymbol{B}\begin{pmatrix} \boldsymbol{0} \\ 1 \end{pmatrix} + \boldsymbol{B}\begin{pmatrix} \boldsymbol{0} \\ \hat{h}_n \end{pmatrix}} \right), \tag{19}$$

$$\widehat{\mathsf{MSK}}'_{\mathsf{aux}} = \left(g_2^{\boldsymbol{H}_1^\top \boldsymbol{Z}\begin{pmatrix} \boldsymbol{0} \\ 1 \end{pmatrix} + \boldsymbol{Z}\begin{pmatrix} \boldsymbol{0} \\ \hat{h}_1 \end{pmatrix}}, \ldots, g_2^{\boldsymbol{H}_n^\top \boldsymbol{Z}\begin{pmatrix} \boldsymbol{0} \\ 1 \end{pmatrix} + \boldsymbol{Z}\begin{pmatrix} \boldsymbol{0} \\ \hat{h}_n \end{pmatrix}} \right). \tag{20}$$

- $\mathsf{SFEncrypt}(Y, M, \mathsf{PK}, \widehat{\mathsf{PK}}') \to \mathsf{CT}'$: We write the output from the already defined $\mathsf{SFEncrypt}$ algorithm, but now with the input $\widehat{\mathsf{PK}}'$ instead of $\widehat{\mathsf{PK}}$. Concretely, it picks $\boldsymbol{s}_0, \boldsymbol{s}_1, \ldots, \boldsymbol{s}_{w_2} \xleftarrow{\$} \mathbb{Z}_p^{d\times 1}$ and $\hat{s}_0, \hat{s}_1, \ldots, \hat{s}_{w_2} \xleftarrow{\$} \mathbb{Z}_p$. Let

$$\boldsymbol{S} := \left(\begin{pmatrix}\boldsymbol{s}_0\\0\end{pmatrix}, \begin{pmatrix}\boldsymbol{s}_1\\0\end{pmatrix}, \ldots, \begin{pmatrix}\boldsymbol{s}_{w_2}\\0\end{pmatrix}\right) \in (\mathbb{Z}_p^{(d+1)\times 1})^{w_2+1},$$
$$\hat{\boldsymbol{S}} := \left(\begin{pmatrix}\boldsymbol{0}\\\hat{s}_0\end{pmatrix}, \begin{pmatrix}\boldsymbol{0}\\\hat{s}_1\end{pmatrix}, \ldots, \begin{pmatrix}\boldsymbol{0}\\\hat{s}_{w_2}\end{pmatrix}\right) \in (\mathbb{Z}_p^{(d+1)\times 1})^{w_2+1}$$

Denote $\hat{\boldsymbol{h}} = (\hat{h}_1, \ldots, \hat{h}_n)$. We claim that the outputted ciphertext is $\mathsf{CT} = (\boldsymbol{C}, C_0)$ where

$$\boldsymbol{C} = g_1^{\boldsymbol{c_B}(\boldsymbol{S},\mathbb{H})+\boldsymbol{c}'_{\boldsymbol{B}}(\hat{\boldsymbol{S}},\mathbb{H},\hat{\boldsymbol{h}})} \in (\mathbb{G}_1^{(d+1)\times 1})^{w_1}, \qquad C_0 = e(g_1, g_2)^{\boldsymbol{\alpha}^\top \boldsymbol{B}\begin{pmatrix}\boldsymbol{s}_0\\\hat{s}_0\end{pmatrix}} \cdot M \in \mathbb{G}_T. \tag{21}$$

where $\boldsymbol{c}'_{\boldsymbol{B}}(\hat{\boldsymbol{S}}, \mathbb{H}, \hat{\boldsymbol{h}})$ is newly defined via

$$\boldsymbol{c_B}(\boldsymbol{S}, \mathbb{H}) + \boldsymbol{c}'_{\boldsymbol{B}}(\hat{\boldsymbol{S}}, \mathbb{H}, \hat{\boldsymbol{h}}) = \left\{\left(\sum_{j\in[0,w_2]} a_{i,j}\boldsymbol{B}\begin{pmatrix}\boldsymbol{s}_j\\\hat{s}_j\end{pmatrix}\right) + \left(\sum_{\substack{j\in[0,w_2]\\k\in[1,n]}} a_{i,j,k}\left(\boldsymbol{H}_k\boldsymbol{B}\begin{pmatrix}\boldsymbol{s}_j\\\hat{s}_j\end{pmatrix} + \boldsymbol{B}\begin{pmatrix}\boldsymbol{0}\\\hat{h}_k\hat{s}_j\end{pmatrix}\right)\right)\right\}_{i\in[1,w_1]}. \tag{22}$$

The claim can be verified thanks to the relation $\left(\boldsymbol{X}\begin{pmatrix}\boldsymbol{I}_d\\0\end{pmatrix}\right)\boldsymbol{y} + \left(\boldsymbol{X}\begin{pmatrix}\boldsymbol{0}\\1\end{pmatrix} + \boldsymbol{B}\begin{pmatrix}\boldsymbol{0}\\h\end{pmatrix}\right)\hat{y} = \boldsymbol{X}\begin{pmatrix}\boldsymbol{y}\\\hat{y}\end{pmatrix} + \boldsymbol{B}\begin{pmatrix}\boldsymbol{0}\\h\hat{y}\end{pmatrix}$ for any $\boldsymbol{X}, \boldsymbol{B} \in \mathbb{Z}_p^{(d+1)\times(d+1)}$, $\boldsymbol{y} \in \mathbb{Z}_p^{d\times 1}$, and $h, \hat{y} \in \mathbb{Z}_p$.

- $\mathsf{SFKeyGen}(X, \mathsf{MSK}, \widehat{\mathsf{MSK}}_{\mathsf{base}}, \widehat{\mathsf{MSK}}'_{\mathsf{aux}}, \mathsf{type} \in \{1,2\}, \beta \in \mathbb{Z}_p) \to \mathsf{SK}$: We write the output from the already defined $\mathsf{SFKeyGen}$ algorithm, but now with the input $\widehat{\mathsf{MSK}}'_{\mathsf{aux}}$ instead of $\widehat{\mathsf{MSK}}_{\mathsf{aux}}$ and we consider only $\mathsf{type} = 1, 2$. Concretely, it picks $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_{m_2} \xleftarrow{\$} \mathbb{Z}_p^{d\times 1}$ and $\hat{r}_1, \ldots, \hat{r}_{m_2} \xleftarrow{\$} \mathbb{Z}_p$. Let

$$\boldsymbol{R} := \left(\begin{pmatrix}\boldsymbol{r}_1\\0\end{pmatrix}, \ldots, \begin{pmatrix}\boldsymbol{r}_{m_2}\\0\end{pmatrix}\right) \in (\mathbb{Z}_p^{(d+1)\times 1})^{m_2},$$
$$\hat{\boldsymbol{R}} := \left(\begin{pmatrix}\boldsymbol{0}\\\hat{r}_1\end{pmatrix}, \ldots, \begin{pmatrix}\boldsymbol{0}\\\hat{r}_{m_2}\end{pmatrix}\right) \in (\mathbb{Z}_p^{(d+1)\times 1})^{m_2}$$

We claim that the outputted secret key SK is

$$\mathsf{SK} = \begin{cases} g_2^{\boldsymbol{k_Z}(\boldsymbol{\alpha},\boldsymbol{R},\mathbb{H})+\boldsymbol{k}'_{\boldsymbol{Z}}(0,\hat{\boldsymbol{R}},\mathbb{H},\hat{\boldsymbol{h}})} & \text{if } \mathsf{type} = 1 \quad (23)\\ g_2^{\boldsymbol{k_Z}(\boldsymbol{\alpha},\boldsymbol{R},\mathbb{H})+\boldsymbol{k}'_{\boldsymbol{Z}}(\beta,\hat{\boldsymbol{R}},\mathbb{H},\hat{\boldsymbol{h}})} & \text{if } \mathsf{type} = 2 \quad (24)\end{cases}$$

where $\boldsymbol{k}'_{\boldsymbol{Z}}(\beta, \hat{\boldsymbol{R}}, \mathbb{H}, \hat{\boldsymbol{h}})$ is newly defined via

$$\boldsymbol{k}_{\boldsymbol{Z}}(\boldsymbol{\alpha}, \boldsymbol{R}, \mathbb{H}) + \boldsymbol{k}'_{\boldsymbol{Z}}(\beta, \hat{\boldsymbol{R}}, \mathbb{H}, \hat{\boldsymbol{h}}) = \left\{ b_i\boldsymbol{\alpha} + b_i\boldsymbol{Z}\left(\begin{smallmatrix}\boldsymbol{0}\\ \beta\end{smallmatrix}\right) + \left(\sum_{j\in[1,m_2]} b_{i,j}\boldsymbol{Z}\left(\begin{smallmatrix}\boldsymbol{r}_j\\ \hat{r}_j\end{smallmatrix}\right)\right) + \left(\sum_{\substack{j\in[1,m_2]\\ k\in[1,n]}} b_{i,j,k}\left(\boldsymbol{H}_k^\top\boldsymbol{Z}\left(\begin{smallmatrix}\boldsymbol{r}_j\\ \hat{r}_j\end{smallmatrix}\right) + \boldsymbol{Z}\left(\begin{smallmatrix}\boldsymbol{0}\\ \hat{h}_k\hat{r}_j\end{smallmatrix}\right)\right)\right)\right\}_{i\in[1,m_1]} \tag{25}$$

The claim can be verified thanks to the relation $\left(\boldsymbol{X}\left(\begin{smallmatrix}\boldsymbol{I}_d\\ 0\end{smallmatrix}\right)\right)\boldsymbol{y} + \left(\boldsymbol{X}\left(\begin{smallmatrix}\boldsymbol{0}\\ 1\end{smallmatrix}\right) + \boldsymbol{Z}\left(\begin{smallmatrix}\boldsymbol{0}\\ h\end{smallmatrix}\right)\right)\hat{y} = \boldsymbol{X}\left(\begin{smallmatrix}\boldsymbol{y}\\ \hat{y}\end{smallmatrix}\right) + \boldsymbol{Z}\left(\begin{smallmatrix}\boldsymbol{0}\\ h\hat{y}\end{smallmatrix}\right)$ for any $\boldsymbol{X}, \boldsymbol{Z} \in \mathbb{Z}_p^{(d+1)\times(d+1)}$, $\boldsymbol{y} \in \mathbb{Z}_p^{d\times 1}$, and $h, \hat{y} \in \mathbb{Z}_p$.

For further use, we write explicit forms of $\boldsymbol{c}'_{\boldsymbol{B}}(\hat{\boldsymbol{S}}, \mathbb{H}, \hat{\boldsymbol{h}})$ and $\boldsymbol{k}'_{\boldsymbol{Z}}(\beta, \hat{\boldsymbol{R}}, \mathbb{H}, \hat{\boldsymbol{h}})$ as follows:

$$g_1^{\boldsymbol{c}'_{\boldsymbol{B}}(\hat{\boldsymbol{S}}, \mathbb{H}, \hat{\boldsymbol{h}})} = \left\{ g_1^{\boldsymbol{B}\left(\begin{smallmatrix}\boldsymbol{0}\\ c_\iota(\hat{\boldsymbol{s}}, \hat{\boldsymbol{h}})\end{smallmatrix}\right)} \prod_{\substack{j\in[0,w_2]\\ k\in[1,n]}} g_1^{a_{\iota,j,k}\boldsymbol{H}_k\boldsymbol{B}\left(\begin{smallmatrix}\boldsymbol{0}\\ \hat{s}_j\end{smallmatrix}\right)} \right\}_{\iota\in[1,w_1]}.$$

$$g_2^{\boldsymbol{k}'_{\boldsymbol{Z}}(\beta, \hat{\boldsymbol{R}}, \mathbb{H}, \hat{\boldsymbol{h}})} = \left\{ g_2^{\boldsymbol{Z}\left(\begin{smallmatrix}\boldsymbol{0}\\ k_\iota(\beta, \hat{\boldsymbol{r}}, \hat{\boldsymbol{h}})\end{smallmatrix}\right)} \prod_{\substack{j\in[1,m_2]\\ k\in[1,n]}} g_2^{b_{\iota,j,k}\boldsymbol{H}_k^\top\boldsymbol{Z}\left(\begin{smallmatrix}\boldsymbol{0}\\ \hat{r}_j\end{smallmatrix}\right)} \right\}_{\iota\in[1,m_1]}.$$

The first equation holds from (22) by eliminating (9) and using the definition of $\boldsymbol{c}$ in (4) to inspect. The second equation can be done similarly.

Lemma 6. *The outputs from* $\mathsf{SFSetup}'$ *and* $\mathsf{SFSetup}$ *are identically distributed.*

Proof. From the parameter-hiding lemma (Lemma 2), we have that for $\boldsymbol{H}_i \xleftarrow{\$} \mathbb{Z}_p^{(d+1)\times(d+1)}$, $\boldsymbol{B} \xleftarrow{\$} \mathbb{GL}_{p,d+1}$, and $\hat{h}_i \xleftarrow{\$} \mathbb{Z}_p$, the distribution of

$$F_{\boldsymbol{H}_i,\hat{h}_i} := \boldsymbol{H}_i + \boldsymbol{B} \begin{array}{c|cc} & d & 1 \\ \hline d & \boldsymbol{0} & \boldsymbol{0} \\ 1 & \boldsymbol{0} & \hat{h}_i \end{array} \boldsymbol{B}^{-1} \tag{26}$$

is exactly the same as $\boldsymbol{H}_i$. Therefore, replacing $\boldsymbol{H}_i$ with $F_{\boldsymbol{H}_i,\hat{h}_i}$ for every $i \in [1, n]$, does not affect the output distribution of $\mathsf{SFKeyGen}$. We claim that this modification results in the unmodified elements $\mathsf{PK}, \mathsf{MSK}, \widehat{\mathsf{MSK}}_{\mathsf{base}}$ as shown in Eq. (7), Eq. (13), and the modified elements $\widehat{\mathsf{PK}}', \widehat{\mathsf{MSK}}'_{\mathsf{aux}}$ as shown in Eq. (19), Eq. (20). To prove the claim, it is sufficient to prove that:

$$\begin{aligned}
F_{\boldsymbol{H}_i,\hat{h}_i}\boldsymbol{B}\begin{pmatrix}\boldsymbol{I}_d\\ \boldsymbol{0}\end{pmatrix} &= \boldsymbol{H}_i\boldsymbol{B}\begin{pmatrix}\boldsymbol{I}_d\\ \boldsymbol{0}\end{pmatrix},\\
F^{\top}_{\boldsymbol{H}_i,\hat{h}_i}\boldsymbol{Z}\begin{pmatrix}\boldsymbol{I}_d\\ \boldsymbol{0}\end{pmatrix} &= \boldsymbol{H}_i^{\top}\boldsymbol{Z}\begin{pmatrix}\boldsymbol{I}_d\\ \boldsymbol{0}\end{pmatrix},\\
F_{\boldsymbol{H}_i,\hat{h}_i}\boldsymbol{B}\begin{pmatrix}\boldsymbol{0}\\ 1\end{pmatrix} &= \boldsymbol{H}_i\boldsymbol{B}\begin{pmatrix}\boldsymbol{0}\\ 1\end{pmatrix} + \boldsymbol{B}\begin{pmatrix}\boldsymbol{0}\\ \hat{h}_i\end{pmatrix},\\
F^{\top}_{\boldsymbol{H}_i,\hat{h}_i}\boldsymbol{Z}\begin{pmatrix}\boldsymbol{0}\\ 1\end{pmatrix} &= \boldsymbol{H}_i^{\top}\boldsymbol{Z}\begin{pmatrix}\boldsymbol{0}\\ 1\end{pmatrix} + \boldsymbol{B}^{-\top}\begin{pmatrix}\boldsymbol{0}\\ \hat{h}_i\end{pmatrix}.
\end{aligned}$$

