Dual System Framework in Multilinear Settings and Applications to Fully Secure (Compact) ABE for Unbounded-Size Circuits*

Nuttapong Attrapadung

National Institute of Advanced Industrial Science and Technology (AIST),
Tokyo, Japan.
`n.attrapadung@aist.go.jp`

Abstract. We propose a new generic framework for constructing fully secure attribute based encryption (ABE) in *multilinear* settings. It is applicable in a generic manner to *any predicates*. Previous generic frameworks of this kind are given only in *bilinear* group settings, where applicable predicate classes are limited. Our framework provides an abstraction of dual system paradigms over composite-order graded multilinear encoding schemes in a black-box manner.

As applications, we propose new fully secure ABE systems for general predicates, namely, ABE for circuits. We obtain two schemes for each of key-policy (KP) and ciphertext-policy (CP) variants of ABE. All of our four fully secure schemes can deal with *unbounded-size* circuits, while enjoy *succinctness*, meaning that the key and ciphertext sizes are (less than or) proportional to corresponding circuit sizes. In the CP-ABE case, no scheme ever achieves such properties, even when considering selectively secure systems. Furthermore, our second KP-ABE achieves *constant-size ciphertexts*, whereas our second CP-ABE achieves *constant-size keys*. Previous ABE systems for circuits are either selectively secure (Gorbunov *et al.* STOC'13, Garg *et al.* Crypto'13, and subsequent works), or semi-adaptively secure (Brakerski and Vaikuntanathan Crypto'16), or fully-secure but not succinct and restricted to bounded-size circuits (Garg *et al.* ePrint 2014/622, and Garg *et al.* TCC'16-A).

Keywords. Attribute-based encryption, full security, multilinear maps, dual system, pair encodings, circuits.

1 Introduction

Attribute-based encryption (ABE), introduced by Sahai and Waters [44], is a new paradigm that generalizes traditional public key encryption. Instead of encrypting to a target recipient, a sender can specify in a more general way about who should be able to view the message. In ABE for predicate $R : \mathbb{X} \times \mathbb{Y} \to \{0, 1\}$, a ciphertext encrypting message M is associated with a ciphertext attribute, say,

*This paper subsumes [4]. The full version is available at [6].

$Y \in \mathbb{Y}$, while a secret key, issued by an authority, is associated with a key attribute, say, $X \in \mathbb{X}$, and the decryption will succeed if and only if $R(X, Y) = 1$. From an application point of view, it is instructive to consider one kind of attributes as *policies*, which are Boolean functions, and the other kind as inputs to functions. In this sense, there are two variants of ABE. In Key-Policy (KP) type [33], $\mathbb{X}$ is a set of Boolean functions, while $\mathbb{Y}$ is a set of inputs to functions, and we define $R(f, x) = f(x)$. In Ciphertext-Policy (CP) type [10], the roles of $\mathbb{X}$ and $\mathbb{Y}$ are swapped (that is, ciphertexts are associated with policies).

A central theme to ABE has been to expand the class of allowable boolean functions. Until recently, there were only ABE for simple classes such as boolean formulae [33, 10, 37, 40] and inner product predicate [34, 7, 41]. Only recently, ABE systems that allow any unbounded polynomial-size circuits (but bounded-depth) were proposed independently by Garg *et al.* (GGHSW) [24] and Gorbunov *et al.* (GVW) [31]. The former is based on multi-linear maps (more precisely, graded encoding systems) [23, 20], while the latter is based on the Learning-With-Error assumption. They proposed key-policy variants, and by using universal circuits, ciphertext-policy systems can also be obtained albeit for only bounded-size circuits. Subsequently, Garg *et al.* [28] proposed ABE for circuits based on witness encryption. Boneh *et al.* [13] (BGG+) proposed KP-ABE for circuits with short keys or short ciphertexts.

Full vs Selective Security. The standard security for ABE is *adaptive security*, or often called *full security*. However, previous ABE systems for circuits [24, 31, 28, 13] were proved only in a weaker model called *selective security*. Such a notion requires the adversary to announce a target ciphertext attribute $Y^\star$ upfront before seeing the public key, after then, he can ask for secret keys of X such that $R(X, Y^\star) = 0$. Contrastingly, full security allows the adversary to adaptively ask for secret keys and choose a target in any order.

Complexity Leveraging. There is a trivial method to generically bootstrap selective security to full security called *complexity leveraging* [12]. In this approach, the security reduction would incur a loss factor $|\mathbb{Y}|$ (stemmed from the probability of guessing $Y^\star$ from the ciphertext attribute domain $\mathbb{Y}$). In KP-ABE for circuits that allows inputs of length n, we have $|\mathbb{Y}| = 2^n$, hence the loss factor is *exponential*. Although this loss can be compensated by increasing the security parameter by n, this is undesirable by two reasons. First, as a direct consequence, it makes the resulting scheme inefficient. Second, and perhaps more importantly, the resulting security reduction becomes "unfalsifiable" in the sense that even an attacker with probability 1 in attacking the scheme cannot be used to solve the underlying hard problem in sub-exponential time [16].

Fully Secure CP-ABE for Circuits. The situation for CP-ABE for unbounded-size circuits is even more devastating since the loss factor can be as large as *double exponential*, as the number of all Boolean functions with n inputs is 2^{2^n}. In this case, complexity leveraging cannot be used since we cannot compensate by increasing the security parameter by 2^n, which is exponential. Moreover, even when we restrict to bounded-depth circuits, the loss factor can still be super-exponential or large exponential functions (in parameters such as depth ℓ).[1,2]

Problem Statement. To this end, we consider the following problem:

> Problem 1: *Is it possible to construct fully secure KP-ABE and CP-ABE for circuits with polynomial reductions (in all parameters) to some non-interactive assumptions?*

Unbounded-size Circuits and Succinctness. It is desirable for new fully secure schemes to preserve functionalities and efficiency from previous selectively secure systems. For functionalities, the goal is ABE that allows *unbounded-size* circuits. For efficiency, we require *succinctness*: the size of a key (resp., a ciphertext) for circuit f is less than or proportional to the circuit size in KP-ABE (resp., CP-ABE). In KP-ABE case, we refine our question to:

> Problem 1$'$: *Is it possible to construct fully secure KP-ABE that allows unbounded circuits (possibly bounded-depth) and/or admits succinctness (again, with polynomial reductions to non-interactive assumptions)?*

In CP-ABE case, however, all the available schemes [24, 31] are for bounded-size circuits and do not admit succinctness, not to mention that they are selectively secure. This is due to the use of universal circuits [46]. We thus ask:

> Problem 2: *Is it possible to construct (even selectively secure) CP-ABE that allows unbounded-size circuits and/or admits succinctness?*

Short Ciphertexts and Short Keys. Finally, we focus on optimizing the size of a ciphertext (resp. a key) for an input string x in KP-ABE (resp. CP-ABE). We say that a scheme admits *constant-size* ciphertext (resp., key) if the size besides the description of x is constant in term of the length n of x. We ask:

> Problem 3: *Is it possible to construct fully-secure KP-ABE with constant-size ciphertexts, fully-secure CP-ABE with constant-size keys (again, for unbounded-size circuits and with polynomial reductions)?*

1.1 Our Contributions on ABE Instantiations

Our contribution is twofold: a generic framework and instantiations. We first introduce our results regarding instantiations, which are summarized as:

Theorem 1. *(*Instantiations*, informally). There exist fully secure KP-ABE, CP-ABE for unbounded-size bounded-depth circuits with polynomial reductions to some non-interactive assumptions on composite-order 3ℓ-multilinear maps, where ℓ is the bounded depth. Constructively, we obtain 4 schemes:*

[1]We do not elaborate the exact number as it is quite tricky to count the number of all Boolean functions that can be computed by unbounded-size bounded-depth circuits.

[2]When we further restrict to bounded-size circuits, the loss factor is $2^{\mathsf{poly}(g_{max})}$, where g_{max} is the maximum circuit size. This is exactly the reduction loss for all the available fully secure CP-ABE via complexity leveraging (see Table 2).

1. *fully secure KP-ABE admitting succinctness.*
2. *fully secure KP-ABE admitting succinctness and constant-size ciphertexts.*
3. *fully secure CP-ABE admitting succinctness.*
4. *fully secure CP-ABE admitting succinctness and constant-size keys.*

Our schemes affirmatively answer Problem 1, constructing fully secure ABE with polynomial reductions. (See below for independent works [26, 27] that also solve Problem 1.) Moreover, both of our KP-ABE schemes and both of our CP-ABE schemes are the first to affirmatively answer Problem $1'$ and Problem 2, respectively. Finally, our second KP-ABE and our second CP-ABE provide the first positive answers to Problem 3.

We provide comparisons to the other schemes in the literature in Table 1,2 (and with sizes provided in more details in Table 3,4).

Comparisons. From Table 1,2, we can see that our first and second (fully secure) KP-ABE schemes are comparable to the (selectively secure) KP-ABE of GGHSW [24] and BGG+2 [13] in both functionality (unbounded-size circuits) and efficiency (succinctness, constant-size ciphertext). On the other hand, both of our (fully secure) CP-ABE schemes perform much better than all the previous (selectively secure) schemes in both functionality (ours are the first to allow unbounded-size circuits) and efficiency (ours are the first to be succinct).

In independent[3] works, Garg *et al.* proposed fully-secure ABE [26] (and FE [27], see §1.4) for circuits, thus also answer Problem 1; however, their schemes are for bounded-size circuits and do not admit succinctness, due to their essential use of "fixed-once and for all" universal circuits. Moreover, as shown in Table 1,2, our schemes require much less multi-linearity and admit tighter reductions.

On Assumptions. To prove security of our schemes, we introduce some new non-interactive assumptions (thus, they are falsifiable [39]). They somewhat extend the Multi-linear Decisional Diffie-Hellman Assumption (MDDH) [15, 23, 20]. These assumptions are of "parameterized" type (or often called "q-type"), where the size of assumption grows depending on some parameters. Although they are not standard, we prove that they hold in the generic model. To compare these assumptions quantitatively, in Table 1,2, we represent their complexities in terms of their assumption sizes. Intuitively, but not necessarily, the larger the size, the stronger the assumption is. We note that, in our schemes, the parameters for the assumptions depend only on the depth ℓ, width m, or input length n, of a circuit in one query (and not on the number of key queries). The reduction cost in our schemes is $O(q_1)$ where q_1 is the number of pre-challenge key queries.

Implementations. Unfortunately, currently there is no known secure multilinear map (see more later in §1.4). Hence, at present, our results can be considered as only theoretical black-box reductions from fully secure succinct ABE for unbounded circuits to (composite-order) multi-linear maps. Nevertheless, due to the nature of black-box usages, any future secure candidates can be used.

[3]Our preliminary version [4] has been made available shortly after [26, 27].

Table 1: KP-ABE for Circuits.