This can be verified as follows, where we recall $\boldsymbol{D} = \left(\begin{smallmatrix}\tilde{\boldsymbol{D}} & \boldsymbol{0}\\ \boldsymbol{0} & 1\end{smallmatrix}\right)$,

$$\begin{aligned}
F_{\boldsymbol{H}_i,\hat{h}_i}\boldsymbol{B}\left(\begin{smallmatrix}\boldsymbol{I}_d\\ 0\end{smallmatrix}\right) &= \left(\boldsymbol{H}_i + \boldsymbol{B}\left(\begin{smallmatrix}\boldsymbol{0} & \boldsymbol{0}\\ \boldsymbol{0} & \hat{h}_i\end{smallmatrix}\right)\boldsymbol{B}^{-1}\right)\boldsymbol{B}\left(\begin{smallmatrix}\boldsymbol{I}_d\\ 0\end{smallmatrix}\right) = \boldsymbol{H}_i\boldsymbol{B}\left(\begin{smallmatrix}\boldsymbol{I}_d\\ 0\end{smallmatrix}\right) + \boldsymbol{B}\left(\begin{smallmatrix}\boldsymbol{0} & \boldsymbol{0}\\ \boldsymbol{0} & \hat{h}_i\end{smallmatrix}\right)\left(\begin{smallmatrix}\boldsymbol{I}_d\\ 0\end{smallmatrix}\right)\\
&= \boldsymbol{H}_i\boldsymbol{B}\left(\begin{smallmatrix}\boldsymbol{I}_d\\ 0\end{smallmatrix}\right)\\
F^{\top}_{\boldsymbol{H}_i,\hat{h}_i}\boldsymbol{Z}\left(\begin{smallmatrix}\boldsymbol{I}_d\\ 0\end{smallmatrix}\right) &= \left(\boldsymbol{H}_i^{\top} + \boldsymbol{B}^{-\top}\left(\begin{smallmatrix}\boldsymbol{0} & \boldsymbol{0}\\ \boldsymbol{0} & \hat{h}_i\end{smallmatrix}\right)\boldsymbol{B}^{\top}\right)(\boldsymbol{B}^{-\top}\boldsymbol{D})\left(\begin{smallmatrix}\boldsymbol{I}_d\\ 0\end{smallmatrix}\right)\\
&= \boldsymbol{H}_i^{\top}\boldsymbol{Z}\left(\begin{smallmatrix}\boldsymbol{I}_d\\ 0\end{smallmatrix}\right) + \boldsymbol{B}^{-\top}\left(\begin{smallmatrix}\boldsymbol{0} & \boldsymbol{0}\\ \boldsymbol{0} & \hat{h}_i\end{smallmatrix}\right)\boldsymbol{D}\left(\begin{smallmatrix}\boldsymbol{I}_d\\ 0\end{smallmatrix}\right)\\
&= \boldsymbol{H}_i^{\top}\boldsymbol{Z}\left(\begin{smallmatrix}\boldsymbol{I}_d\\ 0\end{smallmatrix}\right) + \boldsymbol{B}^{-\top}\left(\begin{smallmatrix}\boldsymbol{0} & \boldsymbol{0}\\ \boldsymbol{0} & \hat{h}_i\end{smallmatrix}\right)\left(\begin{smallmatrix}\tilde{\boldsymbol{D}} & \boldsymbol{0}\\ \boldsymbol{0} & 1\end{smallmatrix}\right)\left(\begin{smallmatrix}\boldsymbol{I}_d\\ 0\end{smallmatrix}\right) = \boldsymbol{H}_i^{\top}\boldsymbol{Z}\left(\begin{smallmatrix}\boldsymbol{I}_d\\ 0\end{smallmatrix}\right),\\
F_{\boldsymbol{H}_i,\hat{h}_i}\boldsymbol{B}\left(\begin{smallmatrix}\boldsymbol{0}\\ 1\end{smallmatrix}\right) &= \left(\boldsymbol{H}_i + \boldsymbol{B}\left(\begin{smallmatrix}\boldsymbol{0} & \boldsymbol{0}\\ \boldsymbol{0} & \hat{h}_i\end{smallmatrix}\right)\boldsymbol{B}^{-1}\right)\boldsymbol{B}\left(\begin{smallmatrix}\boldsymbol{0}\\ 1\end{smallmatrix}\right) = \boldsymbol{H}_i\boldsymbol{B}\left(\begin{smallmatrix}\boldsymbol{0}\\ 1\end{smallmatrix}\right) + \boldsymbol{B}\left(\begin{smallmatrix}\boldsymbol{0} & \boldsymbol{0}\\ \boldsymbol{0} & \hat{h}_i\end{smallmatrix}\right)\left(\begin{smallmatrix}\boldsymbol{0}\\ 1\end{smallmatrix}\right)\\
&= \boldsymbol{H}_i\boldsymbol{B}\left(\begin{smallmatrix}\boldsymbol{0}\\ 1\end{smallmatrix}\right) + \boldsymbol{B}\left(\begin{smallmatrix}\boldsymbol{0}\\ \hat{h}_i\end{smallmatrix}\right),\\
F^{\top}_{\boldsymbol{H}_i,\hat{h}_i}\boldsymbol{Z}\left(\begin{smallmatrix}\boldsymbol{0}\\ 1\end{smallmatrix}\right) &= \left(\boldsymbol{H}_i^{\top} + \boldsymbol{B}^{-\top}\left(\begin{smallmatrix}\boldsymbol{0} & \boldsymbol{0}\\ \boldsymbol{0} & \hat{h}_i\end{smallmatrix}\right)\boldsymbol{B}^{\top}\right)(\boldsymbol{B}^{-\top}\boldsymbol{D})\left(\begin{smallmatrix}\boldsymbol{0}\\ 1\end{smallmatrix}\right)\\
&= \boldsymbol{H}_i^{\top}\boldsymbol{Z}\left(\begin{smallmatrix}\boldsymbol{0}\\ 1\end{smallmatrix}\right) + \boldsymbol{B}^{-\top}\left(\begin{smallmatrix}\boldsymbol{0} & \boldsymbol{0}\\ \boldsymbol{0} & \hat{h}_i\end{smallmatrix}\right)\boldsymbol{D}\left(\begin{smallmatrix}\boldsymbol{0}\\ 1\end{smallmatrix}\right)\\
&= \boldsymbol{H}_i^{\top}\boldsymbol{Z}\left(\begin{smallmatrix}\boldsymbol{0}\\ 1\end{smallmatrix}\right) + \boldsymbol{B}^{-\top}\left(\begin{smallmatrix}\boldsymbol{0} & \boldsymbol{0}\\ \boldsymbol{0} & \hat{h}_i\end{smallmatrix}\right)\left(\begin{smallmatrix}\tilde{\boldsymbol{D}} & \boldsymbol{0}\\ \boldsymbol{0} & 1\end{smallmatrix}\right)\left(\begin{smallmatrix}\boldsymbol{0}\\ 1\end{smallmatrix}\right) = \boldsymbol{H}_i^{\top}\boldsymbol{Z}\left(\begin{smallmatrix}\boldsymbol{0}\\ 1\end{smallmatrix}\right) + \boldsymbol{B}^{-\top}\left(\begin{smallmatrix}\boldsymbol{0}\\ \hat{h}_i\end{smallmatrix}\right).
\end{aligned}$$

This concludes the proof. □

5.4 Type-1 to Type-2 Semi-functional Key in Phase 1

Lemma 7 ($\mathsf{G}_{i,1}$ to $\mathsf{G}_{i,2}$)**.** *For any adversary $\mathcal{A}$ against* ABE*, there exists an algorithm $\mathcal{B}$ that breaks the co-selective master-key hiding security of encoding with* $|\mathsf{G}_{i,1}\mathsf{Adv}^{\mathsf{ABE}}_{\mathcal{A}}(\lambda) - \mathsf{G}_{i,2}\mathsf{Adv}^{\mathsf{ABE}}_{\mathcal{A}}(\lambda)| \le \mathsf{Adv}^{\mathsf{CMH}}_{\mathcal{B}}(\lambda)$.

Proof. Suppose we have an adversary $\mathcal{A}$ that can distinguish between $\mathsf{G}_{i,1}$ and $\mathsf{G}_{i,2}$ with non-negligible probability. We construct a simulator $\mathcal{B}$ that would win the co-selective game for master-key hiding for P by simulating $\mathsf{G}_{i,1}$ or $\mathsf{G}_{i,2}$ for $\mathcal{A}$. In the co-selective game for $\mathcal{B}$, $\mathcal{B}$ is given the group description $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$ from its challenger.

Setup. Algorithm $\mathcal{B}$ generates PK as in the real construction but using the given g_1, g_2. More precisely, $\mathcal{B}$ picks $\boldsymbol{H}_1, \ldots, \boldsymbol{H}_n \xleftarrow{\$} \mathbb{Z}_p^{(d+1)\times(d+1)}$, $\boldsymbol{B} \xleftarrow{\$} \mathbb{GL}_{p,d+1} \subset \mathbb{Z}_p^{(d+1)\times(d+1)}$. It chooses $\tilde{\boldsymbol{D}} \xleftarrow{\$} \mathbb{GL}_{p,d}$ and defines $\boldsymbol{D} := \left(\begin{smallmatrix}\tilde{\boldsymbol{D}} & \boldsymbol{0}\\ \boldsymbol{0} & 1\end{smallmatrix}\right) \in \mathbb{GL}_{p,d+1}$ and $\boldsymbol{Z} := \boldsymbol{B}^{-\top}\boldsymbol{D}$. It chooses $\boldsymbol{\alpha} \xleftarrow{\$} \mathbb{Z}_p^{(d+1)\times 1}$.

It computes PK, MSK as in Eq.(7) and $\widehat{\mathsf{MSK}}_{\mathsf{base}}$ as in Eq.(13). It sends PK to the adversary $\mathcal{A}$. We note that these elements distributed as if they are from $\mathsf{SFSetup}'$.

Phase 1. When $\mathcal{A}$ makes the j-th private key query for $X_{\mathsf{j}} \in \mathbb{X}$, $\mathcal{B}$ does as follows.

[**Case** $\mathsf{j} < i$] In this case, the algorithm $\mathcal{B}$ picks $\beta_{\mathsf{j}} \xleftarrow{\$} \mathbb{Z}_p$ and creates a type-3 semi-functional key by computing $\mathsf{SK} \leftarrow \mathsf{SFKeyGen}(X, \mathsf{MSK}, \widehat{\mathsf{MSK}}_{\mathsf{base}}, -, 3, \beta_{\mathsf{j}})$.

[**Case** $\mathsf{j} = i$] The algorithm $\mathcal{B}$ makes a key query X_i to its challenger and receives $\boldsymbol{V} = g_2^{\boldsymbol{k}(\beta, \hat{\boldsymbol{r}}, \hat{\boldsymbol{h}})}$ where $(\boldsymbol{k}; m_2) = \mathsf{Enc1}(X_i)$. This is the challenge for $\mathcal{B}$ to guess if $\beta = 0$ or β is randomly chosen in $\mathbb{Z}_p$. The crucial point here is that up to this point the semi-functional parameter $\hat{\boldsymbol{h}} = (\hat{h}_1, \ldots, \hat{h}_n)$, which will be used to define $\widehat{\mathsf{PK}}, \widehat{\mathsf{MSK}}_{\mathsf{aux}}$ for $\mathsf{SFSetup}'$, has not been defined since it is not required for all the previous keys. (This feature is known as *delayed parameter*). $\mathcal{B}$ will thus implicitly define the semi-functional parameter to $\hat{\boldsymbol{h}}$ that is defined in $\boldsymbol{V}$. (Note that $\hat{\boldsymbol{h}}$ is unknown to $\mathcal{B}$). This is done by answering back to $\mathcal{A}$ the key for X_i by using $\boldsymbol{V}$. More precisely, $\mathcal{B}$ does as follows.

1. $\mathcal{B}$ computes the normal part of the key as usual. More precisely, this is done by picking $\boldsymbol{\alpha} \xleftarrow{\$} \mathbb{Z}_p^{(d+1)\times 1}$, $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_{m_2} \xleftarrow{\$} \mathbb{Z}_p^{d\times 1}$ and setting $\boldsymbol{R} := \left(\left(\begin{smallmatrix} \boldsymbol{r}_1 \\ 0 \end{smallmatrix}\right), \ldots, \left(\begin{smallmatrix} \boldsymbol{r}_{m_2} \\ 0 \end{smallmatrix}\right)\right) \in (\mathbb{Z}_p^{(d+1)\times 1})^{m_2}$, and obtaining $g_2^{\boldsymbol{k}_{\boldsymbol{Z}}(\boldsymbol{\alpha}, \boldsymbol{R}, \mathbb{H})}$ by Eq.(11).

2. $\mathcal{B}$ computes the semi-functional part of the key by using $\boldsymbol{V}$ as follows. $\mathcal{B}$ implicitly sets $\hat{\boldsymbol{R}} = \left(\left(\begin{smallmatrix} \boldsymbol{0} \\ \hat{r}_1 \end{smallmatrix}\right), \ldots, \left(\begin{smallmatrix} \boldsymbol{0} \\ \hat{r}_{m_2} \end{smallmatrix}\right)\right) \in (\mathbb{Z}_p^{(d+1)\times 1})^{m_2}$ by using $\hat{\boldsymbol{r}} = (\hat{r}_1, \ldots, \hat{r}_{m_2})$ that is defined in $\boldsymbol{V}$. It also implicitly sets β_i as β that is defined in $\boldsymbol{V}$. These are done by computing
$$g_2^{\boldsymbol{k}'_{\boldsymbol{Z}}(\beta_i, \hat{\boldsymbol{R}}, \mathbb{H}, \hat{\boldsymbol{h}})} = \left\{ g_2^{\boldsymbol{Z}\left(\begin{smallmatrix} \boldsymbol{0} \\ k_\iota(\beta, \hat{\boldsymbol{r}}, \hat{\boldsymbol{h}}) \end{smallmatrix}\right)} \prod_{\substack{j\in[1,m_2] \\ k\in[1,n]}} g_2^{b_{\iota,j,k}\boldsymbol{H}_k^\top \boldsymbol{Z}\left(\begin{smallmatrix} \boldsymbol{0} \\ \hat{r}_j \end{smallmatrix}\right)} \right\}_{\iota\in[1,m_1]}.$$
We argue that this is computable as follows. The first term $g_2^{\boldsymbol{Z}\left(\begin{smallmatrix} \boldsymbol{0} \\ k_\iota(\beta, \hat{\boldsymbol{r}}, \hat{\boldsymbol{h}}) \end{smallmatrix}\right)}$ can be computed from $V_\iota = g_2^{k_\iota(\beta, \hat{\boldsymbol{r}}, \hat{\boldsymbol{h}})}$ and the known $\boldsymbol{Z}$, where we write $\boldsymbol{V} = (V_1, \ldots, V_{m_1})$. The second term (the product term) can be computed since for each $j \in [1, m_2]$ we either have the following.
   - For j such that there is ι' where $k_{\iota'}(\beta, \hat{\boldsymbol{r}}, \hat{\boldsymbol{h}}) = \hat{r}_j$, we have $g_2^{\hat{r}_j}$ available as $V_{\iota'} = g_2^{k_{\iota'}(\beta, \hat{\boldsymbol{r}}, \hat{\boldsymbol{h}})} = g_2^{\hat{r}_j}$ in $\boldsymbol{V}$.
   - For j such that there is no ι' as above, we have that by *regularity* of encoding (the second rule, in Definition 1), $b_{\iota'',j,k} = 0$ for all ι'', k.

   From these two facts and the known $\boldsymbol{Z}$ and $\boldsymbol{H}_k$, we can compute the product term.

3. $\mathcal{B}$ outputs $\mathsf{SK} = g_2^{\boldsymbol{k}_{\boldsymbol{Z}}(\boldsymbol{\alpha}, \boldsymbol{R}, \mathbb{H})} g_2^{\boldsymbol{k}'_{\boldsymbol{Z}}(\beta_i, \hat{\boldsymbol{R}}, \mathbb{H}, \hat{\boldsymbol{h}})}$.

This is a properly distributed semi-functional key of type-1 if $\beta = 0$ or type-2 if β is random.

[**Case** $\mathsf{j} > i$] The algorithm $\mathcal{B}$ generates a normal key as in the construction by computing $\mathsf{SK} \leftarrow \mathsf{KeyGen}(X_{\mathsf{j}}, \mathsf{MSK})$.

Challenge. In the challenge phase, the adversary $\mathcal{A}$ outputs messages $M_0, M_1 \in \mathbb{G}_T$ along with her target $Y^\star$ such that $R(X_j, Y^\star) = 0$ for all $j \in [1, q_1]$. $\mathcal{B}$ then makes a ciphertext query for $Y^\star$ to its challenger and receives back $\boldsymbol{U} = g_1^{\boldsymbol{c}(\hat{\boldsymbol{s}},\hat{\boldsymbol{h}})}$ where $(\boldsymbol{c}; w_2) \leftarrow \mathsf{Enc2}(Y^\star)$. This query can be made since $R(X_i, Y^\star) = 0$. Then, $\mathcal{B}$ flips a coin $b \xleftarrow{\$} \{0,1\}$, and does as follows.

1. $\mathcal{B}$ computes the normal part of the ciphertext as usual. More precisely, we pick $\boldsymbol{s}_0, \boldsymbol{s}_1, \ldots, \boldsymbol{s}_{w_2} \xleftarrow{\$} \mathbb{Z}_p^{d\times 1}$ and let $\boldsymbol{S} := \left(\left(\begin{smallmatrix} \boldsymbol{s}_0 \\ 0 \end{smallmatrix}\right), \left(\begin{smallmatrix} \boldsymbol{s}_1 \\ 0 \end{smallmatrix}\right), \ldots, \left(\begin{smallmatrix} \boldsymbol{s}_{w_2} \\ 0 \end{smallmatrix}\right) \right) \in (\mathbb{Z}_p^{(d+1)\times 1})^{w_2+1}$, and obtaining $\bar{C}_0 = e(g_1, g_2)^{\boldsymbol{\alpha}^\top \boldsymbol{B} \left(\begin{smallmatrix} \boldsymbol{s}_0 \\ 0 \end{smallmatrix}\right)} \cdot M_b$ and $g_1^{\boldsymbol{c}_{\boldsymbol{B}}(\boldsymbol{S},\mathbb{H})}$ by Eq.(9).

2. $\mathcal{B}$ computes the semi-functional part of the ciphertext by using $\boldsymbol{U}$ as follows. $\mathcal{B}$ implicitly sets $\hat{\boldsymbol{S}} = \left(\left(\begin{smallmatrix} \boldsymbol{0} \\ \hat{s}_0 \end{smallmatrix}\right), \left(\begin{smallmatrix} \boldsymbol{0} \\ \hat{s}_1 \end{smallmatrix}\right), \ldots, \left(\begin{smallmatrix} \boldsymbol{0} \\ \hat{s}_{w_2} \end{smallmatrix}\right) \right) \in (\mathbb{Z}_p^{(d+1)\times 1})^{w_2+1}$ by using $\hat{\boldsymbol{s}} = (\hat{s}_1, \ldots, \hat{s}_{w_2})$ that is defined in $\boldsymbol{U}$. This is done by computing

$$g_1^{\boldsymbol{c}'_{\boldsymbol{B}}(\hat{\boldsymbol{S}},\mathbb{H},\hat{\boldsymbol{h}})} = \left\{ g_1^{\boldsymbol{B}\left(\begin{smallmatrix} \boldsymbol{0} \\ c_\iota(\hat{\boldsymbol{s}},\hat{\boldsymbol{h}}) \end{smallmatrix}\right)} \prod_{\substack{j\in[0,w_2] \\ k\in[1,n]}} g_1^{a_{\iota,j,k}\boldsymbol{H}_k\boldsymbol{B}\left(\begin{smallmatrix} \boldsymbol{0} \\ \hat{s}_j \end{smallmatrix}\right)} \right\}_{\iota\in[1,w_1]}.$$

We argue that this is computable as follows. The first term $g_1^{\boldsymbol{B}\left(\begin{smallmatrix} \boldsymbol{0} \\ c_\iota(\hat{\boldsymbol{s}},\hat{\boldsymbol{h}}) \end{smallmatrix}\right)}$ can be computed from $D_i = g_1^{c_\iota(\hat{\boldsymbol{s}},\hat{\boldsymbol{h}})}$ and the known $\boldsymbol{B}$, where we write $\boldsymbol{U} = (U_1, \ldots, U_{w_1})$. The second term (the product term) can be computed since for each $j \in [1, w_2]$ we either have the following.

   – For j such that there is ι' where $c_{\iota'}(\hat{\boldsymbol{s}},\hat{\boldsymbol{h}}) = \hat{s}_j$, we have $g_1^{\hat{s}_j}$ available as $U_{\iota'} = g_1^{c_{\iota'}(\hat{\boldsymbol{s}},\hat{\boldsymbol{h}})} = g_1^{\hat{s}_j}$ in $\boldsymbol{U}$.

   – For j such that there is no ι' as above, we have that by *regularity* of encoding (the third rule, in Definition 1), $a_{\iota'',j,k} = 0$ for all ι'', k.

   From these two facts and the known $\boldsymbol{B}$ and $\boldsymbol{H}_k$, we can compute the product term.

3. $\mathcal{B}$ also computes $\hat{C}_0 = e(g_1^{\boldsymbol{\alpha}^\top\boldsymbol{B}\left(\begin{smallmatrix} \boldsymbol{0} \\ \hat{s}_0 \end{smallmatrix}\right)}, g_2)$. This can be done since $c_1 = g_1^{\hat{s}_0}$ is available in $\boldsymbol{U}$, also due to the *regularity* of encoding (the fourth rule, in Definition 1).

4. $\mathcal{B}$ computes $\boldsymbol{C} = g_1^{\boldsymbol{c}_{\boldsymbol{B}}(\boldsymbol{S},\mathbb{H})} g_1^{\boldsymbol{c}'_{\boldsymbol{B}}(\hat{\boldsymbol{S}},\mathbb{H},\hat{\boldsymbol{h}})}$ and $C_0 = \bar{C}_0\hat{C}_0$, and returns $\mathsf{CT} = (\boldsymbol{C}, C_0)$.

This is a properly distributed semi-functional ciphertext of Equation (14).

Phase 2. The algorithm $\mathcal{B}$ generates a normal key as in the construction by computing $\mathsf{SK} \leftarrow \mathsf{KeyGen}(X_j, \mathsf{MSK})$.

Guess. The algorithm $\mathcal{B}$ has properly simulated $\mathsf{G}_{i,1}$ if $\beta = 0$, and $\mathsf{G}_{i,2}$ if β is random. Hence, $\mathcal{B}$ can use the output of $\mathcal{A}$ to guess β. $\square$

5.5 Type-2 to Type-3 Semi-functional Key in Phase 1

Lemma 8 ($\mathsf{G}_{i,2}$ to $\mathsf{G}_{i,3}$)**.** *For any adversary $\mathcal{A}$ against* ABE*, there exists an algorithm $\mathcal{B}$ that breaks the $\mathcal{D}_d$-Matrix-DH Assumption with* $|\mathsf{G}_{i,2}\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ABE}}(\lambda) - \mathsf{G}_{i,3}\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ABE}}(\lambda)| \leq \mathsf{Adv}_{\mathcal{B}}^{\mathcal{D}_d\text{-}\mathsf{MatDH}}(\lambda)$.

Proof. The proof is exactly the same as that of Lemma 5, where we moved from normal to type-1 semi-functional key, except that the i-th key has an additional vector $g_2^{\boldsymbol{k}_{\boldsymbol{Z}}(\boldsymbol{Z}\binom{\boldsymbol{0}}{\beta_i}, \boldsymbol{0}, \mathbb{H})} = \{g_2^{b_\iota \boldsymbol{Z}\binom{\boldsymbol{0}}{\beta_i}}\}_{\iota\in[1,m_1]}$ multiplied with it (see Eq.(17),(18)). To compute this, $\mathcal{B}$ samples $\beta_i \in \mathbb{Z}_p$, and computes $g_2^{\boldsymbol{Z}\binom{\boldsymbol{0}}{\beta_i}} = g_2^{\boldsymbol{Z}\binom{\boldsymbol{0}}{1}\beta_i}$. We recall that $\mathcal{B}$ possesses $\widehat{\mathsf{MSK}}_{\mathsf{base}} = g_2^{\boldsymbol{Z}\binom{\boldsymbol{0}}{1}}$. Following the proof of Lemma 5, we can see that if $\hat{\boldsymbol{y}} = \boldsymbol{0}$ then the key is a type-3 semi-functional, and if $\hat{\boldsymbol{y}} \xleftarrow{\$} \mathbb{Z}_p^{1\times m_2}$ then the key is type-2 semi-functional. The rest of the proof follows exactly the same as that of Lemma 5. $\square$

5.6 Normal to Type-1 Semi-functional Keys in Phase 2

Lemma 9 ($\mathsf{G}_{q_1,3}$ to G_{q_1+1})**.** *For any adversary $\mathcal{A}$ against* ABE*, there exists an algorithm $\mathcal{B}$ that breaks the $\mathcal{D}_d$-Matrix-DH Assumption with* $|\mathsf{G}_{q_1,3}\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ABE}}(\lambda) - \mathsf{G}_{q_1+1}\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ABE}}(\lambda)| \leq \mathsf{Adv}_{\mathcal{B}}^{\mathcal{D}_d\text{-}\mathsf{MatDH}}(\lambda)$.

Proof. The proof is exactly the same as when we modify normal to type-1 semi-functional key in phase 1 (the proof of Lemma 5), except that this time we modify the post-challenge keys *all at once*, instead of one key at a time. In particular, the simulation of $\mathsf{PK}, \mathsf{MSK}, \widehat{\mathsf{MSK}}_{\mathsf{base}}, \widehat{\mathsf{MSK}}_{\mathsf{aux}}$ and the challenge ciphertext is exactly the same. The simulation for type-3 semi-functional key queries in phase 1 is done using $\widehat{\mathsf{MSK}}_{\mathsf{base}}$. The simulation of every key in phase 2 can be done by extending the Matrix-DH Assumption to q_2m_2-fold by random self reducibility (instead of only m_2 as in the proof of Lemma 5). It obtains $(g_2^{\boldsymbol{T}}, g_2^{\boldsymbol{T}\binom{\boldsymbol{Y}}{\hat{\boldsymbol{y}}}})$ where either $\hat{\boldsymbol{y}} = \boldsymbol{0}$ or $\hat{\boldsymbol{y}} \xleftarrow{\$} \mathbb{Z}_p^{1\times(q_2m_2)}$ with $\boldsymbol{T} \xleftarrow{\$} \mathcal{D}_d$, $\boldsymbol{Y} \xleftarrow{\$} \mathbb{Z}_p^{d\times(q_2m_2)}$. Each key will use m_2 columns of randomness in $\binom{\boldsymbol{Y}}{\hat{\boldsymbol{y}}}$, in an analogous way to the i-th key in the proof of Lemma 5. For example, the first key query in phase 2 will use the randomness in column 1 to m_2, the second key query in phase 2 will then use the randomness in column m_2+1 to $2m_2$, and so on. Following the proof of Lemma 5, we can see that if $\hat{\boldsymbol{y}} = \boldsymbol{0}$ then every key in phase 2 will be a normal key, and if $\hat{\boldsymbol{y}} \xleftarrow{\$} \mathbb{Z}_p^{1\times(q_2m_2)}$ then it is type-1 semi-functional. $\square$

5.7 Type-1 to Type-2 Semi-functional Key in Phase 2

Lemma 10 (G_{q_1+1} to G_{q_1+2})**.** *For any adversary $\mathcal{A}$ against* ABE*, there exists an algorithm $\mathcal{B}$ that breaks the selective master-key hiding security of encoding with* $|\mathsf{G}_{q_1+1}\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ABE}}(\lambda) - \mathsf{G}_{q_1+2}\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ABE}}(\lambda)| \leq \mathsf{Adv}_{\mathcal{B}}^{\mathsf{SMH}}(\lambda)$.

Proof. Suppose we have an adversary $\mathcal{A}$ that can distinguish between G_{q_1+1} and G_{q_1+2} with non-negligible probability. We construct a simulator $\mathcal{B}$ that would win the selective game for master-key hiding for P by simulating G_{q_1+1} or G_{q_1+2} for $\mathcal{A}$. In the selective game for $\mathcal{B}$, $\mathcal{B}$ is given the group description $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$ from its challenger.

Setup. Algorithm $\mathcal{B}$ generates PK as in the real construction but using the given g_1, g_2. More precisely, $\mathcal{B}$ picks $\boldsymbol{H}_1, \dots, \boldsymbol{H}_n \xleftarrow{\$} \mathbb{Z}_p^{(d+1)\times(d+1)}$, $\boldsymbol{B} \xleftarrow{\$} \mathbb{GL}_{p,d+1} \subset \mathbb{Z}_p^{(d+1)\times(d+1)}$. It chooses

$\tilde{\boldsymbol{D}} \xleftarrow{\$} \mathbb{GL}_{p,d}$ and defines $\boldsymbol{D} := \left(\begin{smallmatrix} \tilde{\boldsymbol{D}} & \boldsymbol{0} \\ \boldsymbol{0} & 1 \end{smallmatrix}\right) \in \mathbb{GL}_{p,d+1}$ and $\boldsymbol{Z} := \boldsymbol{B}^{-\top}\boldsymbol{D}$. It chooses $\boldsymbol{\alpha} \xleftarrow{\$} \mathbb{Z}_p^{(d+1)\times 1}$. It computes PK, MSK as in Eq.(7) and $\widehat{\mathsf{MSK}}_{\mathsf{base}}$ as in Eq.(13). It sends PK to the adversary $\mathcal{A}$. We note that these elements distributed as if they are from $\mathsf{SFSetup}'$.

Phase 1. When $\mathcal{A}$ makes the j-th private key query for $X_j \in \mathbb{X}$, $\mathcal{B}$ picks $\beta_j \xleftarrow{\$} \mathbb{Z}_p$ and runs $\mathsf{SK} \leftarrow \mathsf{SFKeyGen}(X, \mathsf{MSK}, \widehat{\mathsf{MSK}}_{\mathsf{base}}, -, 3, \beta_j)$ for a type-3 semi-functional key.

Challenge. In the challenge phase, the adversary $\mathcal{A}$ outputs messages $M_0, M_1 \in \mathbb{G}_T$ along with her target $Y^\star$ such that $R(X_j, Y^\star) = 0$ for all $j \in [1, q_1]$. $\mathcal{B}$ then makes a ciphertext query for $Y^\star$ to its challenger and receives back $\boldsymbol{U} = g_1^{\boldsymbol{c}(\hat{\boldsymbol{s}}, \hat{\boldsymbol{h}})}$ where $(\boldsymbol{c}; w_2) \leftarrow \mathsf{Enc2}(Y^\star)$. The crucial point here is that up to this point the semi-functional parameter have not been defined since it is not necessary for all the previous keys. $\mathcal{B}$ will thus implicitly define the semi-functional parameter to $\hat{\boldsymbol{h}}$ that is defined in $\boldsymbol{U}$. (Note that $\hat{\boldsymbol{h}}$ is unknown to $\mathcal{B}$). This is done by answering back to $\mathcal{A}$ the ciphertext for $Y^\star$ by using $\boldsymbol{U}$. More precisely, $\mathcal{B}$ first flips a coin $b \xleftarrow{\$} \{0, 1\}$, and does as follows.

1. $\mathcal{B}$ computes the normal part of the ciphertext as usual. More precisely, we randomly pick $\boldsymbol{s}_0, \boldsymbol{s}_1, \ldots, \boldsymbol{s}_{w_2} \xleftarrow{\$} \mathbb{Z}_p^{d\times 1}$, let $\boldsymbol{S} := \left(\left(\begin{smallmatrix} \boldsymbol{s}_0 \\ 0 \end{smallmatrix}\right), \left(\begin{smallmatrix} \boldsymbol{s}_1 \\ 0 \end{smallmatrix}\right), \ldots, \left(\begin{smallmatrix} \boldsymbol{s}_{w_2} \\ 0 \end{smallmatrix}\right)\right) \in (\mathbb{Z}_p^{(d+1)\times 1})^{w_2+1}$, and obtain $\bar{C}_0 = e(g_1, g_2)^{\boldsymbol{\alpha}^\top \boldsymbol{B}\left(\begin{smallmatrix} \boldsymbol{s}_0 \\ 0 \end{smallmatrix}\right)} \cdot M_b$ and $g_1^{\boldsymbol{c}_{\boldsymbol{B}}(\boldsymbol{S}, \mathbb{H})}$ by Eq.(9).