Schemes	$\lvert$Cipher$\rvert$†	$\lvert$Key$\rvert$†	Unbound $\lvert$circuit$\rvert$?	Tool	Security‡	Reduction	Assumptions§
GVW [31]	$O(n)$	$O(g)$	yes	LWE	full selective	$O(2^n)$ $O(1)$	$2^{O(n^\epsilon)}$-hardness of LWE $2^{O(n^\epsilon)}$-hardness of LWE
GGHSW [24]	$O(n')$	$O(g)$	yes	ℓ-multmap	full selective	$O(2^n)$ $O(1)$	param-ass. size $O(\ell)$ param-ass. size $O(\ell)$
BGG+1 [13]	$O(n)$	$O(1)$	yes	LWE	full selective	$O(2^n)$ $O(1)$	$2^{O(n^\epsilon)}$-hardness of LWE $2^{O(n^\epsilon)}$-hardness of LWE
BGG+2 [13]	$O(1)$	$O(n^2+g)$	yes	ℓ-multmap	full selective	$O(2^n)$ $O(1)$	param-ass. size $O(\ell+n)$ param-ass. size $O(\ell+n)$
GGHZ1,2 [26, 27]	$\mathsf{poly}(g_{max})$	$\mathsf{poly}(g_{max})$	no	$\mathsf{poly}(g_{max})$-multmap	full	$\mathsf{poly}(g_{max}, q_{\rm all})$	param-ass. size $\mathsf{poly}(g_{max})$
BV [11]**	$O(n)$	$O(1)$	yes	LWE	semi-adapt	$O(1)$	subexp-hardness of LWE
Our KP1 (§4)	$O(n')$	$O(g)$	yes	3ℓ-multmap	full semi-adapt	$O(q_1)$ $O(1)$	param-ass. size $O(\ell m^2)$ param-ass. size $O(\ell)$
Our KP2 (§5)	$O(1)$	$O(n^2+g)$	yes	3ℓ-multmap	full semi-adapt	$O(q_1)$ $O(1)$	param-ass. size $O(\ell m^2+n^2)$ param-ass. size $O(\ell+n)$

Table 2: CP-ABE for Circuits.

Schemes	$\lvert$Cipher$\rvert$†	$\lvert$Key$\rvert$†	Unbound $\lvert$circuit$\rvert$?	Tool	Security‡	Reduction	Assumptions§
GVW [31]¶	$\mathsf{poly}(g_{max})$	$\mathsf{poly}(g_{max})$	no	LWE	full selective	$2^{\mathsf{poly}(g_{max})}$ $O(1)$	$2^{O(n^\epsilon)}$-hardness of LWE $2^{O(n^\epsilon)}$-hardness of LWE
GGHSW [24]¶	$\mathsf{poly}(g_{max})$	$\mathsf{poly}(g_{max})$	no	$\mathsf{poly}(g_{max})$-multmap	full selective	$2^{\mathsf{poly}(g_{max})}$ $O(1)$	param-ass. size $\mathsf{poly}(g_{max})$ param-ass. size $\mathsf{poly}(g_{max})$
BGG+1 [13]¶	$\mathsf{poly}(g_{max})$	$O(1)$	no	LWE	full selective	$2^{\mathsf{poly}(g_{max})}$ $O(1)$	$2^{O(n^\epsilon)}$-hardness of LWE $2^{O(n^\epsilon)}$-hardness of LWE
BGG+2 [13]¶	$O(1)$	$\mathsf{poly}(n^2, g_{max})$	no	$\mathsf{poly}(g_{max})$-multmap	full selective	$2^{\mathsf{poly}(g_{max})}$ $O(1)$	param-ass. size $\mathsf{poly}(g_{max})$ param-ass. size $\mathsf{poly}(g_{max})$
GGHZ1,2 [26, 27]	$\mathsf{poly}(g_{max})$	$\mathsf{poly}(g_{max})$	no	$\mathsf{poly}(g_{max})$-multmap	full	$\mathsf{poly}(g_{max}, q_{\rm all})$	param-ass. size $\mathsf{poly}(g_{max})$
Our CP1 (§6.2)	$O(g)$	$O(n')$	yes	3ℓ-multmap	full	$O(q_{\rm all})$	param-ass. size $O(\ell m^2)$
Our CP2 (§6.2)	$O(n^2+g)$	$O(1)$	yes	3ℓ-multmap	full	$O(q_{\rm all})$	param-ass. size $O(\ell m^2+n^2)$

* Notation for variables: n is the length of input to a circuit; $n'(\le n)$ is the number of 1's in the input bit string to circuits; g is the size of a circuit (the number of gates including input nodes); g_{max} is the maximum bound for g (if bounded); m is the width of a circuit; ℓ is the bounded depth of circuits; q_1 are the number of pre-challenge key queries; $q_{\rm all}$ is the number of all key queries. ϵ is a parameter for LWE ($0 < \epsilon < 1/2$) [31].

** Only ABE of [11] achieves unbounded input length, *i.e.,* the input string length n is not a-priori bounded.

† Sizes ($\lvert$Cipher$\rvert$, $\lvert$Key$\rvert$) are shown in the number of "unit" elements naturally defined in the respective underlying tool. Let λ be the security parameter. For multi-linear maps, one unit element is a graded encoded element; for previous (now-broken) candidates [20, 22], the size of one unit is $\mathsf{poly}(\lambda, \kappa)$ bits, for κ-multilinear maps. For LWE, intuitively, one unit element is a matrix that defines a single instance of the LWE assumption; the size for one unit is $\mathsf{poly}(\lambda, \ell^{1/\epsilon})$ bits for the GVW [31] and the BGG+1 systems [13]. The overall ciphertext size is then $\lvert\text{Cipher}\rvert\lvert\text{unit}\rvert + \lvert Y\rvert$, where $\lvert Y\rvert$ is the description size of ciphertext attribute (circuit f for CP, input string x for KP). Similarly, The overall key size is $\lvert\text{Key}\rvert\lvert\text{unit}\rvert + \lvert X\rvert$. We provide overall sizes in Table 3,4.

‡ For each scheme satisfying two levels of security, we provide respective reduction/assumptions in each line.

§ All multi-linear map based schemes in the tables use "parameterized" assumptions (param-ass.). To be able to compare quantitively, we write their complexities in terms of the assumption size. (Intuitively but not necessarily, the larger the size, the stronger the assumption is). All of these schemes use at most three assumptions, and the size in the table represents the largest one.

¶ These CP-ABE schemes were obtained by converting from KP-ABE via universal circuits. In doing so, one must fix g_{max}, *i.e.,* the resulting schemes are for bounded-size circuit. An (asymptotically) optimal universal circuit [46] has size $O(g_{max} \log g_{max})$ and depth $O(g_{max})$, hence related parameters can be given by $\mathsf{poly}(g_{max})$.

Table 3: KP-ABE for Circuits (sizes given in more details).

Schemes	|Cipher| (no. of bits)	|Key| (no. of bits)
GVW [31]	$O(n)\mathsf{poly}(\lambda, \ell^{1/\epsilon})$	$O(g)\mathsf{poly}(\lambda, \ell^{1/\epsilon})$
GGHSW [24]	$O(n')\mathsf{poly}(\lambda, \ell) + n$	$O(g)\mathsf{poly}(\lambda, \ell)$
BGG+1,BV [13, 11]	$O(n)\mathsf{poly}(\lambda, \ell^{1/\epsilon})$	$\mathsf{poly}(\lambda, \ell^{1/\epsilon}) + \|f\|$
BGG+2 [13]	$\mathsf{poly}(\lambda, \ell) + n$	$O(n^2 + g)\mathsf{poly}(\lambda, \ell)$
GGHZ1,2 [26, 27]	$\mathsf{poly}(g_{max})\mathsf{poly}(\lambda, g_{max})$	$\mathsf{poly}(g_{max})\mathsf{poly}(\lambda, g_{max})$
Our KP1 (§4)	$O(n')\mathsf{poly}(\lambda, \ell) + n$	$O(g)\mathsf{poly}(\lambda, \ell)$
Our KP2 (§5)	$\mathsf{poly}(\lambda, \ell) + n$	$O(n^2 + g)\mathsf{poly}(\lambda, \ell)$

Table 4: CP-ABE for Circuits (sizes given in more details).

Schemes	|Cipher| (no. of bits)	|Key| (no. of bits)
GVW [31]	$\mathsf{poly}(g_{max})\mathsf{poly}(\lambda, g_{max}^{1/\epsilon})$	$\mathsf{poly}(g_{max})\mathsf{poly}(\lambda, g_{max}^{1/\epsilon})$
GGHSW [24]	$\mathsf{poly}(g_{max})\mathsf{poly}(\lambda, g_{max})$	$\mathsf{poly}(g_{max})\mathsf{poly}(\lambda, g_{max})$
BGG+1 [13]	$\mathsf{poly}(g_{max})\mathsf{poly}(\lambda, g_{max}^{1/\epsilon})$	$\mathsf{poly}(\lambda, g_{max}^{1/\epsilon}) + n$
BGG+2 [13]	$\mathsf{poly}(\lambda, g_{max}) + \|f\|$	$\mathsf{poly}(g_{max})\mathsf{poly}(\lambda, g_{max})$
GGHZ1,2 [26, 27]	$\mathsf{poly}(g_{max})\mathsf{poly}(\lambda, g_{max})$	$\mathsf{poly}(g_{max})\mathsf{poly}(\lambda, g_{max})$
Our CP1 (§6.2)	$O(g)\mathsf{poly}(\lambda, \ell)$	$O(n')\mathsf{poly}(\lambda, \ell) + n$
Our CP2 (§6.2)	$O(n^2 + g)\mathsf{poly}(\lambda, \ell)$	$\mathsf{poly}(\lambda, \ell) + n$

1.2 Our Contributions on New Framework

The main building block behind our ABE schemes is a new generic and modular framework, based on a new primitive called *multilinear pair encoding*. Our main result for framework can be summarized as:

Theorem 2. *(*Framework*, informally). Suppose that a (new) subgroup decision assumption in multilinear settings holds. A "doubly-selectively" secure multilinear pair encoding scheme for predicate R implies a fully secure ABE scheme for predicate R via a generic construction.*

Our Formalization. Our framework generalizes the recent framework by Attrapadung [3] (and Wee [51]), which works only in bilinear settings, to multi-linear settings. The framework of [3] provides an algebraic abstraction of *dual-system encryption* techniques, introduced by Waters [48] and utilized by many works [35, 37, 40, 36, 51], via a primitive called *pair encoding*. As seemingly inherent to bilinear settings, pair encoding of [3] is confined to only *linear* functions, so that the security proof under subgroup decision assumptions can be achieved. This prevents multiplication of variables in encodings since it would exactly destroy linearity. On the other hand, in generalizing to multi-linear settings, it is exactly the multiplication operation that we would like to enable. We resolve this conundrum by formalizing our multilinear version of pair encoding via a new notion we call *multilinear programs*, which allows both addition and multiplication. Our novelty then lies in identifying a subclass of multilinear programs that we call *associative programs* that will exactly admit the security proof under a subgroup decision assumption. Intuitively, associative programs allow us to compute the same encodings in two equivalent ways (hence the name, associative); one is used for the construction, and the other is used in simulation for the security proof.

"Doubly selective security" of pair encodings [3] can then be generalized to multi-linear settings in a natural manner. This consists of selective and co-selective notions for encodings, which mimic the definitions of selective and co-selective security of ABE. Selective notion refers to the situation where a ciphertext attribute is queried before a key attribute, while in co-selective notion, the order is reversed. This reflects one of the advantages of the framework: to achieve secure encodings in the selective notion, we can borrow algebraic techniques for selective security of ABE, which is much easier to achieve than full security.

Dual Conversion. Another advantage of the pair encoding framework is that it comes equipped with the powerful *dual conversion* [3, 9]. For a predicate $R : \mathbb{X} \times \mathbb{Y} \to \{0, 1\}$, its dual is defined by $\bar{R} : \mathbb{Y} \times \mathbb{X} \to \{0, 1\}$ where $\bar{R}(Y, X) := R(X, Y)$. Hence KP-ABE and CP-ABE are dual to each other. Attrapadung and Yamada [9] described a generic conversion that converts (bilinear) pair encoding P for a predicate R to another scheme $\overline{\mathsf{P}}$ for its dual while preserves doubly selective security and efficiency. More precisely, selective security of P implies co-selective security of $\overline{\mathsf{P}}$ (and analogously in an alternating manner). We generalize to multilinear settings in this paper. This, for the first time, allows us to convert KP-ABE to CP-ABE for circuits without using universal circuits, which was the only known (and highly inefficient) method so far.

Perspective. Ananth *et al.* [2] recently proposed a generic conversion from selective to full security in functional encryption (FE) for sufficiently expressive classes. (More on this later in §1.4.) However, they left an open problem of constructing a similar selective-to-full conversion for ABE. The ABE case is a harder task since the starting primitive, *i.e.,* selectively secure ABE, is less powerful than selectively secure FE. Our framework provides a partial solution by starting with any doubly selectively secure pair encodings (rather than any selective ABE), and converting to fully secure ABE via Theorem 2.

Potential Applications. Although we demonstrate applications of our framework by considering circuits, we may try to use it for plausibly constructing ABE for "moderate" classes in the Chomsky hierarchy (*e.g.,* Pushdown Automata, Linear-bounded Automata) with the hope that it can be done under multilinear maps with much *lower multi-linearity* (*e.g.,* small constant), which itself might be easier to achieve than general-purpose multi-linear maps. Indeed, this is the case for ABE for Deterministic Finite Automata, where the sufficient multi-linearity is *2* (*i.e.,* bilinear) [49, 3].

1.3 Our Techniques

Here, we highlight techniques for constructing new fully secure ABE for circuits. We first quickly note that the "information-theoretic variant" of dual system techniques [51, 3] will not work for circuit predicate due to "backtracking attack" [24] (due to the multi-fanout property of circuits).

We thus seek for "doubly selectively" secure encoding for the circuit predicate, which exhibits the "computational variant" of dual system techniques [36, 3]. Our blueprint starts with KP-ABE of GGHSW [24]. We immediately obtain selectively secure encoding by borrowing techniques for proving selective security of KP-ABE. The missing piece is then to prove the *co-selective* security for this encoding, or equivalently, the selective security of its *dual* encoding. Intuitively, we need new techniques to directly prove selective security of CP-ABE for circuits (without using universal circuits). One evidence that constructing selectively secure CP-ABE for circuits can be hard is that the Waters CP-ABE [47], which is for *Boolean formulae*, is proved under an already more complex (q-type) assumption than the KP-ABE counterpart [33], *à la* the Parallel BDHE [47].

Our goal is to generalize the selective proof of Waters' CP-ABE to the case of circuits. This poses two main issues. First, the output of a gate can be wired as an input to another gate (we call this a hierarchy issue). Second, and more essentially, the output of a gate (or a circuit input) can be wired as inputs to many gates (this is called multi-fanout). In the Waters CP-ABE, these two issues were not problematic since the scheme can be thought of using one big gate (multi-fan-in) that can express a linear secret-sharing scheme.

We solve these issues by designing a new assumption and a security proof that generalize "individual randomness" techniques similar to Waters [47], and Rouselakis and Waters [43] to work with circuits. The security proof works by "chaining" information on the paths from a given input gate to the output gate. One technical difficulty is that the number of chains can be exponential in the number of all gates (which would result in an exponential size assumption). We resolve this by giving out "decomposed" elements separately and letting the reduction multiply these terms on the fly to form the chains. In doing so, we carefully avoid enabling multiplication that results in a term that would trivially break the assumption. We note that our resulting assumption itself will not be tied to any particular circuit; it is only parameterized by the width, the input length, and the depth of the queried circuit.

Semi-adaptive Security under Simpler Assumptions. For the purpose of basing our schemes under *simple* assumptions, we consider *semi-adaptive* security of ABE [19, 45], which is an intermediate notion between selective and full security. We establish a *tight reduction* from semi-adaptive security of our generic construction to the selective security of pair encodings. Loosely speaking, this enables us to upgrade the KP-ABE of GGHSW [24] from selective to semi-adaptive security *for almost free*[4], since the selective security of our encodings relies on a similar (simple) assumption as that of GGHSW. See Table 1,2.

1.4 Related Work

Multilinear Map Candidates. Our framework is based on multi-linear maps. More precisely, we use *composite-order asymmetric graded encoding* systems (in a black-box manner). Multi-linear graded encoding systems was first proposed by Garg *et al.* [23] and subsequently by Coron *et al.* [20] (CLT13). Gentry *et al.* [29] extended the CLT13 system to the composite-order setting. Unfortunately, these candidates (and their variants, notably CLT15 [22]) were later shown to be broken [17, 21, 18].[5] As an alternative approach, multilinear maps based on indistinguishability obfuscation (IO) are recently proposed in [1]. However, the current security proof of IO under a polynomial-size set of assumptions requires complexity leveraging and hence exponential loss in reduction [30]. Nevertheless, this sheds some light on possibility of multilinear maps in the future.

[4]We although still need the subgroup decision assumption required for framework.

[5]As a caveat, some schemes are plausibly secure in the setting where encodings of zero are not given out. However, in ABE, we will need them for our security proof.

Fully Secure FE. Recently, Waters [50] and Ananth *et al.* [2] obtained fully-secure *functional encryption* (FE) for circuits. Waters provides a direct scheme based on IO [25, 30], while Ananth *et al.* provide a generic conversion from selective to full security for FE in unconditional manner and one can then use selectively secure FE from [25, 50], which is again based on IO. Due to an implicit exponential loss via IO, we do not elaborately include [50, 2] in Table 1,2.

As mentioned earlier, Garg *et al.* [27] obtained fully secure FE for circuits without obfuscation, hence also implies ABE with polynomial reduction. As in [26], it uses universal circuits, and thus can deal only with bounded-size circuits. Its asymptotic efficiency is also similar to [26] (*cf.* Table 1,2), albeit with much larger polynomials. Moreover, it requires stronger multilinear maps with the so-called Extension functionality [27].

Semi-adaptive Secure ABE. Very recently, Brakerski and Vaikuntanathan [11] obtained semi-adaptively secure KP-ABE for circuits that also achieves a remarkable feature of unbounded input length. Also very recently, Goyal *et al.* [32] proposed a generic selective-to-*semi-adaptive* conversion for ABE.

2 Preliminaries

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

ABE Syntax. An ABE scheme for predicate R consists of the following:

- $\mathsf{Setup}(1^\Lambda) \to (\mathsf{PK}, \mathsf{MSK})$: takes as input a a family index Λ (which includes the security parameter λ) 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 $Y \in \mathbb{Y}_\Lambda$, a message $M \in \mathcal{M}$ (the message space), and PK. It outputs a ciphertext CT.
- $\mathsf{KeyGen}(X, \mathsf{MSK}, \mathsf{PK}) \to \mathsf{SK}$: takes as input a key attribute $X \in \mathbb{X}_\Lambda$ 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 Λ, all $M \in \mathcal{M}$, $X \in \mathbb{X}_\Lambda$, $Y \in \mathbb{Y}_\Lambda$ such that $R_\Lambda(X, Y) = 1$. 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 $\mathsf{Setup}(1^\Lambda)$, then $\mathsf{Decrypt}(\mathsf{CT}, \mathsf{SK}) \to M$.

Security Notions for ABE. We use the standard definitions for full security and semi-adaptive security. Due to the lack of space, we refer to the full version. The advantages of $\mathcal{A}$ against the full and semi-adaptive security of the scheme ABE are denoted by $\mathsf{Adv}^{\mathsf{ABE}}_{\mathcal{A}}(\lambda), \mathsf{Adv}^{\mathsf{semi,ABE}}_{\mathcal{A}}(\lambda)$, respectively.

Circuit Notations. A circuit consists of six tuples $f = (\ell, n, \{m_i\}_{i \in [2,\ell]}, \mathsf{L}, \mathsf{R}, \mathsf{Type})$. We first note that it is wlog that we consider only *monotone* and *layered* circuits [24]. We let ℓ be the number of layers (the depth), n be the number of inputs, and m_i be the number of gates in the i-th layer for $i \in [2, \ell]$. Define $m := \max_{i\in[2,\ell]} m_i$, which represents the width. We also define $m_1 = n$. We define $\mathsf{Inputs} = \{w_{1,1}, \dots, w_{1,n}\}$, and for $i \in [2, \ell]$, $\mathsf{Gates}_i = \{w_{i,1}, \dots, w_{i,m_i}\}$. We let $\mathsf{Gates} = \bigcup_{i\in[2,n]} \mathsf{Gates}_i$, and let $\mathsf{Nodes} = \mathsf{Inputs} \cup \mathsf{Gates}$. Also denote $w_{\mathsf{top}} = w_{\ell,1}$ (the output gate). We define $\mathsf{Depth}(w_{i,j}) = i$ and $\mathsf{Num}(w_{i,j}) = j$. The two functions $\mathsf{L} : \mathsf{Gates} \to \mathsf{Gates} \smallsetminus \{w_{\mathsf{top}}\}$ and $\mathsf{R} : \mathsf{Gates} \to \mathsf{Gates} \smallsetminus \{w_{\mathsf{top}}\}$ identify the two input gates; that is, $\mathsf{L}(w_{i,j})$, $\mathsf{R}(w_{i,j})$ have outputs wired to $w_{i,j}$ as the first input (left input) and the second input (right input), respectively. We require that $\mathsf{Num}(\mathsf{L}(w_{i,j})) < \mathsf{Num}(\mathsf{R}(w_{i,j}))$. The function $\mathsf{Type} : \mathsf{Gates} \to \{\mathsf{OR}, \mathsf{AND}\}$ specifies the type of gate as either OR or AND. For $w \in \mathsf{Gates}$, we denote $f_w(x)$ to be the circuit evaluation of x at the output of w.

The predicate of KP-ABE for circuits is $R_{\lambda,n,\ell} : \mathbb{F}_{n,\ell} \times \{0,1\}^n \to \{0,1\}$ where $R(f, x) = f(x)$, where $\mathbb{F}_{n,\ell}$ is the set of all circuits with bounded input length n and bounded depth ℓ.

Composite-order Graded Encoding. We use the same syntax of (composite-order) graded encoding schemes as in [23, 20, 22]. Due to the lack of space, we postpone the full definition to the full version and only give a short description here. A composite-order asymmetric graded encoding scheme is parameterized by multi-linearity $\kappa \in \mathbb{N}$ and the number of subrings $\nu \in \mathbb{N}$. It allows us to encode a scalar a in a given ring $\mathcal{R} = \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_\nu}$, together with an index, which is a set $S \subseteq [1, \kappa]$, to a corresponding encoding, which we denote it by $[\,a\,]_S$. Intuitively, it is hard to recover the original scalar from its encoding, yet we are still allowed to perform some arithmetic operations on encodings. More precisely, we are allowed to perform operations $+, -, \cdot$ on encodings as

$$[\,a\,]_S + [\,b\,]_S = [\,a + b\,]_S, \qquad\qquad [\,a\,]_{S_1} \cdot [\,b\,]_{S_2} = [\,a \cdot b\,]_{S_1 \cup S_2},$$

and $-[\,a\,]_S = [\,-a\,]_S$, for $a, b \in \mathcal{R}$, $S, S_1, S_2 \subseteq [1, \kappa]$ such that $S_1 \cap S_2 = \emptyset$.