2. $\mathcal{B}$ computes the semi-functional part of the ciphertext by using $\boldsymbol{U}$ as follows. $\mathcal{B}$ implicitly sets $\hat{\boldsymbol{S}} = \left(\left(\begin{smallmatrix} \boldsymbol{0} \\ \hat{s}_0 \end{smallmatrix}\right), \left(\begin{smallmatrix} \boldsymbol{0} \\ \hat{s}_1 \end{smallmatrix}\right), \ldots, \left(\begin{smallmatrix} \boldsymbol{0} \\ \hat{s}_{w_2} \end{smallmatrix}\right)\right) \in (\mathbb{Z}_p^{(d+1)\times 1})^{w_2+1}$ by using $\hat{\boldsymbol{s}} = (\hat{s}_1, \ldots, \hat{s}_{w_2})$ that is defined in $\boldsymbol{U}$. This is done by computing
$$g_1^{\boldsymbol{c}'_{\boldsymbol{B}}(\hat{\boldsymbol{S}}, \mathbb{H}, \hat{\boldsymbol{h}})} = \left\{ g_1^{\boldsymbol{B}\left(\begin{smallmatrix} \boldsymbol{0} \\ c_\iota(\hat{\boldsymbol{s}}, \hat{\boldsymbol{h}}) \end{smallmatrix}\right)} \prod_{\substack{j\in[0,w_2] \\ k\in[1,n]}} g_1^{a_{\iota,j,k}\boldsymbol{H}_k \boldsymbol{B}\left(\begin{smallmatrix} \boldsymbol{0} \\ \hat{s}_j \end{smallmatrix}\right)} \right\}_{\iota\in[1,w_1]}.$$
We argue that this is computable as follows. The first term $g_1^{\boldsymbol{B}\left(\begin{smallmatrix} \boldsymbol{0} \\ c_\iota(\hat{\boldsymbol{s}}, \hat{\boldsymbol{h}}) \end{smallmatrix}\right)}$ can be computed from $D_i = g_1^{c_\iota(\hat{\boldsymbol{s}}, \hat{\boldsymbol{h}})}$ and the known $\boldsymbol{B}$, where we write $\boldsymbol{U} = (U_1, \ldots, U_{w_1})$. The second term (the product term) can be computed since for each $j \in [1, w_2]$ we either have the following.
   - For j such that there is ι' where $c_{\iota'}(\hat{\boldsymbol{s}}, \hat{\boldsymbol{h}}) = \hat{s}_j$, we have $g_1^{\hat{s}_j}$ available as $U_{\iota'} = g_1^{c_{\iota'}(\hat{\boldsymbol{s}}, \hat{\boldsymbol{h}})} = g_1^{\hat{s}_j}$ in $\boldsymbol{U}$.
   - For j such that there is no ι' as above, we have that by *regularity* of encoding (the third rule, in Definition 1), $a_{\iota'',j,k} = 0$ for all ι'', k.

   From these two facts and the known $\boldsymbol{B}$ and $\boldsymbol{H}_k$, we can compute the product term.

3. $\mathcal{B}$ also computes $\hat{C}_0 = e(g_1^{\boldsymbol{\alpha}^\top \boldsymbol{B}\left(\begin{smallmatrix} \boldsymbol{0} \\ \hat{s}_0 \end{smallmatrix}\right)}, g_2)$. This can be done since $c_1 = g_1^{\hat{s}_0}$ is available in $\boldsymbol{U}$, also due to the *regularity* of encoding (the fourth rule, in Definition 1).

4. $\mathcal{B}$ computes $\boldsymbol{C} = g_1^{\boldsymbol{c}_{\boldsymbol{B}}(\boldsymbol{S}, \mathbb{H})} g_1^{\boldsymbol{c}'_{\boldsymbol{B}}(\hat{\boldsymbol{S}}, \mathbb{H}, \hat{\boldsymbol{h}})}$ and $C_0 = \bar{C}_0 \hat{C}_0$, and returns $\mathsf{CT} = (\boldsymbol{C}, C_0)$.

This is a properly distributed semi-functional ciphertext of Equation (14).

Phase 2. When $\mathcal{A}$ makes the j-th private key query for $X_{\mathrm{j}} \in \mathbb{X}$, $\mathcal{B}$ does as follows. The algorithm $\mathcal{B}$ makes a key query X_{j} to its challenger and receives $\boldsymbol{V} = g_2^{\boldsymbol{k}(\beta,\hat{\boldsymbol{r}},\hat{\boldsymbol{h}})}$ where $(\boldsymbol{k}; m_2) = \mathsf{Enc1}(X_{\mathrm{j}})$. This query can be made since $R(X_{\mathrm{j}}, Y^\star) = 0$. These are the challenges for $\mathcal{B}$ to guess if $\beta = 0$ or β is randomly chosen in $\mathbb{Z}_p$, where we note that β is the same for all queries. This matches our definition of all the post-challenge keys, which use the same β, as defined in Figure 1.

1. $\mathcal{B}$ computes the normal part of the key as usual. More precisely, this is done by picking $\boldsymbol{\alpha} \xleftarrow{\$} \mathbb{Z}_p^{(d+1)\times 1}$, $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_{m_2} \xleftarrow{\$} \mathbb{Z}_p^{d\times 1}$ and setting $\boldsymbol{R} := \left(\left(\begin{smallmatrix} \boldsymbol{r}_1 \\ 0 \end{smallmatrix} \right), \ldots, \left(\begin{smallmatrix} \boldsymbol{r}_{m_2} \\ 0 \end{smallmatrix} \right) \right) \in (\mathbb{Z}_p^{(d+1)\times 1})^{m_2}$, and obtaining $g_2^{\boldsymbol{k}_{\boldsymbol{Z}}(\boldsymbol{\alpha},\boldsymbol{R},\mathbb{H})}$ by Eq.(11).

2. $\mathcal{B}$ computes the semi-functional part of the key by using $\boldsymbol{V}$ as follows. $\mathcal{B}$ implicitly sets $\hat{\boldsymbol{R}} = \left(\left(\begin{smallmatrix} \boldsymbol{0} \\ \hat{r}_1 \end{smallmatrix} \right), \ldots, \left(\begin{smallmatrix} \boldsymbol{0} \\ \hat{r}_{m_2} \end{smallmatrix} \right) \right) \in (\mathbb{Z}_p^{(d+1)\times 1})^{m_2}$ by using $\hat{\boldsymbol{r}} = (\hat{r}_1, \ldots, \hat{r}_{m_2})$ that is defined in $\boldsymbol{V}$. It also implicitly sets β as β that is defined in $\boldsymbol{V}$. These are done by computing

$$g_2^{\boldsymbol{k}'_{\boldsymbol{Z}}(\beta,\hat{\boldsymbol{R}},\mathbb{H},\hat{\boldsymbol{h}})} = \left\{ g_2^{\boldsymbol{Z}\left(\begin{smallmatrix} \boldsymbol{0} \\ k_\iota(\beta,\hat{\boldsymbol{r}},\hat{\boldsymbol{h}}) \end{smallmatrix} \right)} \prod_{\substack{j\in[1,m_2] \\ k\in[1,n]}} g_2^{b_{\iota,j,k}\boldsymbol{H}_k^\top \boldsymbol{Z}\left(\begin{smallmatrix} \boldsymbol{0} \\ \hat{r}_j \end{smallmatrix} \right)} \right\}_{\iota\in[1,m_1]}.$$

We argue that this is computable as follows. The first term $g_2^{\boldsymbol{Z}\left(\begin{smallmatrix} \boldsymbol{0} \\ k_\iota(\beta,\hat{\boldsymbol{r}},\hat{\boldsymbol{h}}) \end{smallmatrix} \right)}$ can be computed from $V_\iota = g_2^{k_\iota(\beta,\hat{\boldsymbol{r}},\hat{\boldsymbol{h}})}$ and the known $\boldsymbol{Z}$, where we write $\boldsymbol{V} = (V_1, \ldots, V_{m_1})$. The second term (the product term) can be computed since for each $j \in [1, m_2]$ we either have the following.

   – For j such that there is ι' where $k_{\iota'}(\beta, \hat{\boldsymbol{r}}, \hat{\boldsymbol{h}}) = \hat{r}_j$, we have $g_2^{\hat{r}_j}$ available as $V_{\iota'} = g_2^{k_{\iota'}(\beta,\hat{\boldsymbol{r}},\hat{\boldsymbol{h}})} = g_2^{\hat{r}_j}$ in $\boldsymbol{V}$.

   – For j such that there is no ι' as above, we have that by *regularity* of encoding (the second rule, in Definition 1), $b_{\iota'',j,k} = 0$ for all ι'', k.

   From these two facts and the known $\boldsymbol{Z}$ and $\boldsymbol{H}_k$, we can compute the product term.

3. $\mathcal{B}$ outputs $\mathsf{SK} = g_2^{\boldsymbol{k}_{\boldsymbol{Z}}(\boldsymbol{\alpha},\boldsymbol{R},\mathbb{H})} g_2^{\boldsymbol{k}'_{\boldsymbol{Z}}(\beta,\hat{\boldsymbol{R}},\mathbb{H},\hat{\boldsymbol{h}})}$.

This is a properly distributed semi-functional key of type-1 if $\beta = 0$ or type-2 if β is random.

Guess. The algorithm $\mathcal{B}$ has properly simulated G_{q_1+1} if $\beta = 0$, and G_{q_1+2} if β is random. Hence, $\mathcal{B}$ can use the output of $\mathcal{A}$ to guess β. □

5.8 Type-2 to Type-3 Semi-functional Key in Phase 2

Lemma 11 (G_{q_1+2} to G_{q_1+3})**.** *For any adversary $\mathcal{A}$ against* ABE*, there exists an algorithm $\mathcal{B}$ that breaks the $\mathcal{D}_d$-Matrix-DH Assumption with* $|\mathsf{G}_{q_1+2}\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ABE}}(\lambda) - \mathsf{G}_{q_1+3}\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ABE}}(\lambda)| \leq \mathsf{Adv}_{\mathcal{B}}^{\mathcal{D}_d\text{-}\mathsf{MatDH}}(\lambda)$.

Proof. The proof is exactly the same as that of Lemma 9, where we switched every key in phase 2 from normal to type-1 semi-functional all at once, except that every key has an additional vector $g_2^{\boldsymbol{k}_{\boldsymbol{Z}}(\boldsymbol{Z}\binom{\mathbf{0}}{\beta}, \mathbf{0}, \mathbb{H})} = \{g_2^{b_\iota \boldsymbol{Z}\binom{\mathbf{0}}{\beta}}\}_{\iota \in [1, m_1]}$ multiplied with it, where we note that β is the same across all the keys in phase 2 (as defined in Figure 1). To compute this, $\mathcal{B}$ samples $\beta \in \mathbb{Z}_p$ *at the beginning of phase 2*, and computes $= g_2^{\boldsymbol{Z}\binom{\mathbf{0}}{\beta}} = g_2^{\boldsymbol{Z}\binom{\mathbf{0}}{1}\beta}$. We recall that $\mathcal{B}$ possesses $\widehat{\mathsf{MSK}}_{\mathsf{base}} = g_2^{\boldsymbol{Z}\binom{\mathbf{0}}{1}}$. Following the proof of Lemma 9, we can see that if $\hat{\boldsymbol{y}} = \mathbf{0}$ then the key is a type-3 semi-functional, and if $\hat{\boldsymbol{y}} \xleftarrow{\$} \mathbb{Z}_p^{1 \times (q_2 m_2)}$ then the key is type-2 semi-functional. The rest of the proof follows exactly the same as that of Lemma 9. □

5.9 Final Game

Lemma 12 (G_{q_1+3} to $\mathsf{G}_{\mathrm{final}}$)**.** *We have* $\mathsf{G}_{q_1+3}\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ABE}}(\lambda) = \mathsf{G}_{\mathrm{final}}\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ABE}}(\lambda)$.

Proof. Since $\boldsymbol{Z} \in \mathbb{GL}_{p,d+1}$, we can define $\begin{pmatrix} \boldsymbol{\delta} \\ \hat{\delta} \end{pmatrix} = \boldsymbol{Z}^{-1}\boldsymbol{\alpha}$, where $\boldsymbol{\delta} \in \mathbb{Z}_p^{d \times 1}, \hat{\delta} \in \mathbb{Z}_p$. That is, $\boldsymbol{\alpha} = \boldsymbol{Z}\begin{pmatrix} \boldsymbol{\delta} \\ \hat{\delta} \end{pmatrix}$. We first claim that in G_{q_1+3}, $\hat{\delta}$ is uniformly random in $\mathbb{Z}_p$ to the adversary $\mathcal{A}$'view. The claim holds since every appearance of $\hat{\delta}$ in all the semi-functional keys (of type-3) is added by a uniformly random value: β_j for each pre-challenge j-th key and β for all the post-challenge keys (as defined in Eq.(18) and Figure 1). Hence, $\hat{\delta}$ in C_0 in the challenge ciphertext is uniformly random to $\mathcal{A}$. Therefore, we can modify $\hat{\delta}$ to $\hat{\delta} + u$ for uniformly random $u \xleftarrow{\$} \mathbb{Z}_p$. This results in changing $\boldsymbol{\alpha}$ to $\boldsymbol{\alpha} + \boldsymbol{Z}\binom{\mathbf{0}}{u}$ and hence modifying $C_0 = e(g_1, g_2)^{\boldsymbol{\alpha}^\top \boldsymbol{B}\binom{\boldsymbol{s}_0}{\hat{s}_0}} \cdot M$ to C_0' where

$$C_0' = e(g_1, g_2)^{\boldsymbol{\alpha}^\top \boldsymbol{B}\binom{\boldsymbol{s}_0}{\hat{s}_0} + (\mathbf{0}\ u)\boldsymbol{Z}^\top \boldsymbol{B}\binom{\boldsymbol{s}_0}{\hat{s}_0}} \cdot M = e(g_1, g_2)^{\boldsymbol{\alpha}^\top \boldsymbol{B}\binom{\boldsymbol{s}_0}{\hat{s}_0}} e(g_1, g_2)^{u\hat{s}_0} \cdot M$$

where we can verify that $(\mathbf{0}\ u)\boldsymbol{Z}^\top \boldsymbol{B}\binom{\boldsymbol{s}_0}{\hat{s}_0} = (\mathbf{0}\ u)\begin{pmatrix} \tilde{\boldsymbol{D}}^\top & \mathbf{0} \\ \mathbf{0} & 1 \end{pmatrix}\binom{\boldsymbol{s}_0}{\hat{s}_0} = (\mathbf{0}\ u)\binom{\boldsymbol{s}_0}{\hat{s}_0} = u\hat{s}_0$, where we recall that $\boldsymbol{Z}^\top \boldsymbol{B} = \begin{pmatrix} \tilde{\boldsymbol{D}}^\top & \mathbf{0} \\ \mathbf{0} & 1 \end{pmatrix}$ by definition. Finally, we observe that C_0' encrypts $e(g_1, g_2)^{u\hat{s}_0} \cdot M$, which is uniformly random in $\mathbb{G}_T$ since $u \xleftarrow{\$} \mathbb{Z}_p$. This is exactly the description of the final game, and hence concludes the proof. □

6 New Instantiations from Our Framework

In this section, we describe new instantiations from our framework. We consider both existing and new predicates. For existing predicates, all new instantiations are obtained via our generic construction instantiated with existing encodings. These new schemes are the first fully-secure prime-order schemes for given predicates and properties. We provide a summary in Table 2, which shows prime-order schemes. Our instantiations are marked "**New**". For comparison, we also provide Table 1 which shows composite-order schemes.

As for new predicates, we consider variants of ABE for Branching Programs (ABE-BP) consisting of *unbounded* schemes and *constant-size* (key or ciphertext) schemes. See below.

Table 1: Composite-order ABE schemes (for comparing to Table 2)

Predicate	Properties		Unbounded		KP	CP	DP
	Security	Universe	Input	Multi-use			
ABE-PDS	full	-	-	-	A14 [1]	AY15 [6]	AY15 [6]
Unbounded ABE-MSP	selective	large	yes	yes	LW11 [36],	sub	sub
	full	small	yes	yes	sub	LW12 [37]	sub
	full	large	yes	no	sub	sub	sub
	full	large	yes	yes	A14 [1]	AY15 [6]	AY15 [6]
Short-Cipher ABE-MSP	selective	large	no	yes	sub	open	open
	full	large	no	yes	A14 [1]	open	open
Short-Key ABE-MSP	selective	large	no	yes	sub	sub	open
	full	large	no	yes	open	AY15 [6]	open
(Bounded) ABE-MSP	selective	large	no	yes	sub	sub	sub
	full	small	no	no	LOS+10 [38], A14 [1], W14 [51],	LOS+10 [38], A14 [1] W14 [51]	AY15 [6]
	full	large	no	no	sub	sub	AY15 [6]
ABE-RL	selective	small	-	-	sub	sub	sub
	full	large	-	-	A14 [1]	A14 [1]	AY15 [6]
(Bounded) ABE-BP	selective	-	no	-	GVW13 [27]	sub	sub

Acronym: "ABE-PDS" = *ABE for policy over doubly-spatial relations*, "ABE-MSP" = *ABE for monotone span programs*, "ABE-RL" = *ABE for regular languages*, "ABE-BP" = *ABE for branching programs.* "KP" means *key-policy.* "CP" means *ciphertext-policy.* "DP" means *dual-policy.* "sub" means *subsumed* (no previous work but is subsumed by another system with stronger properties such as full security or prime-order). "open" means *open problem.* "-" means *not defined.* "Unbounded input" refers to unbounded size of attribute set size per ciphertext in KP-ABE-MSP, attribute set size per key in CP-ABE-MSP, and input string in ABE-BP.