We also give some notation, originally appeared in [26], when the encoded scalar is projected to only subring components. In our ABE scheme, we will use $\nu = 2$. We denote $[\,a\,]^1_S := [\,a_1\,]_S$ where we set $a_1 \equiv a \pmod{N_1}$ and $a_1 \equiv 0 \pmod{N_2}$. $[\,a\,]^2_S$ is denoted similarly. Thus, $[\,a\,]^1_S$ and $[\,a\,]^2_S$ are independently distributed due to the Chinese Remainder Theorem. Also, we can decompose $[\,a\,]_S$ uniquely to $[\,a\,]_S = [\,a\,]^1_S + [\,a\,]^2_S$. Moreover, we have orthogonality: $[\,a_1\,]^1_{S_1} \cdot [\,a_2\,]^2_{S_2} = [\,0\,]_{S_1 \cup S_2}$, for any $a_1, a_2 \in \mathcal{R}$ (and disjoint S_1, S_2). More importantly, we can establish some subgroup decision problems. We describe this in §3.4.

Our scheme will not require public encoding functionality of *any* element; instead, we only need public encoding procedures of *unknown random* elements (as is the case for previous candidates [23, 20, 22]). We denote it by $[\,a\,]_\emptyset \leftarrow \mathsf{Samp}(\mathsf{param})$, which gives us a level-$\emptyset$ encoding of an unknown random element $a \in \mathcal{R}$. To encode it to level S, we need an encoding of 1, namely, $[\,1\,]_S$, to compute $[\,a\,]_\emptyset \cdot [\,1\,]_S = [\,a\,]_S$.

We briefly describe procedures for graded encodings here. $\mathsf{InstGen}(1^\lambda, \kappa, \nu)$ outputs $(\mathsf{param}, \mathsf{esk}, \{N_i\}_i)$, where param is public parameter, esk is a secret encoding key, and the order $\{N_i\}_i$ of subrings. By using esk, one can encode any $a \in R$ to $[\,a\,]_S^V$ for any S, V. Extraction algorithm Ext takes param and a level-$[1,\kappa]$ encoding $[\,a\,]_{[1,\kappa]}$ as inputs, and outputs a string $K \in \{0,1\}^\lambda$. We require that if $a \in_R \mathcal{R}$, then $K \in_R \{0,1\}^\lambda$ ($\in_R$ means uniformly distributed). As in all previous candidates, encodings may be non-deterministic. In such a case, we have a re-randomization procedure, and we require that the extraction of two encodings of the same value will result in the same string.

3 Our Dual System Framework in Multilinear Settings

In this section, we describe our framework for constructing ABE for any given predicate R from a new primitive called *multilinear pair encoding scheme* (for predicate R). This primitive is defined using formal variables in polynomials. To capture a formal system of graded encoding, we introduce the following notion of *indexed polynomials*, which are basically formal polynomials with the index being sets, and their operations mimic those of graded encodings.

Definition 1 (Formal Variables and Polynomials). *A* formal variable *is a bit string, and distinct variables denote different strings. A* fresh *variable is any string that has not been assigned to another former variable. A* formal polynomial *is a polynomial with formal variables.*

Definition 2 (Indexed Polynomial). *An* indexed polynomial *p is defined as a pair of formal polynomial a with coefficients in $\mathbb{Z}$ and a set $S \subseteq [1,\kappa]$. We denote it as $p = (\,a\,)_S$. We define its* formal operations *$+, -, \cdot$ as*

$$(\,a_1\,)_S + (\,a_2\,)_S = (\,a_1 + a_2\,)_S, \qquad (\,a_1\,)_{S_1} \cdot (\,a_2\,)_{S_2} = (\,a_1 \cdot a_2\,)_{S_1 \cup S_2},$$

and $-(\,a\,)_S = (\,-a\,)_S$, for $S, S_1, S_2 \subseteq [1,\kappa]$ and $S_1 \cap S_2 = \emptyset$.

Definition 3 (Indexed Singleton). *An* indexed singleton *is an indexed polynomial of a single variable (degree-1 monomial of a variable with coefficient 1) or a constant. The former is also called* indexed variable.

We formalize algorithms that perform formal operations on indexed polynomials as *multilinear programs*. Below, we then capture a kind of multilinear programs, called *associative programs*, that will be useful for our framework. We will typically denote a vector of indexed polynomials using bold fonts.

Definition 4 (Multilinear Program). *A* multilinear program*, say $\mathcal{P}$, is a procedure that takes a vector of indexed polynomials, say $\boldsymbol{x}$, as an input, performs only formal operations on its elements, and outputs a vector of indexed polynomials, say $\boldsymbol{v}$. When a security parameter λ is considered, we require the number of formal operations to be polynomial in λ.*

Definition 5 (Associative Program). *We say that a multilinear program $\mathcal{P}$ is* associative *over an ordered pair of vectors $(\boldsymbol{x}, \boldsymbol{w})$ of indexed singletons if its input is a vector of indexed polynomials each of which is of the form*[6]

$$(\,x_i\,)_\emptyset \cdot (\,w_{j_1}\,)_{T_{j_1}} \cdots (\,w_{j_k}\,)_{T_{j_k}},$$

[6]This form implies that all $T_{j_1}, \ldots, T_{j_k}$ are pairwise disjoint.

for some $(\, w_{j_1}\,)_{T_{j_1}}, \ldots, (\, w_{j_k}\,)_{T_{j_k}} \in \boldsymbol{w}$ *(for some* k*)*[7] *and some variable* x_i *such that there exists* $(\, x_i\,)_{S_i} \in \boldsymbol{x}$ *where* $T_{j_1} \cup \cdots \cup T_{j_k} = S_i$*.*

Using Associative Programs. The reason why we define associative programs is that we can identify the following associativity property:

$$(\, x_i\,)_{\emptyset} \cdot (\, w_{j_1}\,)_{T_{j_1}} \cdots (\, w_{j_k}\,)_{T_{j_k}} = (\, x_i\,)_{S_i} \cdot (\, w_{j_1}\,)_{\emptyset} \cdots (\, w_{j_k}\,)_{\emptyset} \tag{1}$$

(where $T_{j_1} \cup \cdots \cup T_{j_k} = S_i$). Intuitively, this property will allows us to have two ways of obtaining an equivalent element to be input to the program. Looking forward in our ABE context, one way will allow us to define ABE constructions and the other will allow us to simulate equivalent elements in the security proof. More precisely, we have a lemma below. Before that, we define two more notions.

A Useful Notation. We define the notation of *index-less projection* that maps a vector $\boldsymbol{a}$ of indexed polynomials to the same vector but with all indexes being $\emptyset$, denoted $\mathsf{V}_{\boldsymbol{a}}$. That is,

$$\boldsymbol{a} = \{\, (\, a_i\,)_{S_i} \mid i \in [1,k]\,\} \mapsto \mathsf{V}_{\boldsymbol{a}} := \{\, (\, a_i\,)_{\emptyset} \mid i \in [1,k]\,\}\,.$$

Extended Program. For a multilinear program $\mathcal{P}$ that is associative over $(\boldsymbol{x}, \boldsymbol{w})$, we define its *canonically extended multilinear program*, denoted as $\mathcal{E}_{\mathcal{P}}$, that takes $(\mathsf{V}_{\boldsymbol{x}}, \boldsymbol{w})$ as inputs, and does as follows. From $\mathsf{V}_{\boldsymbol{x}}$ and $\boldsymbol{w}$, $\mathcal{E}_{\mathcal{P}}$ computes each indexed polynomial $(\, x_i\,)_{\emptyset} \cdot (\, w_{j_1}\,)_{T_{j_1}} \cdots (\, w_{j_k}\,)_{T_{j_k}}$ that appears in the input set of $\mathcal{P}$ by formal multiplications. These thus comprise the whole input set to $\mathcal{P}$ and $\mathcal{E}_{\mathcal{P}}$ then finally computes $\mathcal{P}$ and outputs the result. We have the following:

Lemma 1 (Associativity). *For any vectors* $\boldsymbol{x}, \boldsymbol{w}$ *of indexed polynomials, for any multilinear program* $\mathcal{P}$ *that is associative over* $(\boldsymbol{x}, \boldsymbol{w})$*, we have*

$$\mathcal{E}_{\mathcal{P}}(\mathsf{V}_{\boldsymbol{x}}, \boldsymbol{w}) = \mathcal{E}_{\mathcal{P}}(\boldsymbol{x}, \mathsf{V}_{\boldsymbol{w}}).$$

Proof. The left-hand side and the right-hand side programs compute each input to $\mathcal{P}$ in the form of left-hand side and right-hand side of Eq.(1), respectively, which are equal. From that point on, both compute the same program $\mathcal{P}$. $\square$

Applying Graded-encoding Schemes to Formal System. Let us fix a graded encoding scheme and use the bracket notation. For an indexed polynomial $p = (\, a\,)_S$, we denote its corresponding graded-encoded element as $[\, p\,] = [\, a\,]_S$, where we abuse the bracket notation. It also applies component-wise to vectors.

Let $\mathcal{P}$ be a multilinear program with an input size z and an output size z' (sizes are the length of vectors). We define a corresponding algorithm that takes a vector of z graded-encoded elements as an input and outputs a vector of z' graded-encoded elements. This algorithm has the same procedure as $\mathcal{P}$ but replaces each formal operation $+, -, \cdot$ on indexed polynomials to operation $+, -, \cdot$ on graded-encoded elements, resp. We thus abuse the notation and denote this algorithm also as $\mathcal{P}$. The following lemma will be useful in the proof.

[7]Here, for a vector $\boldsymbol{x}$, the notation '$z \in \boldsymbol{x}$' means that z is an element in $\boldsymbol{x}$.

Lemma 2 (Decomposability). *For any multilinear program $\mathcal{P}$, any input $\boldsymbol{x}$, we have* $\mathcal{P}([\,\boldsymbol{x}\,]) = \mathcal{P}([\,\boldsymbol{x}\,]^1) + \mathcal{P}([\,\boldsymbol{x}\,]^2)$.

Proof. We decompose $[\,\boldsymbol{x}\,] = [\,\boldsymbol{x}\,]^1 + [\,\boldsymbol{x}\,]^2$. We claim that the decomposition will be preserved for each operation. For $+, -$, it is trivial. For multiplication we see that $([\,x\,]^1_{S_1} + [\,x\,]^2_{S_1})([\,x'\,]^1_{S_2} + [\,x'\,]^2_{S_2}) = [\,x\,]^1_{S_1} \cdot [\,x'\,]^1_{S_2} + [\,x\,]^2_{S_1} \cdot [\,x'\,]^2_{S_2}$, due to orthogonality. Hence, multiplication also preserves decomposition. $\square$

We also obtain the following two corollaries from Lemma 1,2, resp., which will be used in the security proof. They hold for any vectors $\boldsymbol{x}, \boldsymbol{w}$ of indexed polynomials, and for any multilinear program $\mathcal{P}$ that is associative over $(\boldsymbol{x}, \boldsymbol{w})$.

Corollary 1. $\mathcal{E}_{\mathcal{P}}([\,\mathsf{V}_{\boldsymbol{x}}\,], [\,\boldsymbol{w}\,]^1) = \mathcal{E}_{\mathcal{P}}([\,\boldsymbol{x}\,]^1, [\,\mathsf{V}_{\boldsymbol{w}}\,]) = \mathcal{E}_{\mathcal{P}}([\,\boldsymbol{x}\,]^1, [\,\mathsf{V}_{\boldsymbol{w}}\,]^1)$.