6.1 The First Prime-order Schemes for Existing Predicates

ABE for Policy over Doubly-Spatial Relation (ABE-PDS). The predicate of *policy over doubly-spatial encryption* was defined in [1]. It generalizes the notion of doubly-spatial encryption [30] and ABE for span programs (and formulae) into one primitive. By using exactly the same encodings in [1, 6], we automatically obtain the first fully-secure prime-order KP-ABE-PDS, CP-ABE-PDS, DP-ABE-PDS schemes. DP stands for *dual-policy* type [3] which is the conjunctive predicate of key-policy and ciphertext-policy.

ABE for Monotone Span Programs (ABE-MSP). We obtain various schemes for ABE-MSP, and hence ABE for boolean formulae, as follows.

– **Unbounded ABE-MSP.** We obtain the first fully-secure prime-order completely unbounded schemes for KP-ABE-MSP, CP-ABE-MSP, and DP-ABE-MSP. Such schemes should pose no bounds such as the attribute set or policy size per ciphertext or key, the

Table 2: Prime-order ABE schemes

Predicate	Properties		Unbounded		KP	CP	DP
	Security	Universe	Input	Multi-use			
ABE-PDS	full	-	-	-	**New**	**New**	**New**
Unbounded ABE-MSP	selective	large	yes	yes	RW13 [45]	RW13 [45]	sub
	full	small	yes	yes	sub	LW12 [37]	sub
	full	large	yes	no	OT12 [43]	OT12 [43]	sub
	full	large	yes	yes	**New**	**New**	**New**
Short-Cipher ABE-MSP	selective	large	no	yes	ALP11 [5]	open	open
	full	large	no	yes	**New**	open	open
Short-Key ABE-MSP	selective	large	no	yes	remark	sub	open
	full	large	no	yes	open	**New**	open
(Bounded) ABE-MSP	selective	large	no	yes	GPSW06 [28]	W11 [49]	AI09 [3]
	full	small	no	no	CGW15 [14], (**New**)	CGW15 [14], (**New**)	**New**
	full	large	no	no	OT10 [41], (**New**)	OT10 [41], (**New**)	**New**
ABE-RL	selective	small	-	-	W12 [50]	sub	sub
	full	large	-	-	**New**	**New**	**New**
Unbounded ABE-BP	full	-	yes	-	**New**	**New**	**New**
Short-Cipher ABE-BP	full	-	no	-	**New**	open	open
Short-Key ABE-BP	full	-	no	-	open	**New**	open
(Bounded) ABE-BP	full	-	no	-	CGW15 [14], (**New**)	CGW15 [14], (**New**)	**New**

Acronym: Also refer to the acronym of Table 1. "**New**" means new instantiations (implied by our framework) that are the first such schemes for given predicates and properties. "(**New**)" means new instantiations but not the first of a kind. "remark" means such a primitive existed based on other tools (namely, [10], based on the LWE assumption).

attribute universe size, and the number of attribute repetition (also called multi-use) in a policy. These are obtained from the encodings of [1, 6] (for KP-ABE-MSP and CP-ABE-MSP case), and of [6] (for DP-ABE-MSP case). Or equivalently, we can use the fact that any pair encoding for XP-ABE-PDS implies an encoding for completely-unbounded XP-ABE-MSP as a special case, where XP is KP, CP, or DP. This is shown for the key-policy case in [1], but is also straightforward for the ciphertext-policy and dual-policy case.

– **ABE-MSP with Short Ciphertexts.** We obtain the first fully-secure prime-order KP-ABE-MSP with short ciphertexts. By short ciphertexts, we mean that each ciphertext requires only a constant number of group elements. This is obtained from the encoding in [1]. Or equivalently, it is implied by KP-ABE-PDS [1]. Note that this primitive requires bounded-size attribute set per ciphertext.

- **ABE-MSP with Short Keys.** We obtain the first fully-secure prime-order CP-ABE-MSP with short keys. By short keys, we mean each key requires only a constant number of group elements. This is obtained from the encoding in [6]. Or equivalently, it is implied by CP-ABE-PDS [6]. Note that this primitive requires bounded-size attribute set per key.
- **(Bounded) ABE-MSP.** We obtain the first fully-secure prime-order bounded DP-ABE-MSP schemes for small and large universes. These systems require the bounds on the size of attribute sets per keys and ciphertexts (in DP-ABE-MSP, we have both KP and CP parts). This is obtained from the encoding in [6]. Nevertheless, the underlying security of the encoding is perfectly master-key hiding, which is for free (no assumption needed for it), hence these systems use only the DLIN assumption (or any Matrix-DH) required for our framework. We remark that we also obtain bounded CP-ABE-MSP, KP-ABE-MSP schemes for small and large universes from the encodings in [1]. But these are not the first of a kind. The work by [41] already subsumes them.

ABE for Regular Languages (ABE-RL). In KP-ABE for regular languages, we have a key associated to the description of a deterministic finite automata (DFA) M, while a ciphertext is associated to a string w, and $R(M, w) = 1$ if the automata M accepts the string w. We refer to [50, 1] for detailed definitions. The CP-ABE is its dual where a key is associated to a string, and a ciphertext is associated to a DFA. We obtain the first fully-secure prime-order KP-ABE, CP-ABE, DP-ABE for regular languages. These are obtained from the encodings in [1] (for KP-ABE and CP-ABE) and in [6] (for DP-ABE). We show the encoding construction of KP-ABE for regular languages of [1] in §F.

6.2 New Predicates

In this section, we describe the first unbounded KP-ABE, CP-ABE for branching program, the first KP-ABE for branching program with constant-size ciphertexts, and the first CP-ABE for for branching program with constant-size keys. These are the first such instantiations with given properties (unbounded, constant-size), *even among composite-order schemes*. We obtain these via a new generic implication from any ABE for monotone span programs. We begin with definitions.

Predicate Definition for (Unbounded) ABE for Branching Program (ABE-BP). In (key-policy) ABE for Branching Program (ABE-BP) [27], we have a key associated to the description of a branching program Γ, while a ciphertext is associated to a binary string $w \in \{0,1\}^m$, where m is a bound for length of strings, and $R_m(\Gamma, w) = 1$ if the branching program Γ accepts the string w. More precisely, ABE-RP considers the predicate family $\{R_m\}_{m\in\mathbb{N}}$; hence, the setup of the ABE system (and public key) depends on m. On the other hand, in *Unbounded* ABE-BP, such a bound m is not required; that is, a ciphertext can be defined for a string w of any (polynomial-size) length in $\{0,1\}^*$.

We recall that a branching program Γ is a directed acyclic graph in which every non-terminal node has exactly two outgoing edges labeled $(i, 0)$ and $(i, 1)$ for some $i \in \mathbb{N}$. For an edge j, denote its label as ℓ_j. In bounded ABE-BP with the bound m, we require $i \leq m$; while in unbounded ABE-BP, i is not bounded. Moreover, there is a distinguished terminal node called accept node. We can assume wlog that there is exactly one start node. Every input binary string w induces a subgraph Γ_w that contains exactly all the edges labeled (i, w_i)

for $i \in [1, |w|]$, where we write $w = (w_1, \ldots, w_{|w|})$ as the binary representation of w. We say that Γ accepts w if there is a path from the start node to the accept node in Γ_w. We can assume wlog that there is at most only one edge connecting any two nodes in Γ (See [27]).

ABE-MSP Implies ABE-BP. We will show that ABE for monotone span program (ABE-MSP) for large universe implies ABE for branching program. In particular, unbounded or constant-size properties are preserved via the implication.

We first recall the predicate definition of ABE-MSP [28, 38, 45]. Let $\mathcal{U}$ be the universe of attributes. If $|\mathcal{U}|$ is of super-polynomial size, it is called large universe [28, 45]. A monotone span program is specified by (M, ρ) where M is an integer matrix of dimension $r \times c$ for some r, c and $\rho : [1, r] \to \mathcal{U}$. For $S \in \mathcal{U}$, let $M|_S$ be the sub-matrix of M that takes all the rows j such that $\rho(j) \in S$. We say that (M, ρ) accepts a set of attributes $S \in \mathcal{U}$ if $(1, 0, \ldots, 0) \in \mathrm{rowspan}(M|_S)$.

Theorem 13. *ABE-MSP for large universe implies ABE-BP.*

Proof. To show the implication, we will define two maps: π_1 maps a branching program to a monotone span program, and π_2 maps a string to a set of attributes, as follows. First, let E_Γ, V_Γ be the sets of edges and nodes of Γ, respectively. We can assume wlog that $E_\Gamma = [1, |E_\Gamma|]$, $V_\Gamma = [1, |V_\Gamma|]$, and the terminal accept node is denoted by the number 1, while the start node is denoted by $|V_\Gamma|$. An edge j is directed from its tail node, denoted t_j, to its head, denoted h_j.

We define $\pi_1 : \Gamma \mapsto (M, \rho)$ for which an access matrix M of dimension $|E_\Gamma| \times (|V_\Gamma| - 1)$ and a row label function $\rho : E_\Gamma \to \mathbb{N} \times \{0, 1\}$ are set as follows. We define the entry (j, k) of M as follows:

$$M_{j,k} := \begin{cases} -1 & \text{if } k \text{ is the tail node of edge } j, \\ 1 & \text{if } k \text{ is the head node of edge } j, \\ 0 & \text{otherwise.} \end{cases}$$

We define ρ to map the row number $j \in [1, |E_\Gamma|]$ to the label ℓ_j of the edge j in Γ. We define $\pi_2 : \{0, 1\}^* \to 2^{\mathbb{N} \times \{0,1\}}$ by mapping $\pi_2 : w \mapsto \{\, (i, w_i) \mid i \in [1, |w|] \,\}$. We claim the following lemma.

Lemma 14. *The branching program Γ accepts the string w if and only if the monotone span program $\pi_1(\Gamma)$ accepts the attribute set $\pi_2(w)$.*

We can naturally use the maps π_1 and π_2 to construct an (unbounded) ABE-BP scheme from an (unbounded) ABE for monotone span programs (ABE-MSP): a ciphertext for w in ABE-BP is constructed by encrypting $\pi_2(w)$ in ABE-MSP, while a key for Γ in ABE-BP is obtained as a key for $\pi_1(\Gamma)$ in ABE-MSP. The *if* and the *only-if* part of the lemma will guarantee that the *security* and the *correctness* are preserved, respectively.[10] This concludes the proof of the theorem. □

It remains to prove the above lemma. Let $(M, \rho) = \pi_1(\Gamma)$, $S = \pi_2(w)$.

[10] This was formalized and called the embedding lemma in [12].

Proof of Lemma 14. We first proof for $(\Rightarrow)$. Assume that Γ accepts w. Hence, there exists a path from the start node (node $|V_\Gamma|$) to the accept node (node 1) in Γ_w. Let $(|V_\Gamma|, v_1, \ldots, v_t, 1)$ be the nodes on the path. Also let $(j_1, \ldots, j_{t+1})$ be the edges on this path. Hence, by definition, $M|_S$ is the matrix that exactly takes the row $j_1, \ldots, j_{t+1}$ from M. Now, consider the sum of all row vectors in $M|_S$. We have:

- The row j_{t+1} contributes 1 at column 1, and -1 at column v_t.
- For i from t down to 2, the row j_i contributes 1 at column v_i, and -1 at column v_{i-1}.
- The row j_1 contributes 1 at column v_1.

Therefore, all the values at column $v_1, \ldots, v_t$ are canceled out to 0. Hence, the sum is exactly $(1, 0, \ldots, 0)$. Therefore, $(1, 0, \ldots, 0) \in \mathrm{rowspan}(M|_S)$ which means that (M, ρ) accepts S.

We now prove for $(\Leftarrow)$. Assume that Γ does not accept w. Let Γ'_w be the *undirected* graph obtained from Γ_w by treating every edge as an undirected edge. We have the following properties:[11]

1. Since Γ does not accept w, we have that the start node and the accept node lie in different connected components of Γ'_w.
2. Γ'_w contains no cycle. This is since Γ_w is acyclic and every non-terminal node has exactly one outgoing edge.

Assume for contradiction that $(1, 0, \ldots, 0) \in \mathrm{rowspan}(M|_S)$. Let $J = \{\, j \mid \rho(j) \in S \,\}$. (Hence, J is the set of edges of Γ_w). We write this linear combination as $(1, 0, \ldots, 0) = \sum_{j \in J} c_j M_j$, where M_j is the row j of M and c_j is some coefficient. For each node $k \in [1, |V_\Gamma|]$, let J_k be the set of edges j in J that are adjacent to k and that $c_j \neq 0$. From the linear combination, we must have that:

$$\sum_{j \in J_k} c_j M_{j,k} = \begin{cases} 1 & \text{if } k = 1, \\ 0 & \text{if } k \in [2, |V_\Gamma| - 1]. \end{cases} \tag{27}$$

Let Γ''_w be the subgraph of Γ'_w that takes all edges $j \in J$ such that $c_j \neq 0$ (while treating as undirected edges). We observe that for every node $k \in [2, |V_\Gamma| - 1]$ (*i.e.,* not the accept nor the start node), there are *at least two edges adjacent to it in* Γ''_w. This is since otherwise the sum $\sum_{j \in J_k} c_j M_{j,k}$ would not be canceled out to 0, where we observe that for $j \in J_k$ we have $M_{j,k} \neq 0$. We claim that Γ''_w will always contain a cycle. Hence, this will contradict the property 2, and the proof will be concluded. It now remains to prove the claim. We consider an arbitrary node $k \in [2, |V_\Gamma| - 1]$. We have three cases:

- If k is not connected to neither the accept nor the start node in Γ''_w, then the largest connected subgraph of Γ''_w that contains k has all nodes with at least two adjacent edges.
- If k is connected to the accept node, then it is not connected to the start node by the property 1. Hence, the largest connected subgraph of Γ''_w that contains k has at most one node (the accept node) that may have one adjacent edge.

[11]These properties were also used, albeit differently, for proving selective security for the ABE-BP of [27].

- If k is connected to the start node, then it is not connected to the accept node by the property 1. Hence, the largest connected subgraph of Γ''_w that contains k has at most one node (the start node) that may have one adjacent edge.

In all cases, the considering connected subgraph has at most one node that may have one adjacent edge. Hence, it always contain a cycle. This concludes the proof of the claim, and hence the lemma. □

Instantiations of ABE for Branching Programs. We obtain the following schemes.

– **Unbounded ABE-BP.** Observing the proof of Theorem 13, we can see that if key-policy ABE-MSP requires no bound for attribute set per ciphertext, then the resulting key-policy ABE-BP is unbounded. Analogously, if ciphertext-policy ABE-MSP requires no bound for attribute set per key, then the resulting ciphertext-policy ABE-BP is unbounded. Hence, we can instantiate these from our unbounded ABE-MSP. We also obtain dual-policy schemes.

– **ABE-BP with Short Ciphertexts.** Again, observing the proof of Theorem 13, we have that the property of constant-size ciphertexts is preserved. Starting from our KP-ABE-MSP with constant-size ciphertexts, we obtain KP-ABE-BP with constant-size ciphertexts.

– **ABE-BP with Short Keys.** Analogously, the property of constant-size keys is preserved. Starting from our CP-ABE-MSP with constant-size keys, we obtain CP-ABE-BP with constant-size keys.

– **(Bounded) ABE-BP.** We also obtain bounded ABE-BP from bounded ABE-MSP. Independently, Chen *et al.* [14] also obtains (bounded) ABE for branching programs (as a special case of their schemes for arithmetic branching programs).

Acknowledgement. I would like to thank Shota Yamada and anonymous reviewers of TCC 2015 for their useful comments.

References

[1] N. Attrapadung. Dual System Encryption via Doubly Selective Security: Framework, Fully-secure Functional Encryption for Regular Languages, and More. In *Eurocrypt 2014*, pp. 557-577, 2014.

[2] N. Attrapadung. Fully Secure and Succinct Attribute Based Encryption for Circuits from Multi-linear Maps. Cryptology ePrint Archive: Report 2014/772.