Proof. Since each input to $\mathcal{P}$ is of the form in Definition 5, when $[\,\boldsymbol{x}\,]$ is projected to $[\,\boldsymbol{x}\,]^1$, the input term to $\mathcal{P}$ is also projected due to orthogonality. Hence, we have the latter equality. The rest follows from Lemma 1. $\square$

Corollary 2. $\mathcal{E}_{\mathcal{P}}([\,\boldsymbol{x}\,], [\,\mathsf{V}_{\boldsymbol{w}}\,]) = \mathcal{E}_{\mathcal{P}}([\,\boldsymbol{x}\,]^1, [\,\mathsf{V}_{\boldsymbol{w}}\,]) + \mathcal{E}_{\mathcal{P}}([\,\boldsymbol{x}\,]^2, [\,\mathsf{V}_{\boldsymbol{w}}\,])$.

3.1 Multilinear Pair Encodings

Syntax. A multilinear pair encoding scheme for predicate family R consists of four deterministic polynomial-time algorithms as $\mathsf{P} = (\mathsf{Init}, \mathsf{EncK}, \mathsf{EncC}, \mathsf{Pair})$:[8]

- $\mathsf{Init}(\Lambda) \to \big(\kappa, \boldsymbol{h}_{\mathsf{c}}, \boldsymbol{h}_{\mathsf{k}}, n\big)$. It outputs a multi-linearity level κ, two vectors $\boldsymbol{h}_{\mathsf{c}}, \boldsymbol{h}_{\mathsf{k}}$ of indexed singletons, and an integer n specifying the number of all variables in $\boldsymbol{h}_{\mathsf{c}}, \boldsymbol{h}_{\mathsf{k}}$. We require that each singleton is h_i or 1, where $h_1, \ldots, h_n$ are variables. Let $\mathcal{S}_{\mathsf{c}}, \mathcal{S}_{\mathsf{k}}$ be the set of all indexes in $\boldsymbol{h}_{\mathsf{c}}, \boldsymbol{h}_{\mathsf{k}}$, respectively. Also let $S_{\mathsf{c}} = \bigcup_{S \in \mathcal{S}_{\mathsf{c}}} S$, $S_{\mathsf{k}} = \bigcup_{S \in \mathcal{S}_{\mathsf{k}}} S$. We require that $S_{\mathsf{c}} \cap S_{\mathsf{k}} = \emptyset$ and $S_{\mathsf{c}} \cup S_{\mathsf{k}} = [1, \kappa]$. Also we require $(\,1\,)_{S_{\mathsf{k}}} \in \boldsymbol{h}_{\mathsf{k}}$.[9]
- $\mathsf{EncK}(X, \boldsymbol{h}_{\mathsf{k}}) \to \big(\boldsymbol{B}, \boldsymbol{r}, \mathcal{P}_X\big)$. The *Key Encoding algorithm* takes $X \in \mathbb{X}_\Lambda$ and $\boldsymbol{h}_{\mathsf{k}}$ as inputs. It outputs two vectors $\boldsymbol{B}, \boldsymbol{r}$ of indexed polynomials, and a multilinear program $\mathcal{P}_X$. We require that $\boldsymbol{r} = \big((\,r_1\,)_{S_1}, \ldots, (\,r_m\,)_{S_m}\big)$, where $r_1, \ldots, r_m$ are fresh variables, for some $S_1, \ldots, S_m \subseteq S_{\mathsf{k}}$ for some integer m. We require that

  $$\mathcal{P}_X \text{ is associative over } (\boldsymbol{r}, \boldsymbol{h}_{\mathsf{k}}) \qquad \text{and} \qquad \mathcal{E}_{\mathcal{P}_X}(\mathsf{V}_{\boldsymbol{r}}, \boldsymbol{h}_{\mathsf{k}}) = \boldsymbol{B}.$$

  We distinguish the first indexed polynomial in $\boldsymbol{B}$ and require it to have index S_{k}; we call it the *Master-key Masking term*[10] and denote it as $(\,K_0\,)_{S_{\mathsf{k}}}$. Hence, $\boldsymbol{B} = \big((\,K_0\,)_{S_{\mathsf{k}}}, \boldsymbol{K}\big)$.

[8]We define *syntax* in such a way that it does not refer to multilinear maps. We do this so that it can accommodate both perfect and computational flavor of *security* (the former *will not* refer to mult-maps while the latter *will*, *cf.* §3.2), similarly to [3, 5].
[9]Or, $(\,1\,)_{S_{\mathsf{k}}}$ is computable from $\boldsymbol{h}_{\mathsf{k}}$. This is only for our purpose of dual conversion in §6.
[10]It will be used to mask the master-key in our generic scheme in §3.3, hence the name.

- $\mathsf{EncC}(Y, \boldsymbol{h}_{\mathsf{c}}) \to \big(\boldsymbol{C}, \boldsymbol{s}, \mathcal{P}_Y\big)$. The *Ciphertext Encoding algorithm* takes $Y \in \mathbb{Y}_\Lambda$ and $\boldsymbol{h}_{\mathsf{c}}$ as inputs. It outputs two vectors $\boldsymbol{C}, \boldsymbol{s}$ of indexed polynomials, and a multilinear program $\mathcal{P}_Y$. We require that $\boldsymbol{s} = \big((\, s_0\,)_{S_{\mathsf{c}}}, (\, s_1\,)_{T_1}, \ldots, (\, s_w\,)_{T_w}\big)$, where $s_0, s_1, \ldots, s_w$ are fresh variables, for some $T_1, \ldots, T_w \subseteq S_{\mathsf{c}}$ for some integer w. We require that

$$\mathcal{P}_Y \text{ is associative over } (\boldsymbol{s}, \boldsymbol{h}_{\mathsf{c}}) \qquad \text{and} \qquad \mathcal{E}_{\mathcal{P}_Y}(\mathsf{V}_{\boldsymbol{s}}, \boldsymbol{h}_{\mathsf{c}}) = \boldsymbol{C}.$$

  We distinguish the first indexed variable $(\, s_0\,)_{S_{\mathsf{c}}}$ in $\boldsymbol{s}$ where we require it to have index S_{c}. Also, we require that $(\, s_0\,)_{S_{\mathsf{c}}} \in \boldsymbol{C}$ and call it the *Base Randomness term.* (Wlog, we let it be the first indexed polynomial in $\boldsymbol{C}$).
- $\mathsf{Pair}(X, Y) \to \mathcal{P}_{X,Y}$. It outputs a description of multilinear program $\mathcal{P}_{X,Y}$.

Correctness. If $R(X, Y) = 1$ then $\mathcal{P}_{X,Y}(\boldsymbol{K}, \boldsymbol{C}) = (\, K_0 s_0\,)_{[1,\kappa]}$, for $\boldsymbol{K}, K_0, \boldsymbol{C}, s_0$ defined as above. In particular, $(\, K_0\,)_{S_{\mathsf{k}}}$ and $(\, s_0\,)_{S_{\mathsf{c}}}$ are the master-key masking term in $\boldsymbol{K}$ and the base randomness term in $\boldsymbol{C}$, respectively.

3.2 Security Definitions for Multilinear Pair Encoding

In this section, we formalize security notions for multilinear pair encoding. Looking forward, intuitively, they are formalized so as to provide indistinguishability between certain game switchings in the security proof for ABE. Nevertheless, it is simpler than the full security of ABE as the adversary will not obtain elements corresponding to public keys (graded-encoded $\boldsymbol{h}_{\mathsf{c}}$ in our context).

We formalize the computational security here, and postpone the information-theoretic one to the full version. It generalizes that of (bilinear) pair encoding in [3] (with a refinement regarding the number of queries in [9]). It consists of two sub-notions: *selective* and *co-selective master-key hiding* ($\mathsf{SMH}, \mathsf{CMH}$) in a graded encoding system G. We recall that $\mathsf{G.Samp}$ gives a level-$\emptyset$ encoding of random element. We use the same notation for a vector $\boldsymbol{x}$ of indexed polynomial: that is, $[\,\mathsf{V}_{\boldsymbol{x}}\,] \leftarrow \mathsf{Samp}(\mathsf{param})$ gives $[\, x_1\,]_\emptyset, \cdots, [\, x_k\,]_\emptyset \leftarrow \mathsf{Samp}(\mathsf{param})$.

Selective Master-key Hiding. Let $t_1, t_2 \in \mathbb{N}$. The (t_1, t_2)-SMH security is defined via the following game between the challenger $\mathcal{C}$ and the adversary $\mathcal{A}$ in the following order. For a definitional purpose, we fix $b \in \{0, 1\}$.

1 **Setup**: The challenger $\mathcal{C}$ setups the pair encoding $\mathsf{P.Init}(\Lambda) \to \big(\kappa, \boldsymbol{h}_{\mathsf{c}}, \boldsymbol{h}_{\mathsf{k}}, n\big)$, and setups the graded encoding $\mathsf{G.InstGen}(1^\lambda, \kappa, 2) \to (\mathsf{param}, \mathsf{esk}, N_1, N_2)$. $\mathcal{C}$ graded-encodes 1 for all indexes in $\mathbb{S}_{\mathsf{c}} \cup \mathbb{S}_{\mathsf{k}}$ to obtain $I := \big\{[\,1\,]^1_S, [\,1\,]^2_S\big\}_{S \in \mathbb{S}_{\mathsf{c}} \cup \mathbb{S}_{\mathsf{k}}}$. The input to $\mathcal{A}$ is (param, I). $\mathcal{C}$ further samples $[\,\mathsf{V}_{\boldsymbol{h}}\,], [\,\beta\,]_\emptyset \leftarrow \mathsf{Samp}(\mathsf{param})$ for using in the next phases. From b, define

$$\beta^\star := 0 \quad \text{if } b = 0 \qquad\qquad \text{and} \qquad\qquad \beta^\star := \beta \quad \text{if } b = 1.$$

2 **Ciphertext query phase**: $\mathcal{A}$ makes a query Y for graded-encoded EncC. $\mathcal{C}$ then runs $\mathsf{P.EncC}(Y, \boldsymbol{h}_{\mathsf{c}}) \to \big(\boldsymbol{C}, \boldsymbol{s}, \mathcal{P}_Y\big)$, samples $[\,\mathsf{V}_{\boldsymbol{s}}\,] \leftarrow \mathsf{Samp}(\mathsf{param})$, and returns $[\,\boldsymbol{C}\,]^2$ to $\mathcal{A}$. At most t_1 ciphertext queries are allowed.

3 **Key query phase**: $\mathcal{A}$ makes a query X for graded-encoded EncK. We require that $R(X,Y) = 0$ for all queries Y in the previous phase. $\mathcal{C}$ runs $\mathsf{P.EncK}(X, \boldsymbol{h}_{\mathsf{k}}) \to (\boldsymbol{B}, \boldsymbol{r}, \mathcal{P}_X)$, samples $[\,\mathsf{V}_{\boldsymbol{r}}\,] \leftarrow \mathsf{Samp}(\mathsf{param})$. Parse $\boldsymbol{B} = ((\,K_0\,)_{S_{\mathsf{k}}}, \boldsymbol{K})$ and returns

$$([\,\beta^\star\,]^2_{S_{\mathsf{k}}} + [\,K_0\,]^2_{S_{\mathsf{k}}}, [\,\boldsymbol{K}\,]^2)$$

to $\mathcal{A}$. At most t_2 key queries are allowed.

4 **Guess**: The adversary $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$.

Let $\mathsf{Exp}_b(\lambda)$ denote the output of the game. We define the advantage of $\mathcal{A}$ as $\mathsf{Adv}_{\mathcal{A}}^{(t_1,t_2)\text{-}\mathsf{SMH(P)}}(\lambda) := |\Pr[\mathsf{Exp}_0(\lambda) = 1] - \Pr[\mathsf{Exp}_1(\lambda) = 1]|$. We say that P is (t_1, t_2)-SMH in G if the advantage is negligible for all polynomial time attackers $\mathcal{A}$. If t_i is not a-priori bounded, we denote $t_i = \mathsf{poly}$.

Remark 1. We note that, in the above game, $\mathcal{C}$ can compute the returned graded-encoded elements by using I and known level-$\emptyset$ graded-encoded variables, $[\,\mathsf{V}_{\boldsymbol{h}}\,]$, $[\,\mathsf{V}_{\boldsymbol{s}}\,], [\,\mathsf{V}_{\boldsymbol{r}}\,]$. Also note that, if graded encoding is noisy, $\mathcal{C}$ re-randomizes answers to have a certain noise level before returning back to $\mathcal{A}$.

Co-selective Master-key Hiding. The (t_1, t_2)-CMH security is defined in exactly the same manner as that of SMH except that we swap the order of the two query phases: we let the key query phase comes before the ciphertext query phase. Now, t_1, t_2 denotes the number of key and ciphertext queries, respectively. We note that an analogous restriction is required in the ciphertext query phase.

3.3 Our Generic ABE Construction for Any Predicate

Construction. From a multi-linear pair encoding scheme P for predicate R and a graded encoding system G, we construct an ABE scheme for R, denoted $\mathsf{ABE(P, G)}$, as follows. We let the message space be $\mathcal{M} = \{0,1\}^\lambda$.

- $\mathsf{Setup}(1^\Lambda) \to (\mathsf{PK}, \mathsf{MSK})$. Initialize $\mathsf{P.Init}(\Lambda) \to (\kappa, \boldsymbol{h}_{\mathsf{c}}, \boldsymbol{h}_{\mathsf{k}}, n)$ and generate $\mathsf{G.InstGen}(1^\lambda, \kappa, 2) \to (\mathsf{param}, \mathsf{esk}, N_1, N_2)$. For $i \in [1,n]$, sample $h_i \xleftarrow{\$} \mathcal{R}$. Sample $\alpha \xleftarrow{\$} \mathcal{R}$. It graded-encodes all elements in $\boldsymbol{h}_{\mathsf{c}}, \boldsymbol{h}_{\mathsf{k}}$, in $\mathbb{Z}_{N_1}$ components (by using the secret encoding key esk). Output:

$$\mathsf{PK} = \left(\mathsf{param}, [\,\boldsymbol{h}_{\mathsf{c}}\,]^1, [\,\alpha\,]^1_{[1,\kappa]}\right), \qquad \mathsf{MSK} = \left(\mathsf{param}, [\,\boldsymbol{h}_{\mathsf{k}}\,]^1, [\,\alpha\,]^{1,2}_{S_{\mathsf{k}}}\right).$$

- $\mathsf{Encrypt}(\mathsf{PK}, Y, M) \to \mathsf{CT}$. Run $\mathsf{P.EncC}(Y, \boldsymbol{h}_{\mathsf{c}}) \to (\boldsymbol{C}, \boldsymbol{s}, \mathcal{P}_Y)$. Sample $[\,\mathsf{V}_{\boldsymbol{s}}\,] \leftarrow \mathsf{Samp}(\mathsf{param})$. Compute

$$[\,\boldsymbol{C}\,]^1 = \mathcal{E}_{\mathcal{P}_Y}([\,\mathsf{V}_{\boldsymbol{s}}\,], [\,\boldsymbol{h}_{\mathsf{c}}\,]^1).$$

  It then computes $[\,\alpha s_0\,]^1_{[1,\kappa]} = [\,\alpha\,]^1_{[1,\kappa]} \cdot [\,s_0\,]_\emptyset$ and $C_0 = \mathsf{G.Ext}(\mathsf{param}, [\,\alpha s_0\,]^1_{[1,\kappa]}) \oplus M$. Output $\mathsf{CT} = ([\,\boldsymbol{C}\,]^1, C_0)$.

- $\mathsf{KeyGen}\big(\mathsf{MSK}, X\big) \to \mathsf{SK}$. Run $\mathsf{P.EncK}(X, \boldsymbol{h}_{\mathsf{k}}) \to \big((K_0)_{S_{\mathsf{k}}}, \boldsymbol{K}, \boldsymbol{r}, \mathcal{P}_X\big)$. Sample $[\mathsf{V}_{\boldsymbol{r}}] \leftarrow \mathsf{Samp}(\mathsf{param})$. Compute
$$([K_0]^1_{S_{\mathsf{k}}}, [\boldsymbol{K}]^1) = \mathcal{E}_{\mathcal{P}_X}([\mathsf{V}_{\boldsymbol{r}}], [\boldsymbol{h}_{\mathsf{k}}]^1)$$
Output $\mathsf{SK} = \Big([\alpha]^{1,2}_{S_{\mathsf{k}}} + [K_0]^1_{S_{\mathsf{k}}}, [\boldsymbol{K}]^1\Big)$.
- $\mathsf{Decrypt}(\mathsf{SK}, \mathsf{CT}) \to M$. Assume $R(X,Y) = 1$. Parse $[s_0]^1_{S_{\mathsf{c}}}$ from CT. Run $\mathsf{P.Pair}(X,Y) \to \mathcal{P}_{X,Y}$. Compute $\mathcal{P}_{X,Y}([\boldsymbol{K}]^1, [\boldsymbol{C}]^1) \to [K_0 s_0]^1_{[1,\kappa]}$ and
$$([\alpha]^{1,2}_{S_{\mathsf{k}}} + [K_0]^1_{S_{\mathsf{k}}}) \cdot [s_0]^1_{S_{\mathsf{c}}} - [K_0 s_0]^1_{[1,\kappa]} = [\alpha s_0]^1_{[1,\kappa]},$$
and obtain M as $C_0 \oplus \mathsf{G.Ext}(\mathsf{param}, [\alpha s_0]^1_{[1,\kappa]})$.

Semi-functional Algorithms. In the security proof, we will use semi-functional algorithms defined below. In these, we will use hatted variables which are fresh variables (thus are independent from their non-hatted counterparts). For a vector $\boldsymbol{x}$ of indexed variables, let $\hat{\boldsymbol{x}}$ be a vector of indexed variables where we swap each variable in $\boldsymbol{x}$ with its hatted counterpart. In particular, this defines $\hat{\boldsymbol{h}}_{\mathsf{c}}, \hat{\boldsymbol{h}}_{\mathsf{k}}, \hat{\boldsymbol{s}}, \hat{\boldsymbol{r}}$.

- $\mathsf{SFSetup}(1^\Lambda) \to (\mathsf{PK}, \mathsf{MSK}, \widehat{\mathsf{PK}}, \widehat{\mathsf{MSK}})$. This is exactly the same as Setup albeit it additionally outputs $\widehat{\mathsf{PK}}$, $\widehat{\mathsf{MSK}}$ as follows. For $i \in [1,n]$, sample $\hat{h}_i \xleftarrow{\$} \mathcal{R}$. It graded-encodes all elements in $\hat{\boldsymbol{h}}_{\mathsf{c}}, \hat{\boldsymbol{h}}_{\mathsf{k}}$ projecting to subring $\mathbb{Z}_{N_2}$ and outputs:
$$\widehat{\mathsf{PK}} = \Big([\hat{\boldsymbol{h}}_{\mathsf{C}}]^2, [\alpha]^2_{[1,\kappa]}\Big), \qquad\qquad \widehat{\mathsf{MSK}} = [\hat{\boldsymbol{h}}_{\mathsf{K}}]^2,$$
It also outputs $[1]^2_{S_{\mathsf{k}}}$ (for using as an input to $\mathsf{SFKeyGen}$ below).
- $\mathsf{SFEncrypt}\big(\mathsf{PK}, Y, M, \widehat{\mathsf{PK}}\big) \to \mathsf{CT}$. First, proceed as $\mathsf{Encrypt}\big(\mathsf{PK}, Y, M\big)$ to obtain $[\boldsymbol{C}]^1$ and $[\alpha s_0]^1_{[1,\kappa]}$. Sample $[\mathsf{V}_{\hat{\boldsymbol{s}}}] \leftarrow \mathsf{Samp}(\mathsf{param})$. Compute
$$[\widehat{\boldsymbol{C}}]^2 \coloneqq \mathcal{E}_{\mathcal{P}_Y}\Big([\mathsf{V}_{\hat{\boldsymbol{s}}}], [\hat{\boldsymbol{h}}_{\mathsf{c}}]^2\Big).$$
Compute $[\alpha \hat{s}_0]^2_{[1,\kappa]} = [\alpha]^2_{[1,\kappa]} \cdot [\hat{s}_0]_\emptyset$, and $\widehat{C}_0 = \mathsf{G.Ext}(\mathsf{param}, [\alpha s_0]^1_{[1,\kappa]} + [\alpha \hat{s}_0]^2_{[1,\kappa]}) \oplus M$. Output $\mathsf{CT} = \Big([\boldsymbol{C}]^1 + [\widehat{\boldsymbol{C}}]^2,\ \widehat{C}_0\Big)$.
- $\mathsf{SFKeyGen}\big(\mathsf{MSK}, X, \mathsf{type}, \mathsf{aux}\big) \to \mathsf{SK}$. aux is an auxiliary input. If $\mathsf{type} = 1$, let $\mathsf{aux} = \widehat{\mathsf{MSK}}$. If $\mathsf{type} = 2$, let $\mathsf{aux} = (\widehat{\mathsf{MSK}}, [1]^2_{S_{\mathsf{k}}}, [\beta]_\emptyset)$. If $\mathsf{type} = 3$, let $\mathsf{aux} = ([1]^2_{S_{\mathsf{k}}}, [\beta]_\emptyset)$. First, run $\mathsf{KeyGen}\big(\mathsf{MSK}, X\big) \to ([\alpha]^{1,2}_{S_{\mathsf{k}}} + [K_0]^1_{S_{\mathsf{k}}}, [\boldsymbol{K}]^1)$. Sample $[\mathsf{V}_{\hat{\boldsymbol{r}}}] \leftarrow \mathsf{Samp}(\mathsf{param})$. If $\mathsf{type} = 1$ or 2, compute
$$([\widehat{K_0}]^2_{S_{\mathsf{k}}}, [\widehat{\boldsymbol{K}}]^2) \coloneqq \mathcal{E}_{\mathcal{P}_X}\Big([\mathsf{V}_{\hat{\boldsymbol{r}}}], [\hat{\boldsymbol{h}}_{\mathsf{k}}]^2\Big).$$
For $\mathsf{type} = 2$ or 3, also compute $[\beta]^2_{S_{\mathsf{k}}} = [1]^2_{S_{\mathsf{k}}} \cdot [\beta]_\emptyset$. Output
$$\mathsf{SK} = \begin{cases} \big([\alpha]^{1,2}_{S_{\mathsf{k}}} + [K_0]^1_{S_{\mathsf{k}}} \phantom{+ [\beta]^2_{S_{\mathsf{k}}}} + [\widehat{K_0}]^2_{S_{\mathsf{k}}},\ [\boldsymbol{K}]^1 + [\widehat{\boldsymbol{K}}]^2\big) & \text{if } \mathsf{type} = 1 \\ \big([\alpha]^{1,2}_{S_{\mathsf{k}}} + [K_0]^1_{S_{\mathsf{k}}} + [\beta]^2_{S_{\mathsf{k}}} + [\widehat{K_0}]^2_{S_{\mathsf{k}}},\ [\boldsymbol{K}]^1 + [\widehat{\boldsymbol{K}}]^2\big) & \text{if } \mathsf{type} = 2\,. \\ \big([\alpha]^{1,2}_{S_{\mathsf{k}}} + [K_0]^1_{S_{\mathsf{k}}} + [\beta]^2_{S_{\mathsf{k}}} \phantom{+ [\widehat{K_0}]^2_{S_{\mathsf{k}}}},\ [\boldsymbol{K}]^1 \phantom{+ [\widehat{\boldsymbol{K}}]^2}\big) & \text{if } \mathsf{type} = 3 \end{cases}$$

3.4 Multilinear Subgroup Decision Assumption

We introduce a new subgroup decision assumption in multilinear settings. It generalizes the First and Second Subgroup Decision Assumptions in [35, 37, 3], which are defined in bilinear groups, to multilinear settings. We require the composite settings with only two subrings, instead of three as in [35, 37, 51, 3].