[3] N. Attrapadung, H. Imai. Dual-Policy Attribute Based Encryption. In *ACNS'09*, pp. 168–185, 2009.

[4] N. Attrapadung, B. Libert. Functional Encryption for Inner Product: Achieving Constant-Size Ciphertexts with Adaptive Security or Support for Negation. In *PKC 2010*, pp. 384–402, 2010.

[5] N. Attrapadung, B. Libert, E. Panafieu. Expressive Key-Policy Attribute-Based Encryption with Constant-Size Ciphertexts In *PKC 2011*, pp. 90–108, 2010.

[6] N. Attrapadung, S. Yamada. Duality in ABE: Converting Attribute Based Encryption for Dual Predicate and Dual Policy via Computational Encodings. In *CT-RSA 2015*, to appear, 2015.

[7] J. Bethencourt, A. Sahai, B. Waters. Ciphertext-Policy Attribute-Based Encryption. IEEE Symposium on Security and Privacy (S&P), pp. 321-334, 2007.

[8] D. Boneh, X. Boyen. Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles. In *Eurocrypt 2004*, *LNCS* 3027, pp. 223–238, 2004.

[9] D. Boneh, X. Boyen, H. Shacham. Short Group Signatures. In *Crypto 2004*, pp. 41-55, 2004.

[10] D. Boneh, C. Gentry, S. Gorbunov, S. Halevi, V. Nikolaenko, G. Segev, V. Vaikuntanathan, D. Vinayagamurthy. Fully Key-Homomorphic Encryption, Arithmetic Circuit ABE and Compact Garbled Circuits. In *Eurocrypt 2014*, pp. 533–556, 2014.

[11] D. Boneh, E. Goh, K. Nissim. Evaluating 2-DNF Formulas on Ciphertexts. In *TCC 2005*, pp. 325-341, 2005.

[12] D. Boneh, M. Hamburg. Generalized Identity Based and Broadcast Encryption Schemes. In *Asiacrypt 2008*, *LNCS* 5350, pp. 455–470, 2008.

[13] D. Boneh, A. Sahai, B. Waters. Functional Encryption: Definitions and Challenges. In *TCC 2011*, *LNCS* 6597, pp. 253–273, 2011.

[14] J. Chen, R. Gay, H. Wee. Improved Dual System ABE in Prime-Order Groups via Predicate Encodings. In *Eurocrypt'15*, 2015.

[15] J. Chen, H. Wee. Fully, (Almost) Tightly Secure IBE from Standard Assumptions. In *Crypto'13*, pp. 435-460, 2013.

[16] J. Coron, T. Lepoint, M. Tibouchi. Practical Multilinear Maps over the Integers. In *Crypto'13*, 2013.

[17] A. Escala, G. Herold, E. Kiltz, C. Rafols, J. L. Villar. An Algebraic Framework for Diffie-Hellman Assumptions. In *Crypto'13*, pp. 129–147, 2013.

[18] D. M. Freeman. Converting Pairing-Based Cryptosystems from Composite-Order Groups to Prime-Order Groups. In *Eurocrypt'10*, pp. 44–61, 2013.

[19] S. Garg, C. Gentry, S. Halevi. Candidate multilinear maps from ideal lattices In *Eurocrypt'13*, 2013.

[20] S. Garg, C. Gentry, S. Halevi, A. Sahai, B. Waters. Attribute-based encryption for circuits from multilinear maps. In *Crypto'13*, pp. 479-499, 2013.

[21] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, B. Waters. Candidate Indistinguishability Obfuscation and Functional Encryption for all circuits. In *FOCS 2013*, 2013.

[22] S. Garg, C. Gentry, S. Halevi, M. Zhandry. Fully Secure Attribute Based Encryption from Multilinear Maps. Cryptology ePrint Archive: Report 2014/622.

[23] S. Garg, C. Gentry, A. Sahai, B. Waters. Witness encryption and its applications. In *STOC 2013*, pp. 467–476, 2013.

[24] C. Gentry, A. Lewko, B. Waters. Witness Encryption from Instance Independent Assumptions. In *Crypto 2014*, pp. 426–443, 2014.

[25] S. Goldwasser, Y. Kalai, R.A. Popa, V. Vaikuntanathan, N. Zeldovich. Reusable garbled circuits and succinct functional encryption. In *STOC 13*, 2013.

[26] S. Goldwasser, Y. Kalai, R.A. Popa, V. Vaikuntanathan, N. Zeldovich. How to run Turing machines on encrypted data. In *Crypto'13*, pp. 536-553, 2013.

[27] S. Gorbunov, V. Vaikuntanathan, H. Wee. Attribute-based encryption for circuits. In *STOC13*, 2013.

[28] V. Goyal, O. Pandey, A. Sahai, B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM CCS'06*, pp. 89–98, 2006.

[29] A. Guillevic. Comparing the Pairing Efficiency over Composite-Order and Prime-Order Elliptic Curves. In *ACNS 2013*, pp. 357-372, 2013.

[30] M. Hamburg. Spatial Encryption (Ph.D. Thesis). Cryptology ePrint Archive: Report 2011/389.

[31] Y. Ishai, H. Wee. Partial Garbling Schemes and Their Applications. In *ICALP (1) 2014*, pp. 650–662, 2014.

[32] J. Katz, A. Sahai, B. Waters. Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products. In *Eurocrypt 2008*, *LNCS* 4965, pp. 146-162.

[33] A. Lewko Tools for Simulating Features of Composite Order Bilinear Groups in the Prime Order Setting. In *Eurocrypt 2012*. pp. 318-335, 2012.

[34] A. Lewko, B. Waters. New Techniques for Dual System Encryption and Fully Secure HIBE with Short Ciphertexts. In *TCC 2010*, LNCS 5978, pp. 455-479, Springer, 2010.

[35] A. Lewko, B. Waters. Decentralizing Attribute-Based Encryption In *Eurocrypt 2011*. pp. 568-588, 2011.

[36] A. Lewko, B. Waters. Unbounded HIBE and Attribute-Based Encryption In *Eurocrypt 2011*. pp. 547-567, 2011.

[37] A. Lewko, B. Waters. New Proof Methods for Attribute-Based Encryption: Achieving Full Security through Selective Techniques. In *Crypto 2012*. pp. 180-198, 2012.

[38] A. Lewko, T. Okamoto, A. Sahai, K. Takashima, B. Waters. Fully Secure Functional Encryption: Attribute-Based Encryption and (Hierarchical) Inner Product Encryption. In *Eurocrypt 2010*. pp. 62-91, 2010.

[39] S. Meiklejohn, H. Shacham, D. M. Freeman. Limitations on Transformations from Composite-Order to Prime-Order Groups: The Case of Round-Optimal Blind Signatures. In *Asiacrypt 2010*, pp. 519–538, 2010.

[40] T. Okamoto, K. Takashima. Hierarchical Predicate Encryption for Inner-Products. In *Asiacrypt 2009*, *LNCS* 5912, pp. 214–231, 2009.

[41] T. Okamoto, K. Takashima, Fully secure functional encryption with general relations from the decisional linear assumption. In *Crypto 2010*, *LNCS* 6223, pp. 191-208, 2010.

[42] T. Okamoto, K. Takashima, Adaptively Attribute-Hiding (Hierarchical) Inner Product Encryption. In *Eurocrypt 2012*, *LNCS* 7237, pp. 591-608, 2012.

[43] T. Okamoto, K. Takashima, Fully Secure Unbounded Inner-Product and Attribute-Based Encryption. In *Asiacrypt 2012*, pp. 349-366, 2012.

[44] R. Ostrovsky, A. Sahai, B. Waters. Attribute-based encryption with non-monotonic access structures. In *ACM CCS 2007*, pp. 195–203, 2007.

[45] Y. Rouselakis, B. Waters Practical constructions and new proof methods for large universe attribute-based encryption. In *ACM CCS 2013*, pp. 463–474, 2013.

[46] A. Sahai, B. Waters. Fuzzy Identity-Based Encryption In *Eurocrypt 2005*, *LNCS* 3494, pp. 457–473, 2005.

[47] J. H. Seo, J. H. Cheon. Beyond the Limitation of Prime-Order Bilinear Groups, and Round Optimal Blind Signatures. In *TCC 2012*. pp. 133–150, 2012.

[48] B. Waters. Ciphertext-Policy Attribute-Based Encryption: An Expressive, Efficient, and Provably Secure Realization. In *PKC 2011*. pp. 53-70, 2011.

[49] B. Waters. Dual System Encryption: Realizing Fully Secure IBE and HIBE under Simple Assumptions. In *Crypto 2009*, pp. 619-636, 2009.

[50] B. Waters. Functional Encryption for Regular Languages. In *Crypto 2012*, pp. 218-235, 2012.

[51] H. Wee. Dual System Encryption via Predicate Encodings. In *TCC 2014*, pp. 616-637, 2014.

A Security Definition for ABE

Security Notion. An attribute-based encryption scheme for predicate family R is fully secure if no probabilistic polynomial time (PPT) adversary $\mathcal{A}$ has non-negligible advantage in the following game between $\mathcal{A}$ and the challenger $\mathcal{C}$. For our purpose of modifying games in the proof, we write some in the boxes. Let q_1, q_2 be the numbers of queries in Phase 1,2, respectively.

1. **Setup**: $\mathcal{C}$ runs $^{(1)}\boxed{\mathsf{Setup}(1^\lambda, \kappa) \to (\mathsf{PK}, \mathsf{MSK})}$ and hands PK to $\mathcal{A}$.

2. **Phase 1**: $\mathcal{A}$ makes a j-th private key query for $X_j \in \mathbb{X}_\kappa$. $\mathcal{C}$ returns SK_j by computing $^{(2)}\boxed{\mathsf{SK}_j \leftarrow \mathsf{KeyGen}(X_j, \mathsf{MSK}, \mathsf{PK})}$.

3. **Challenge**: $\mathcal{A}$ submits equal-length messages M_0, M_1 and a target ciphertext attribute $Y^\star \in \mathbb{Y}_\kappa$ with the restriction that $R_\kappa(X_j, Y^\star) = 0$ for all $j \in [1, q_1]$. $\mathcal{C}$ flips a bit $b \xleftarrow{\$} \{0,1\}$ and returns the challenge ciphertext $^{(3)}\boxed{\mathsf{CT}^\star \leftarrow \mathsf{Encrypt}(Y^\star, M_b, \mathsf{PK})}$.

4. **Phase 2**: $\mathcal{A}$ continues to make a j-th private key query for $X_j \in \mathbb{X}_\kappa$ under the restriction $R_\kappa(X_j, Y^\star) = 0$. $\mathcal{C}$ returns $^{(4)}\boxed{\mathsf{SK}_j \leftarrow \mathsf{KeyGen}(X_j, \mathsf{MSK}, \mathsf{PK})}$.

5. **Guess**: The adversary $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$ and wins if $b' = b$. The advantage of $\mathcal{A}$ against ABE is defined as $\mathsf{Adv}^{\mathsf{ABE}}_{\mathcal{A}}(\lambda) := |\Pr[b = b'] - \frac{1}{2}|$.

B Recap the Security Definitions for Pair Encodings

The security notions of pair encoding schemes are given in [1], with a refinement regarding the number of queries in [6]. We describe almost the same definitions here and remark slight differences from [1, 6] below.

(Perfect Security). The pair encoding scheme P is *perfectly master-key hiding* (PMH) if the following holds. Suppose $R(X,Y) = 0$. Let $n \leftarrow \mathsf{Param}(\kappa)$, $(\boldsymbol{k}; m_2) \leftarrow \mathsf{Enc1}(X)$, $(\boldsymbol{c}; w_2) \leftarrow \mathsf{Enc2}(Y)$, then the following two distributions are identical:

$$\{\boldsymbol{c}(\boldsymbol{s},\boldsymbol{h}),\ \boldsymbol{k}(0,\boldsymbol{r},\boldsymbol{h})\} \qquad \text{and} \qquad \{\boldsymbol{c}(\boldsymbol{s},\boldsymbol{h}),\ \boldsymbol{k}(\alpha,\boldsymbol{r},\boldsymbol{h})\},$$

where the probability is taken over $\boldsymbol{h} \xleftarrow{\$} \mathbb{Z}_p^n, \alpha \xleftarrow{\$} \mathbb{Z}_p, \boldsymbol{r} \xleftarrow{\$} \mathbb{Z}_p^{m_2}, \boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_p^{(w_2+1)}$.

(Computational Security). We define two flavors for computational security notions: *selectively* and *co-selectively secure master-key hiding* ($\mathsf{SMH}, \mathsf{CMH}$) in a bilinear group generator $\mathcal{G}$. We first define the following game template, denoted as $\mathsf{Exp}_{\mathcal{G},\mathsf{P},\mathsf{G},b,\mathcal{A},t_1,t_2}(\lambda)$, for pair encoding P, a flavor $\mathsf{G} \in \{\mathsf{CMH}, \mathsf{SMH}\}$, $b \in \{0,1\}$, and $t_1, t_2 \in \mathbb{N}$. It takes as input the security parameter λ and does the experiment with the adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, and outputs b'. Denote by st a state information by $\mathcal{A}$. The game is defined as:

$$\begin{aligned}\mathsf{Exp}_{\mathcal{G},\mathsf{G},b,\mathcal{A},t_1,t_2}(\lambda) : {} & (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, p) \leftarrow \mathcal{G}(\lambda);\ g_1 \xleftarrow{\$} \mathbb{G}_1, g_2 \xleftarrow{\$} \mathbb{G}_2, \\ & \alpha \xleftarrow{\$} \mathbb{Z}_p, n \leftarrow \mathsf{Param}(\kappa), \boldsymbol{h} \xleftarrow{\$} \mathbb{Z}_p^n; \\ & \mathsf{st} \leftarrow \mathcal{A}_1^{\mathcal{O}^1_{\mathsf{G},b,\alpha,\boldsymbol{h}}(\cdot)}(g_1, g_2);\ b' \leftarrow \mathcal{A}_2^{\mathcal{O}^2_{\mathsf{G},b,\alpha,\boldsymbol{h}}(\cdot)}(\mathsf{st}),\end{aligned}$$

where each oracle $\mathcal{O}^1, \mathcal{O}^2$ *can be queried at most* t_1, t_2 *times respectively*, and is defined as follows.

- **Selective Security.**

  - $\mathcal{O}^1_{\mathsf{SMH},b,\alpha,\boldsymbol{h}}(Y^\star)$: Run $(\boldsymbol{c}; w_2) \leftarrow \mathsf{Enc2}(Y^\star)$; $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_p^{(w_2+1)}$; return $\boldsymbol{U} \leftarrow g_1^{\boldsymbol{c}(\boldsymbol{s},\boldsymbol{h})}$.

- $\mathcal{O}^2_{\mathsf{SMH},b,\alpha,\boldsymbol{h}}(X)$: If $R(X, Y^\star) = 1$, then return $\perp$.
  Else, run $(\boldsymbol{k}; m_2) \leftarrow \mathsf{Enc1}(X); \boldsymbol{r} \xleftarrow{\$} \mathbb{Z}_p^{m_2}$; return

$$\boldsymbol{V} \leftarrow \begin{cases} g_2^{\boldsymbol{k}(0,\boldsymbol{r},\boldsymbol{h})} & \text{if } b = 0 \\ g_2^{\boldsymbol{k}(\alpha,\boldsymbol{r},\boldsymbol{h})} & \text{if } b = 1 \end{cases} .$$

- **Co-selective Security.**

- $\mathcal{O}^1_{\mathsf{CMH},b,\alpha,\boldsymbol{h}}(X^\star)$: Run $(\boldsymbol{k}; m_2) \leftarrow \mathsf{Enc1}(X^\star); \boldsymbol{r} \xleftarrow{\$} \mathbb{Z}_p^{m_2}$; return

$$\boldsymbol{V} \leftarrow \begin{cases} g_2^{\boldsymbol{k}(0,\boldsymbol{r},\boldsymbol{h})} & \text{if } b = 0 \\ g_2^{\boldsymbol{k}(\alpha,\boldsymbol{r},\boldsymbol{h})} & \text{if } b = 1 \end{cases} .$$

- $\mathcal{O}^2_{\mathsf{CMH},b,\alpha,\boldsymbol{h}}(Y)$: If $R(X^\star, Y) = 1$, then return $\perp$.
  Else, run $(\boldsymbol{c}; w_2) \leftarrow \mathsf{Enc2}(Y); \boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_p^{(w_2+1)}$; return $\boldsymbol{U} \leftarrow g_1^{\boldsymbol{c}(\boldsymbol{s},\boldsymbol{h})}$.