Definition 6 (MSD). *For $\kappa \in \mathbb{N}$, $U \subseteq [1,\kappa]$, we define the (κ, U)-*Multilinear Subgroup Decision Assumption *as follows. Let $\mathsf{InstGen}(1^\lambda, \kappa, 2) \to (\mathsf{param}, \mathsf{esk}, N_1, N_2)$. For $i \in U$, let $z_i \xleftarrow{\$} \mathcal{R}$. Define $\bar{U} = [1,\kappa] \setminus U$. For $i \in \bar{U}$, let $a_i \xleftarrow{\$} \mathcal{R}$. It states that the following distributions are computationally indistinguishable:*

$$\left(D, Z = \left\{[\,z_i\,]^{1}_{\{i\}}\right\}_{i\in U}\right) \qquad \text{and} \qquad \left(D, Z = \left\{[\,z_i\,]^{1,2}_{\{i\}}\right\}_{i\in U}\right),$$

where $D = \left(\mathsf{param}, I = \left\{[\,1\,]^{1}_{\{i\}}\right\}_{i\in[1,\kappa]}, A = \left\{[\,a_i\,]^{1,2}_{\{i\}}\right\}_{i\in\bar{U}}, B = [\,1\,]^{2}_{U}\right)$.

We are able to use only two subrings thanks to *asymmetric* settings. Intuitively, if we were to use symmetric ones, B, which has only the $\mathbb{Z}_{N_2}$ component, can be used to test Z by multiplying to it. (And hence to prevent it, a mask from another subgroup was needed). In asymmetric settings, we cannot multiply B with any element in Z since their indexes intersect.

Properties from MSD. We describe some properties from MSD that will be used in the security proof. We can write $Z_i := [\,z_i\,]^{1,2}_{\{i\}} = [\,z_{i,1}\,]^{1}_{\{i\}} + [\,z_{i,2}\,]^{2}_{\{i\}}$. The problem can be restated as to distinguish whether $z_{i,2} = 0$ for all $i \in U$ or $z_{i,2} \in_R \mathcal{R}$ for all $i \in U$. For further use in the proofs, we denote the following. For $S \subseteq U$, we denote $Z_S := \prod_{i\in S} Z_i$ and $z_{S,j} := \prod_{i\in S} z_{i,j}$, for $j = 1, 2$. Hence, we have $Z_S = [\,z_{S,1}\,]^{1}_{S} + [\,z_{S,2}\,]^{2}_{S}$ by orthogonality. Similarly, we write $A_i := [\,a_i\,]^{1,2}_{\{i\}} = [\,a_{i,1}\,]^{1}_{\{i\}} + [\,a_{i,2}\,]^{2}_{\{i\}}$. For $S \subseteq \bar{U}$, $A_S := \prod_{i\in S} A_i$ and $a_{S,j} := \prod_{i\in S} a_{i,j}$, for $j = 1, 2$; hence, we have $A_S = [\,a_{S,1}\,]^{1}_{S} + [\,a_{S,2}\,]^{2}_{S}$. We also note that from I, for any $S \subseteq [1,\kappa]$, we can compute $\prod_{i\in S}[\,1\,]^{1}_{\{i\}} = [\,1\,]^{1}_{S}$.

3.5 Security for Our Generic Construction

Theorem 3. *Suppose that a pair encoding* P *for predicate* R *is* $(1,1)$-CMH *and* $(1,\mathsf{poly})$-SMH *in* G. *Suppose the* MSD *Assumption holds in* G. *Then, our generic construction,* $\mathsf{ABE}(\mathsf{P},\mathsf{G})$*, for predicate* R *is fully secure. More precisely, for any PPT adversary* $\mathcal{A}$*, there exist PPT algorithms* $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3, \mathcal{B}_4$*, whose running times are the same as* $\mathcal{A}$ *plus some polynomial times, such that for any* λ*,*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{ABE}(\mathsf{P},\mathsf{G})}_{\mathcal{A}}(\lambda) \le \mathsf{Adv}^{(\kappa,S_{\mathsf{c}})\text{-}\mathsf{MSD}}_{\mathcal{B}_1}(\lambda) &+ (2q_1+2)\mathsf{Adv}^{(\kappa,S_{\mathsf{k}})\text{-}\mathsf{MSD}}_{\mathcal{B}_2}(\lambda)\\ &+ q_1\mathsf{Adv}^{(1,1)\text{-}\mathsf{CMH}(\mathsf{P})}_{\mathcal{B}_3}(\lambda) + \mathsf{Adv}^{(1,\mathsf{poly})\text{-}\mathsf{SMH}(\mathsf{P})}_{\mathcal{B}_4}(\lambda),\end{aligned}$$

where q_1 is the number of queries in phase 1, κ is the multi-linearity level, and $S_{\mathsf{c}}, S_{\mathsf{k}} \subseteq [1,\kappa]$ are specified by the encoding scheme P.

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

$$\begin{tikzcd}
\mathsf{G}_{\mathrm{real}} \arrow[r, "\mathsf{MSD}"'] & \mathsf{G}_0 \arrow[r] & \mathsf{G}_{1,1} \arrow[r] & \cdots \arrow[r] & \mathsf{G}_{k-1,3} \arrow[r, "\mathsf{MSD}"'] & \mathsf{G}_{k,1} \arrow[r, "\mathsf{CMH}"'] & \mathsf{G}_{k,2} \arrow[r, "\mathsf{MSD}"'] & \mathsf{G}_{k,3} \arrow[r] & \cdots \arrow[r] & \mathsf{G}_{q_1,3} \arrow[r, "\mathsf{MSD}"'] & \mathsf{G}_{q_1+1} \arrow[r, "\mathsf{SMH}"'] & \mathsf{G}_{q_1+2} \arrow[r, "\mathsf{MSD}"'] & \mathsf{G}_{q_1+3} \arrow[r, "="'] & \mathsf{G}_{\mathrm{final}}
\end{tikzcd}$$

where each game is defined as follows.[11] $\mathsf{G}_{\mathrm{real}}$ is the actual security game. 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.
- $\mathsf{G}_{k,i}$ where $k \in [1, q_1]$, $i \in \{1,2,3\}$: We modify the k-th queried key to be semi-functional of type-i. We use fresh β for each key (for type $i = 2, 3$).
- G_{q_1+i} where $i \in \{1,2,3\}$: We modify all the keys in phase 2 to be semi-functional of type-i at once. We use the same β for all these keys (for type $i = 2, 3$).
- $\mathsf{G}_{\mathrm{final}}$: We modify the challenge to encrypt a random message.

In the final game, the advantage of $\mathcal{A}$ is trivially 0. We prove the indistinguishability between all these adjacent games. Due to the lack of space, we provide only two of these lemmata below and defer the rest to the full version. In these lemmata, we define $\mathsf{G}_j\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ABE(P,G)}}(\lambda)$ to be the advantage of $\mathcal{A}$ in the game G_j. Summing all the advantage differences from these lemmata, we obtain the advantage bound stated as in Theorem 3. $\square$

Proof Intuition. We describe some intuition for proofs of lemmata for game switching with key modifications. We consider two categories. (Ciphertext modification works similarly to the first category below).

For the game switching where β is not changed (normal to type-1 keys, type-2 to type-3 keys), the difference between the two games is exactly the key encodings in the $\mathbb{Z}_{N_2}$ component. We thus simulate the key randomness $[\,\boldsymbol{r}\,]$ using Z from the MSD problem instance, where we have to distinguish whether Z has the $\mathbb{Z}_{N_2}$ component or not. The reduction would then compute $\mathcal{E}_{\mathcal{P}}([\,\boldsymbol{r}\,], [\,\mathsf{V}_{\boldsymbol{h}_{\mathsf{k}}}\,])$, where $[\,\mathsf{V}_{\boldsymbol{h}_{\mathsf{k}}}\,]$ is sampled by the reduction and is used for generating other keys. But, due to *associativity* (Lemma 1), this is equal to $\mathcal{E}_{\mathcal{P}}([\,\mathsf{V}_{\boldsymbol{r}}\,], [\,\boldsymbol{h}_{\mathsf{k}}\,])$, and due to *decomposability* (Lemma 2), we can deduce that it is exactly the form of normal or semi-functional key as per definition, depending on whether Z has the $\mathbb{Z}_{N_2}$ component or not. Hence, the reduction to MSD is established.

For the game switching where β is changed (type-1 to type-2 keys), the difference between the two games is exactly β. We can embed exactly the challenge from the CMH or SMH game, where we have to distinguish if $\beta^\star = 0$ or $\beta^\star$ is random. If the switched key is in phase 1, we use CMH, where the key query comes before the ciphertext query. If the switched key is in phase 2, we use SMH. The parameter $(1, \mathsf{poly})$ of SMH lets us switch all post-challenge keys at once.

[11]More precise definitions of these games are given in the full version.

We provide here the proofs for the game switching from $\mathsf{G}_{\mathrm{real}}$ to G_0 (changing normal to semi-functional ciphertext), and $\mathsf{G}_{k,1}$ to $\mathsf{G}_{k,2}$ (changing type-1 to type-2 semi-functional key).

Lemma 3 **($\mathsf{G}_{\mathrm{real}}$ to G_0).** *For any adversary $\mathcal{A}$, there exists an algorithm $\mathcal{B}$ that breaks the (κ, S_{c})-MSD Assumption with* $|\mathsf{G}_{\mathrm{real}}\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ABE(P,G)}}(\lambda) - \mathsf{G}_0\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ABE(P,G)}}(\lambda)| \leq \mathsf{Adv}_{\mathcal{B}}^{(\kappa,S_{\mathsf{c}})\text{-}\mathsf{MSD}}(\lambda)$.

Proof. As an instance of the (κ, S_{c})-MSD Assumption, the algorithm $\mathcal{B}$ obtains an input $(D, \{Z_i\}_{i \in S_{\mathsf{c}}})$ where $Z_i = [\, z_{i,1} \,]^1_{\{i\}} + [\, z_{i,2} \,]^2_{\{i\}}$. $\mathcal{B}$'s task is to guess whether $z_{i,2} = 0$ or $z_{i,2} \in_R \mathcal{R}$ (both for all $i \in S_{\mathsf{c}}$).