We define the advantage of $\mathcal{A}$ against the pair encoding scheme P in the security game $\mathsf{G} \in \{\mathsf{SMH}, \mathsf{CMH}\}$ for bilinear group generator $\mathcal{G}$ with the bounded number of queries (t_1, t_2) as

$$\mathsf{Adv}_{\mathcal{A}}^{(t_1,t_2)\text{-}\mathsf{G}(\mathsf{P})}(\lambda) := |\Pr[\mathsf{Exp}_{\mathcal{G},\mathsf{P},\mathsf{G},0,\mathcal{A},t_1,t_2}(\lambda) = 1] - \Pr[\mathsf{Exp}_{\mathcal{G},\mathsf{P},\mathsf{G},1,\mathcal{A},t_1,t_2}(\lambda) = 1]|$$

We say that P is (t_1, t_2)-*selectively master-key hiding* in $\mathcal{G}$ if $\mathsf{Adv}_{\mathcal{A}}^{(t_1,t_2)\text{-}\mathsf{SMH}(\mathsf{P})}(\lambda)$ is negligible for all polynomial time attackers $\mathcal{A}$. Analogously, P is (t_1, t_2)-*co-selectively master-key hiding* in $\mathcal{G}$ if $\mathsf{Adv}_{\mathcal{A}}^{(t_1,t_2)\text{-}\mathsf{CMH}(\mathsf{P})}(\lambda)$ is negligible for all polynomial time attackers $\mathcal{A}$.

Poly-many Queries. We also consider the case where t_i is *not a-priori bounded* and hence the corresponding oracle can be queried polynomially many times. In such a case, we denote t_i as poly.

Remark 3. The original notions considered in [1] are $(1, \mathsf{poly})$-SMH, $(1, 1)$-CMH for selective and co-selective master-key hiding security, respectively. The refinement with (t_1, t_2) is done recently in [6]. The purpose of refinement is that there exists a generic conversion that converts a $(1, 1)$-SMH-secure pair encoding scheme for a predicate into another scheme for its *dual predicate* which is $(1, 1)$-CMH-secure, and vice-versa [6]. We note that $(1, \mathsf{poly})$-SMH trivially implies $(1, 1)$-SMH.

Remark 4. The definition of computational security for encoding here is slightly different from that in [1, 6] in that here we define it in *asymmetric* and *prime-order* groups, while it was defined in *symmetric* and *prime-order subgroup of composite-order* groups in [1, 6]. We use asymmetric groups for the purpose of generality, one can obtain schemes in symmetric groups by just setting $\mathbb{G}_1 = \mathbb{G}_2$. Hence, we can use all the proposed encodings in [1, 6] by working on the symmetric group version of our framework. For the latter issue, the difference of definitions between prime-order groups and prime-order subgroups are merely *syntactic*. This is since although the original definition was defined in prime-order subgroups, the hardness of factorization was not assumed (*i.e.,* generators of each subgroup or even factors of composites N can be given out to the adversary). Hence, the encoding schemes in [1, 6] are secure in our definition under the security proofs in their present forms.

C Recap the Composite-order Construction of [1]

For self-containment, we recap the ABE construction in composite-order groups, albeit using asymmetric groups (the original scheme was defined in symmetric groups). It will use a composite-order asymmetric bilinear group generator $\mathcal{G}_{\mathsf{composite}}$ which outputs $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, N, p_1, p_2, p_3) \xleftarrow{\$} \mathcal{G}_{\mathsf{composite}}(\lambda)$, where $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are of order $N = p_1p_2p_3$. The bilinear map takes the form $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Let $\mathbb{G}_{1,p_i}, \mathbb{G}_{2,p_i}$ be the subgroup of order p_i of $\mathbb{G}_1, \mathbb{G}_2$ respectively. Let $g_1 \in \mathbb{G}_{1,p_1}$, $\hat{g}_1 \in \mathbb{G}_{1,p_2}$, $g_2 \in \mathbb{G}_{2,p_1}$, $\hat{g}_2 \in \mathbb{G}_{2,p_2}$ be a generator in each subgroup. The scheme is as follows.

- $\mathsf{Setup}(1^\lambda, \kappa)$: Run $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, N, p_1, p_2, p_3) \xleftarrow{\$} \mathcal{G}_{\mathsf{composite}}(\lambda)$. Pick generators $g_1 \xleftarrow{\$} \mathbb{G}_{1,p_1}$, $g_2 \in \mathbb{G}_{2,p_1}$, $Z_3 \xleftarrow{\$} \mathbb{G}_{2,p_3}$. Obtain $n \leftarrow \mathsf{Param}(\kappa)$. Pick $\boldsymbol{h} \xleftarrow{\$} \mathbb{Z}_N^n$ and $\alpha \xleftarrow{\$} \mathbb{Z}_N$. The public key is $\mathsf{PK} = \big(g_1, g_2, e(g_1, g_2)^\alpha, g_1^{\boldsymbol{h}}, Z_3\big)$. The master secret key is $\mathsf{MSK} = \alpha$.

- $\mathsf{Encrypt}(Y, M, \mathsf{PK})$: Upon input $Y \in \mathbb{Y}_N$, run $(\boldsymbol{c}; w_2) \leftarrow \mathsf{Enc2}(Y)$. Pick $\boldsymbol{s} = (s_0, s_1, \ldots, s_{w_2}) \xleftarrow{\$} \mathbb{Z}_N^{w_2+1}$. Output the ciphertext as $\mathsf{CT} = (\boldsymbol{C}, C_0)$:
$$\boldsymbol{C} = g_1^{\boldsymbol{c}(\boldsymbol{s},\boldsymbol{h})} \in \mathbb{G}^{w_1}, \qquad C_0 = (e(g_1, g_2)^\alpha)^{s_0} M \in \mathbb{G}_T.$$
Note that $\boldsymbol{C}$ can be computed from $g_1^{\boldsymbol{h}}$ and $\boldsymbol{s}$ since $\boldsymbol{c}(\boldsymbol{s}, \boldsymbol{h})$ contains only linear combinations of monomials s, s_i, sh_j, s_ih_j.

- $\mathsf{KeyGen}(X, \mathsf{MSK}, \mathsf{PK})$: Upon input $X \in \mathbb{X}_N$, run $(\boldsymbol{k}; m_2) \leftarrow \mathsf{Enc1}(X)$. Parse $\mathsf{MSK} = \alpha$. Recall that $m_1 = |\boldsymbol{k}|$. Pick $\boldsymbol{r} \xleftarrow{\$} \mathbb{Z}_N^{m_2}, \boldsymbol{R}_3 \xleftarrow{\$} \mathbb{G}_{2,p_3}^{m_1}$. Output the secret key SK:
$$\mathsf{SK} = g_2^{\boldsymbol{k}(\alpha,\boldsymbol{r},\boldsymbol{h})} \cdot \boldsymbol{R}_3 \in \mathbb{G}^{m_1}.$$

- $\mathsf{Decrypt}(\mathsf{CT}, \mathsf{SK})$: Obtain Y, X from CT, SK. Suppose $R(X, Y) = 1$. Run $\boldsymbol{E} \leftarrow \mathsf{Pair}(X, Y)$. Compute
$$e(g_1, g_2)^{\alpha s_0} \leftarrow \prod_{i\in[1,m_1], i'\in[1,w_1]} e(\boldsymbol{C}[i'], \mathsf{SK}[i])^{E_{i,i'}},$$
and obtain $M \leftarrow C_0/e(g_1, g_2)^{\alpha s_0}$.

Correctness. Since $R(X, Y) = 1$, we have
$$\prod_{i\in[1,m_1], i'\in[1,w_1]} e(\boldsymbol{C}[i'], \mathsf{SK}[i])^{E_{i,i'}} = e(g_1, g_2)^{\boldsymbol{k}\boldsymbol{E}\boldsymbol{c}^\top} = e(g_1, g_2)^{\alpha s_0}.$$

Semi-Functional Algorithms.

- $\mathsf{SFSetup}(1^\lambda, \kappa)$: This is exactly the same as $\mathsf{Setup}(1^\lambda, \kappa)$ except that it additionally outputs generators $\hat{g}_1 \in \mathbb{G}_{1,p_2}$, $\hat{g}_2 \in \mathbb{G}_{2,p_2}$ and semi-functional parameter $\hat{\boldsymbol{h}} \xleftarrow{\$} \mathbb{Z}_N^n$.

- $\mathsf{SFEncrypt}(Y, M, \mathsf{PK}, \hat{g}_1, \hat{\boldsymbol{h}})$: Upon inputs $Y, M, \mathsf{PK}, \hat{g}_1$ and $\hat{\boldsymbol{h}}$, first run $(\boldsymbol{c}; w_2) \leftarrow \mathsf{Enc2}(Y)$. Pick $\boldsymbol{s} = (s_0, s_1, \ldots, s_{w_2}), \hat{\boldsymbol{s}} \xleftarrow{\$} \mathbb{Z}_N^{w_2+1}$ Output the ciphertext as $\mathsf{CT} = (\boldsymbol{C}, C_0)$:
$$\boldsymbol{C} = g_1^{\boldsymbol{c}(\boldsymbol{s},\boldsymbol{h})} \hat{g}_1^{\boldsymbol{c}(\hat{\boldsymbol{s}},\hat{\boldsymbol{h}})} \in \mathbb{G}^{w_1}, \qquad C_0 = (e(g_1, g_2)^\alpha)^{s_0} M \in \mathbb{G}_T.$$

- $\mathsf{SFKeyGen}(X, \mathsf{MSK}, \mathsf{PK}, \hat{g}_2, \mathsf{type}, \hat{\alpha}, \hat{\boldsymbol{h}})$: Upon inputs $X, \mathsf{MSK}, \mathsf{PK}, \hat{g}_2$ and $\mathsf{type} \in \{1,2,3\}$, $\hat{\alpha} \in \mathbb{Z}_N$, first run $(\boldsymbol{k}; m_2) \leftarrow \mathsf{Enc1}(X)$. Pick $\boldsymbol{r}, \hat{\boldsymbol{r}} \xleftarrow{\$} \mathbb{Z}_N^{m_2}, \boldsymbol{R}_3 \xleftarrow{\$} \mathbb{G}_{2,p_3}^{m_1}$. Output the secret key SK:

$$\mathsf{SK} = \begin{cases} g_2^{\boldsymbol{k}(\alpha,\boldsymbol{r},\boldsymbol{h})} \cdot \hat{g}_2^{\boldsymbol{k}(0,\hat{\boldsymbol{r}},\hat{\boldsymbol{h}})} \cdot \boldsymbol{R}_3 & \text{if } \mathsf{type} = 1 \\ g_2^{\boldsymbol{k}(\alpha,\boldsymbol{r},\boldsymbol{h})} \cdot \hat{g}_2^{\boldsymbol{k}(\hat{\alpha},\hat{\boldsymbol{r}},\hat{\boldsymbol{h}})} \cdot \boldsymbol{R}_3 & \text{if } \mathsf{type} = 2 \\ g_2^{\boldsymbol{k}(\alpha,\boldsymbol{r},\boldsymbol{h})} \cdot \hat{g}_2^{\boldsymbol{k}(\hat{\alpha},\boldsymbol{0},\boldsymbol{0})} \cdot \boldsymbol{R}_3 & \text{if } \mathsf{type} = 3 \end{cases}$$

  Note that in computing type-1 (resp., type-3) semi-functional keys, $\hat{\alpha}$ (resp., $\hat{\boldsymbol{h}}$) is not needed.

D Proof for Correctness for Our Scheme

In this section, we prove the correctness of decryption of our generic construction in §4.2.

Claim 15. *If $R(X,Y) = 1$ then*

$$\boldsymbol{\alpha}^\top \boldsymbol{B} \left(\begin{smallmatrix} \boldsymbol{s}_0 \\ 0 \end{smallmatrix} \right) = \sum_{i \in [1,m_1], i' \in [1,w_1]} E_{i,i'} \cdot (\boldsymbol{k}_{\boldsymbol{Z}}(\boldsymbol{\alpha}, \boldsymbol{R}, \mathbb{H})[i])^\top \cdot \boldsymbol{c}_{\boldsymbol{B}}(\boldsymbol{S}, \mathbb{H})[i'].$$

Proof. To prove this, we first see that due to the correctness of the pair encoding, $R(X,Y) = 1$ implies

$$\alpha s_0 = \sum_{i \in [1,m_1], i' \in [1,w_1]} E_{i,i'} \cdot k(\alpha, \boldsymbol{r}, \boldsymbol{h})[i] \cdot c(\boldsymbol{s}, \boldsymbol{h})[i']. \tag{28}$$

Each term in the sum can be written as

$$\begin{aligned} & E_{i,i'} \cdot k(\alpha, \boldsymbol{r}, \boldsymbol{h})[i] \cdot c(\boldsymbol{s}, \boldsymbol{h})[i'] = \\ & E_{i,i'} \sum_{j' \in [0,w_2]} b_i a_{i',j'} \alpha s_{j'} + E_{i,i'} \sum_{\substack{j' \in [0,w_2] \\ k' \in [1,n]}} b_i a_{i',j',k'} \alpha h_{k'} s_{j'} \\ & \quad + E_{i,i'} \sum_{\substack{j \in [1,m_2] \\ j' \in [0,w_2]}} b_{i,j} a_{i',j'} r_j s_{j'} \\ & \quad + \underbrace{E_{i,i'} \sum_{\substack{j \in [1,m_2] \\ j' \in [0,w_2] \\ k' \in [1,n]}} b_{i,j} a_{i',j',k'} r_j h_{k'} s_{j'} + E_{i,i'} \sum_{\substack{j \in [1,m_2] \\ k \in [1,n] \\ j' \in [0,w_2]}} b_{i,j,k} a_{i',j'} r_j h_k s_{j'}}_{E_{i,i'} \sum_{\substack{j \in [1,m_2] \\ k \in [1,n] \\ j' \in [0,w_2]}} (b_{i,j} a_{i',j',k} + b_{i,j,k} a_{i',j'}) r_j h_k s_{j'}} \\ & \quad + \underbrace{E_{i,i'} \sum_{\substack{j \in [1,m_2] \\ k \in [1,n] \\ j' \in [0,w_2] \\ k' \in [1,n]}} b_{i,j,k} a_{i',j',k'} r_j h_k h_{k'} s_{j'},}_{= 0} \end{aligned} \tag{29}$$

where we note that the last term, denoted L, is 0 intuitively due to the *regularity* of encoding (the first rule, in Definition 1), which disallows the multiplication of $(r_j h_k)(h_{k'} s_{j'})$. More precisely, we fix i, i'. Then, for j, k which $b_{i,j,k} = 0$, L is trivially 0. Similarly, for j', k' which $a_{i',j',k'} = 0$, L is also trivially 0. For the remaining case of j, k, j', k' where $b_{i,j,k} \neq 0$ and $a_{i',j',k'} \neq 0$, we have that $E_{i,i'} = 0$ exactly due to the first rule of regularity.

This whole term of (29) can be viewed as a polynomial, denoted $p^{i,i'}(\boldsymbol{v})$, with variables in $\boldsymbol{v}$ consisting of

$$\{\alpha s_{j'}\}_{j' \in [0,w_2]}, \qquad \{\alpha h_{k'} s_{j'}\}_{\substack{j' \in [0,w_2] \\ k' \in [1,n]}}, \qquad \{r_j s_{j'}\}_{\substack{j \in [1,m_2] \\ j' \in [0,w_2]}}, \qquad \{r_j h_k s_{j'}\}_{\substack{j \in [1,m_2] \\ k \in [1,n] \\ j' \in [0,w_2]}}. \tag{30}$$

We then substituting variables $\boldsymbol{v}$ with $\boldsymbol{W}$ consisting of

$$\begin{gathered} \{\boldsymbol{\alpha}^\top \boldsymbol{B} \left(\begin{smallmatrix} \boldsymbol{s}_{j'} \\ 0 \end{smallmatrix}\right)\}_{j' \in [0,w_2]}, \qquad \{\boldsymbol{\alpha}^\top \boldsymbol{H}_{k'} \boldsymbol{B} \left(\begin{smallmatrix} \boldsymbol{s}_{j'} \\ 0 \end{smallmatrix}\right)\}_{\substack{j' \in [0,w_2] \\ k' \in [1,n]}}, \\ \{\left(\begin{smallmatrix} \boldsymbol{r}_j^\top & 0 \end{smallmatrix}\right) \boldsymbol{Z}^\top \boldsymbol{B} \left(\begin{smallmatrix} \boldsymbol{s}_{j'} \\ 0 \end{smallmatrix}\right)\}_{\substack{j \in [1,m_2] \\ j' \in [0,w_2]}}, \quad \{\left(\begin{smallmatrix} \boldsymbol{r}_j^\top & 0 \end{smallmatrix}\right) \boldsymbol{Z}^\top \boldsymbol{H}_k \boldsymbol{B} \left(\begin{smallmatrix} \boldsymbol{s}_{j'} \\ 0 \end{smallmatrix}\right)\}_{\substack{j \in [1,m_2] \\ k \in [1,n] \\ j' \in [0,w_2]}}. \end{gathered} \tag{31}$$

in the respective manner (*i.e.,* elements in (31) are in the same order as those in (30)) and obtain $p^{i,i'}(\boldsymbol{W})$. Since Eq.(28) holds symbolically, we thus have

$$\boldsymbol{\alpha}^\top \boldsymbol{B} \left(\begin{smallmatrix} \boldsymbol{s}_0 \\ 0 \end{smallmatrix}\right) = \sum_{i \in [1,m_1], i' \in [1,w_1]} p^{i,i'}(\boldsymbol{W}).$$

Therefore, it remains to prove that

$$p^{i,i'}(\boldsymbol{W}) = E_{i,i'} \cdot (\boldsymbol{k}_{\boldsymbol{Z}}(\boldsymbol{\alpha}, \boldsymbol{R}, \mathbb{H})[i])^\top \cdot \boldsymbol{c}_{\boldsymbol{B}}(\boldsymbol{S}, \mathbb{H})[i']. \tag{32}$$

To prove this, we write the term on the right-hand side of (32) using the definitions of $\boldsymbol{k}_{\boldsymbol{Z}}, \boldsymbol{c}_{\boldsymbol{B}}$ from (9) and (11) as

$$\begin{aligned} E_{i,i'} \Bigg(b_i \boldsymbol{\alpha}^\top + \Bigg(\sum_{j \in [1,m_2]} b_{i,j} \left(\begin{smallmatrix} \boldsymbol{r}_j^\top & 0 \end{smallmatrix}\right) \boldsymbol{Z}^\top \Bigg) + \Bigg(\sum_{\substack{j \in [1,m_2] \\ k \in [1,n]}} b_{i,j,k} \left(\begin{smallmatrix} \boldsymbol{r}_j^\top & 0 \end{smallmatrix}\right) \boldsymbol{Z}^\top \boldsymbol{H}_k \Bigg) \Bigg) \cdot \\ \Bigg(\Bigg(\sum_{j' \in [0,w_2]} a_{i',j'} \boldsymbol{B} \left(\begin{smallmatrix} \boldsymbol{s}_{j'} \\ 0 \end{smallmatrix}\right) \Bigg) + \Bigg(\sum_{\substack{j' \in [0,w_2] \\ k' \in [1,n]}} a_{i',j',k'} \boldsymbol{H}_{k'} \boldsymbol{B} \left(\begin{smallmatrix} \boldsymbol{s}_{j'} \\ 0 \end{smallmatrix}\right) \Bigg) \Bigg). \end{aligned}$$

We then multiply it out and check the coefficient of each variable in $\boldsymbol{W}$ whether it equals to that in $p^{i,i'}(\boldsymbol{W})$, of which the coefficients are shown in (29). By inspection, we can see that this holds. One particular non-trivial point is that, the coefficient of $\left(\begin{smallmatrix} \boldsymbol{r}_j^\top & 0 \end{smallmatrix}\right) \boldsymbol{Z}^\top \boldsymbol{H}_k \boldsymbol{B} \left(\begin{smallmatrix} \boldsymbol{s}_{j'} \\ 0 \end{smallmatrix}\right)$ is $b_{i,j} a_{i',j',k} + b_{i,j,k} a_{i',j'}$ (and hence is the same as in $p^{i,i'}(\boldsymbol{W})$ as shown in Eq.(29)), exactly due to the associativity:

$$\Big(\left(\begin{smallmatrix} \boldsymbol{r}_j^\top & 0 \end{smallmatrix}\right) \boldsymbol{Z}^\top \Big) \Big(\boldsymbol{H}_k \boldsymbol{B} \left(\begin{smallmatrix} \boldsymbol{s}_{j'} \\ 0 \end{smallmatrix}\right) \Big) = \Big(\left(\begin{smallmatrix} \boldsymbol{r}_j^\top & 0 \end{smallmatrix}\right) \boldsymbol{Z}^\top \boldsymbol{H}_k \Big) \Big(\boldsymbol{B} \left(\begin{smallmatrix} \boldsymbol{s}_{j'} \\ 0 \end{smallmatrix}\right) \Big).$$

This concludes the proof. $\square$

E Variants of Security for Our Construction

In this section, we describe two more theorems for our framework.

Theorem 16. *Suppose that a pair encoding scheme* P *for predicate* R *is* $(1,1)$*-selectively and* $(1,1)$*-co-selectively master-key hiding in* $\mathcal{G}$*, and the Matrix-DH Assumption holds in* $\mathcal{G}$*. Then the construction* $\mathsf{ABE}(\mathsf{P})$ *in* $\mathcal{G}$ *of ABE for predicate* R *is fully secure. More precisely, for any PPT adversary* $\mathcal{A}$*, there exist PPT algorithms* $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3$*, whose running times are the same as* $\mathcal{A}$ *plus some polynomial times, such that for any* λ*,*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ABE}}(\lambda) \le (2q_{\rm all}+1)\mathsf{Adv}_{\mathcal{B}_1}^{\mathcal{D}_d\text{-}\mathsf{MatDH}}(\lambda) + q_1\mathsf{Adv}_{\mathcal{B}_2}^{(1,1)\text{-}\mathsf{CMH}}(\lambda) + q_2\mathsf{Adv}_{\mathcal{B}_3}^{(1,1)\text{-}\mathsf{SMH}}(\lambda),$$

where q_1, q_2 *is the number of queries in phase 1,2, resp., and* $q_{\rm all} = q_1 + q_2$*.*

Proof. This follows the proof of our main theorem (Theorem 3). The only difference is that instead of switching all *post-challenge* keys *all at once* for the three games (normal to semi-functional type 1, to type 2, and to type 3), we switch each post-challenge key *one key per one game*, in just the same way as for each pre-challenge key (and as in the traditional dual system encryption proofs). This results in the cost q_2 for the reduction to the SMH security and the additional cost $2q_2 - 2$ for the reduction to $\mathcal{D}_d$-MatDH. □

Corollary 17. *Suppose that a pair encoding scheme* P *for predicate* R *is perfectly master-key hiding, and the Matrix-DH Assumption holds in* $\mathcal{G}$*. Then the construction* $\mathsf{ABE}(\mathsf{P})$ *in* $\mathcal{G}$ *of ABE for predicate* R *is fully secure. More precisely, for any PPT adversary* $\mathcal{A}$*, there exist PPT algorithms* $\mathcal{B}$*, whose running time is the same as* $\mathcal{A}$ *plus some polynomial time, such that for any* λ*,*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ABE}}(\lambda) \le (2q_{\rm all}+1)\mathsf{Adv}_{\mathcal{B}}^{\mathcal{D}_d\text{-}\mathsf{MatDH}}(\lambda).$$

where $q_{\rm all} = q_1 + q_2$ *denotes the number of all queries.*

Proof. Since PMH trivially implies both $(1,1)$-CMH and $(1,1)$-SMH with zero advantage, we obtain the above corollary. □

F Instantiation Example: KP-ABE for Regular Languages

To illustrate how an encoding scheme will satisfy regularity (Definition 1), we show an instantiation example. We pick the encoding scheme for ABE for regular languages proposed in [1]. We show its regularity below. Thus, our generic construction gives a fully-secure ABE scheme for regular languages in prime-order groups. The underlying assumptions besides the Matrix-DH assumption are exactly those that are defined in [1] (the $\mathsf{EDHE1}$ and $\mathsf{EDHE2}$ assumptions) albeit *defined over prime-order groups.* The original assumptions are defined over prime-order subgroups inside composite-order groups; however, the hardness of factorization was not inherently used, so we can neglect other subgroups. We remark that this is the first such scheme. For self-containment, we first give the definition for the predicate, where we mostly copy texts from [1] here.

Predicate Definition of ABE for Regular Languages. In ABE for regular languages, we have a key associated to the description of a deterministic finite automata (DFA) M, while a ciphertext is associated to a string w, and $R(M,w)=1$ if the automata M accepts the string w. A DFA M is a 5-tuple $(Q,\Lambda,\mathcal{T},q_0,F)$ in which Q is the set of states $Q=\{q_0,q_1,\ldots,q_{n-1}\}$, Λ is the alphabet set, $\mathcal{T}$ is the set of transitions, in which each transition is of the form $(q_x,q_y,\sigma)\in Q\times Q\times\Lambda$, $q_0\in Q$ is the start state, and $F\subseteq Q$ is the set of accepted states. We say that M accepts a string $w=(w_1,w_2,\ldots,w_\ell)\in\Lambda^*$ if there exists a sequence of states $\rho_0,\rho_1,\ldots,\rho_\ell\in Q$ such that $\rho_0=q_0$, for $i=1$ to ℓ we have $(\rho_{i-1},\rho_i,w_i)\in\mathcal{T}$, and $\rho_\ell\in F$. This primitive is important since it has a unique unbounded feature that one key for machine M can operate on input string w of *arbitrary* sizes. Such an ABE system was proposed by Waters [50] in the selective security model. The fully secure scheme was achieved via the framework of [1] in composite-order groups. We recap the encoding scheme in [1] as follows, where we simplify some notations of variables.

Pair Encoding Scheme 1: KP-ABE for Regular Languages of [1]

Param $\to 8$. Denote $\boldsymbol{h}=(h_0,h_1,h_2,h_3,h_4,h_5,h_6,h_7)$.

For any DFA $M=(Q,\mathbb{Z}_p,\mathcal{T},q_0,q_{n-1})$, where $n=|Q|$,

let $m=|\mathcal{T}|$, and parse $\mathcal{T}=\{(q_{x_t},q_{y_t},\sigma_t)|t\in[1,m]\}$.

$\mathsf{Enc1}(M)\to \boldsymbol{k}(\alpha,\boldsymbol{r},\boldsymbol{h})=\big(k_1,k_2,k_3,k_4,k_5,\{k_{6,t},k_{7,t},k_{8,t}\}_{t\in[1,m]}\big)$:

$$\left\{\begin{array}{lll} k_1=\alpha+rh_5+uh_7, & k_2=u, & k_3=r,\\ k_4=r_0, & k_5=-u_0+r_0h_0, & k_{6,t}=r_t,\\ k_{7,t}=u_{x_t}+r_t(h_1+h_2\sigma_t), & k_{8,t}=-u_{y_t}+r_t(h_3+h_4\sigma_t) & \end{array}\right\}$$

where $u_{n-1}:=h_6r$ and $\boldsymbol{r}=(r,u,r_0,r_1,\ldots,r_m,\{u_x\}_{q_x\in Q\smallsetminus\{q_{n-1}\}})$.

For $w\in(\mathbb{Z}_p)^*$, let $\ell=|w|$, and parse $w=(w_1,\ldots,w_\ell)$.

$\mathsf{Enc2}(w)\to \boldsymbol{c}(\boldsymbol{s},\boldsymbol{h})=\big(c_1,c_2,c_3,c_4,\{c_{5,i}\}_{i\in[0,\ell]},\{c_{6,i}\}_{i\in[1,\ell]}\big)$:

$$\left\{\begin{array}{lll} c_1=s_0, & c_2=s_0h_7, & c_3=-s_0h_5+s_{\ell+1}h_6,\\ c_4=s_1h_0, & c_{5,i}=s_{i+1}, & c_{6,i}=s_i(h_1+h_2w_i)+s_{i+1}(h_3+h_4w_i) \end{array}\right\}$$

where $\boldsymbol{s}=(s_0,s_1,\ldots,s_{\ell+1})$.

Correctness. The correctness can be shown by providing linear combination of $k_\iota c_j$ which summed up to αs_0. When $R(M,w)=1$, we have that there is a sequence of states $\rho_0,\rho_1,\ldots,\rho_\ell\in Q$ such that $\rho_0=q_0$, for $i=1$ to ℓ we have $(\rho_{i-1},\rho_i,w_i)\in\mathcal{T}$, and $\rho_\ell\in F$. Let $(q_{x_{t_i}},q_{y_{t_i}},\sigma_{t_i})=(\rho_{i-1},\rho_i,w_i)$. Therefore, we have the following bilinear combination:

$$k_1c_1-k_2c_2+k_3c_3-k_4c_4+k_5c_{5,0}+\sum_{i\in[1,\ell]}(-k_{6,t_i}c_{6,i}+k_{7,t_i}c_{5,i-1}+k_{8,t_i}c_{5,i})=\alpha s. \qquad (33)$$

Claim 18. *The above encoding scheme is regular pair encoding.*

Proof. We inspect each requirement from Definition 1 as follows.

1. In the linear combination of $k_\iota c_j$ terms when pairing as shown in Eq.(33), the two terms in each pair does not simultaneously contain elements from $\boldsymbol{h}$. For example, in the product k_1c_1, the polynomial k_1 contains h_5,h_7, while c_1 has no h_i element. On the other hand, in the product k_3c_3, the polynomial k_3 has no h_i element, while c_3 contains h_5,h_6.

2. In the encoding $\boldsymbol{k}(\alpha, \boldsymbol{r}, \boldsymbol{h})$, all the monomials that have randomness in $\boldsymbol{r}$ multiplied with elements from $\boldsymbol{h}$ are $rh_5, uh_7, r_0h_0, r_th_1, r_th_2\sigma_t, r_th_3, r_th_4\sigma_t, rh_6$. We check that it is indeed the case that all the corresponding randomness terms, r, u, r_0, r_t, are given out (as singleton monomial $k_3, k_2, k_4, k_{6,t}$ respectively). On the other hand, since the other randomness elements, u_x's, are not multiplied with any h_i, they are not needed to be given out.

3. In the encoding $\boldsymbol{c}(\boldsymbol{s}, \boldsymbol{h})$, all the randomness $s_0, \ldots, s_{\ell+1}$ are multiplied with some h_i term, and we can see that all monomial s_j's exist in the encoding.

4. The monomial s_0 appear in the encoding $\boldsymbol{c}(\boldsymbol{s}, \boldsymbol{h})$ (as c_1).

This conclude the proof. $\square$

Contents

1 Introduction **1**
- 1.1 Our Contributions . . . 1
- 1.2 Difficulties and Our Approaches . . . 2
- 1.3 Concurrent and Independent Work . . . 7
- 1.4 Related Work . . . 7

2 Preliminaries **8**
- 2.1 Definitions of Attribute Based Encryption . . . 8
- 2.2 Bilinear Groups and Assumptions . . . 9

3 Definition of Pair Encoding **10**
- 3.1 Regular Pair Encoding . . . 11
- 3.2 Security Definitions for Pair Encodings . . . 12

4 Our Framework in Prime-Order Groups **12**
- 4.1 Intuition for Translation to Prime-Order Groups . . . 12
- 4.2 Our Generic Construction for Fully Secure ABE . . . 15

5 Security Theorems and Proofs **16**
- 5.1 Normal to Semi-functional Ciphertext . . . 19
- 5.2 Normal to Type-1 Semi-functional Key in Phase 1 . . . 20
- 5.3 Applying the Parameter-Hiding Lemma . . . 22
- 5.4 Type-1 to Type-2 Semi-functional Key in Phase 1 . . . 25
- 5.5 Type-2 to Type-3 Semi-functional Key in Phase 1 . . . 28
- 5.6 Normal to Type-1 Semi-functional Keys in Phase 2 . . . 28
- 5.7 Type-1 to Type-2 Semi-functional Key in Phase 2 . . . 28
- 5.8 Type-2 to Type-3 Semi-functional Key in Phase 2 . . . 30
- 5.9 Final Game . . . 31

6 New Instantiations from Our Framework **31**
- 6.1 The First Prime-order Schemes for Existing Predicates . . . 32
- 6.2 New Predicates . . . 34

References **37**

A Security Definition for ABE **40**

B Recap the Security Definitions for Pair Encodings **41**

C Recap the Composite-order Construction of [1] **43**

D Proof for Correctness for Our Scheme **44**

E Variants of Security for Our Construction **46**

F Instantiation Example: KP-ABE for Regular Languages **46**