$\mathcal{B}$ simulates $\mathsf{SFSetup}$ as follows. First, $\mathcal{B}$ samples $[\, \tilde{\alpha} \,]_{\emptyset} \leftarrow \mathsf{Samp}(\mathsf{param})$ and sets $[\, \alpha \,]^{1,2}_{S_{\mathsf{k}}} = [\, \tilde{\alpha} \,]_{\emptyset} \cdot A_{S_{\mathsf{k}}}$ for MSK, and $[\, \alpha \,]^1_{[1,\kappa]} = [\, \alpha \,]^{1,2}_{S_{\mathsf{k}}} \cdot [\, 1 \,]^1_{S_{\mathsf{c}}}$ for PK.

For $i \in [1, n]$, $\mathcal{B}$ samples $[\, \tilde{h}_i \,]_{\emptyset} \leftarrow \mathsf{Samp}(\mathsf{param})$. For each indexed variable $(\, h_i \,)_S$ in $\boldsymbol{h}_{\mathsf{c}}$ or $\boldsymbol{h}_{\mathsf{k}}$ (for some S), $\mathcal{B}$ computes $[\, h_i \,]^1_S = [\, \tilde{h}_i \,]_{\emptyset} \cdot [\, 1 \,]^1_S$ (computable since $[\, 1 \,]^1_S$ is available in I) and *implicitly* sets $[\, \hat{h}_i \,]^2_S = [\, \tilde{h}_i \,]_{\emptyset} \cdot [\, 1 \,]^2_S$ (unknown since $[\, 1 \,]^2_S$ is not available). Hence we have $h_i = \tilde{h}_i \bmod N_1$ and $\hat{h}_i = \tilde{h}_i \bmod N_2$. Due to CRT, h_i and $\hat{h}_i$ distribute independently, as required by definition of $\mathsf{SFSetup}$. This feature is called *parameter-hiding* [36, 3]. All these terms completely define $\mathsf{PK}, \mathsf{MSK}$. PK is given to $\mathcal{A}$.

Phase 1,2. When $\mathcal{A}$ makes the j-th key query for $X^{(j)}$, $\mathcal{B}$ generates a key as usual: $\mathsf{SK}_j \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, X^{(j)})$.

Challenge. The adversary $\mathcal{A}$ outputs messages $M_0, M_1 \in \{0,1\}^{\lambda}$ along with a target $Y^{\star}$. $\mathcal{B}$ chooses $\mathfrak{b} \xleftarrow{\$} \{0,1\}$. $\mathcal{B}$ runs $\mathsf{P.EncC}(Y^{\star}, \boldsymbol{h}_{\mathsf{c}}) \to \big(\boldsymbol{C}, \boldsymbol{s}, \mathcal{P}_{Y^{\star}}\big)$. Let $w = |\boldsymbol{s}| - 1$. For $i \in [0, w]$, sample $[\, \tilde{s}_i \,]_{\emptyset} \leftarrow \mathsf{Samp}(\mathsf{param})$. Suppose that $\boldsymbol{s} = \big((\, s_0 \,)_{S_{\mathsf{c}}}, (\, s_1 \,)_{T_1}, \ldots, (\, s_w \,)_{T_w}\big)$. $\mathcal{B}$ then computes

$$\begin{aligned}
[\, \bar{\boldsymbol{s}} \,] &:= \Big([\, \tilde{s_0} \,]_{\emptyset} \cdot Z_{S_{\mathsf{c}}},\, [\, \tilde{s}_1 \,]_{\emptyset} \cdot Z_{T_1}, \ldots,\, [\, \tilde{s}_w \,]_{\emptyset} \cdot Z_{T_w}\Big), \\
[\, \bar{\boldsymbol{C}} \,] &:= \mathcal{E}_{\mathcal{P}_{Y^{\star}}}\Big([\, \bar{\boldsymbol{s}} \,], [\, \mathsf{V}_{\tilde{\boldsymbol{h}}_{\mathsf{c}}} \,]\Big), \\
\bar{C}_0 &:= \mathsf{G.Ext}\Big(\mathsf{param},\, [\, \alpha \,]^{1,2}_{S_{\mathsf{k}}} \cdot [\, \tilde{s_0} \,]_{\emptyset} \cdot Z_{S_{\mathsf{c}}}\Big) \oplus M_{\mathfrak{b}}
\end{aligned}$$

where $Z_S = [\, z_{S,1} \,]^1_S + [\, z_{S,2} \,]^2_S$ for $S \subseteq S_{\mathsf{c}}$ is indeed derivable from the problem instance. (See at the end of §3.4 for the definition of Z_S). $\mathcal{B}$ sets $\mathsf{CT} = ([\, \bar{\boldsymbol{C}} \,], \bar{C}_0)$. We claim that CT properly distributes as a normal or semi-functional ciphertext. To prove this, we observe that

$$[\, \bar{\boldsymbol{C}} \,] = \mathcal{E}_{\mathcal{P}_{Y^{\star}}}\Big([\, \bar{\boldsymbol{s}} \,], [\, \mathsf{V}_{\tilde{\boldsymbol{h}}_{\mathsf{c}}} \,]\Big) = \mathcal{E}_{\mathcal{P}_{Y^{\star}}}\Big([\, \bar{\boldsymbol{s}} \,]^1, [\, \mathsf{V}_{\tilde{\boldsymbol{h}}_{\mathsf{c}}} \,]\Big) + \mathcal{E}_{\mathcal{P}_{Y^{\star}}}\Big([\, \bar{\boldsymbol{s}} \,]^2, [\, \mathsf{V}_{\tilde{\boldsymbol{h}}_{\mathsf{c}}} \,]\Big) \quad (2)$$

$$= \mathcal{E}_{\mathcal{P}_{Y^{\star}}}\Big([\, \mathsf{V}_{\boldsymbol{s}} \,], [\, \boldsymbol{h}_{\mathsf{c}} \,]^1\Big) + \mathcal{E}_{\mathcal{P}_{Y^{\star}}}\Big([\, \mathsf{V}_{\hat{\boldsymbol{s}}} \,], [\, \hat{\boldsymbol{h}}_{\mathsf{c}} \,]^2\Big), \quad (3)$$

where Eq.(2) is due to decomposability (via Corollary 1), while Eq.(3) is due to the associativity (via Corollary 2), where the variable $s_i, \hat{s}_i$ (for $i \in [0, w]$)

in $\boldsymbol{s}, \hat{\boldsymbol{s}}$ are implicitly set as $s_i = \tilde{s}_i z_{T_i,1}$ and $\hat{s}_i = \tilde{s}_i z_{T_i,2}$, respectively. (Denote $T_0 = S_{\mathsf{c}}$). In particular, $s_0 = \tilde{s}_0 z_{S_{\mathsf{c}},1}$ and $\hat{s}_0 = \tilde{s}_0 z_{S_{\mathsf{c}},2}$, hence in $\tilde{C}_0$ we have $[\,\alpha\,]^{1,2}_{S_{\mathsf{k}}} \cdot [\,\tilde{s_0}\,]_{\emptyset} \cdot Z_{S_{\mathsf{c}}} = [\,\alpha s_0\,]^1_{[1,\kappa]} + [\,\alpha \hat{s}_0\,]^2_{[1,\kappa]}$. Hence, if $z_{i,2} = 0$ for all $i \in S_{\mathsf{c}}$, then CT is normal. Otherwise, $z_{i,2} \in_R \mathcal{R}$ for all $i \in S_{\mathsf{c}}$, then CT is semi-functional.

Guess. The algorithm $\mathcal{B}$ has properly simulated G_{real} if $z_{i,2} = 0$ for all $i \in S_{\mathsf{c}}$, and G_0 if $z_{i,2} \in_R \mathcal{R}$ for all $i \in S_{\mathsf{c}}$. Hence, $\mathcal{B}$ can use the output of $\mathcal{A}$ to break the (κ, S_{c})-MSD Assumption. □

Lemma 4 ($\mathsf{G}_{k,1}$ to $\mathsf{G}_{k,2}$). *For any adversary $\mathcal{A}$ against the $\mathsf{ABE}(\mathsf{P},\mathsf{G})$ scheme, there exists an algorithm $\mathcal{B}$ that breaks the $(1,1)$-CMH security of the pair encoding scheme P with $|\mathsf{G}_{k,1}\mathsf{Adv}^{\mathsf{ABE(P,G)}}_{\mathcal{A}}(\lambda) - \mathsf{G}_{k,2}\mathsf{Adv}^{\mathsf{ABE(P,G)}}_{\mathcal{A}}(\lambda)| \leq \mathsf{Adv}^{(1,1)\text{-}\mathsf{CMH}}_{\mathcal{B}}(\lambda)$.*

Proof. In the CMH game, $\mathcal{B}$ is given param and $I = \big\{[\,1\,]^1_S, [\,1\,]^2_S\big\}_{S \in \mathcal{S}_{\mathsf{c}} \cup \mathcal{S}_{\mathsf{k}}}$ from its challenger. It simulates $\mathsf{G}_{k,1}$ or $\mathsf{G}_{k,2}$ for $\mathcal{A}$ as follows.

$\mathcal{B}$ simulates $\mathsf{SFSetup}$ as follows. It generates $\mathsf{PK}, \mathsf{MSK}$ as in the construction but using the given I instead. Namely, $\mathcal{B}$ runs $\mathsf{P.Init}(\Lambda) \to \big(\kappa, \boldsymbol{h}_{\mathsf{c}}, \boldsymbol{h}_{\mathsf{k}}, n\big)$. It samples $[\,\alpha\,]_{\emptyset}, [\,h_1\,]_{\emptyset}, \ldots, [\,h_n\,]_{\emptyset} \leftarrow \mathsf{Samp}(\mathsf{param})$. By using I, $\mathcal{B}$ can then obtain $[\,\boldsymbol{h}_{\mathsf{c}}\,]^1, [\,\alpha\,]^1_{[1,\kappa]}$ for PK, and $[\,\boldsymbol{h}_{\mathsf{k}}\,]^1, [\,\alpha\,]^{1,2}_{S_{\mathsf{k}}}$ for MSK. It sends PK to $\mathcal{A}$. We remark that $[\,\hat{\boldsymbol{h}}_{\mathsf{c}}\,]^2, [\,\hat{\boldsymbol{h}}_{\mathsf{k}}\,]^2$ (as parts of $\widehat{\mathsf{PK}}, \widehat{\mathsf{MSK}}$) are not yet defined until the first query that requires using them, which is the k-th key query below.

Phase 1. When $\mathcal{A}$ makes the j-th key query for $X^{(j)}$, $\mathcal{B}$ does as follows.

(Case $j < k$). $\mathcal{B}$ samples $[\,\beta_j\,]_{\emptyset} \leftarrow \mathsf{Samp}(\mathsf{param})$, and computes a type-3 semi-functional key as $\mathsf{SK}_j \leftarrow \mathsf{SFKeyGen}(\mathsf{MSK}, X^{(j)}, 3, [\,1\,]^2_{S_{\mathsf{k}}}, [\,\beta_j\,]_{\emptyset})$.

(Case $j = k$). $\mathcal{B}$ generates a type-1 or type-2 semi-functional key as follows. $\mathcal{B}$ first obtains $\mathsf{KeyGen}\big(\mathsf{MSK}, X^{(k)}\big) \to \mathsf{SK} = ([\,\alpha\,]^{1,2}_{S_{\mathsf{k}}} + [\,K_0\,]^1_{S_{\mathsf{k}}}, [\,\boldsymbol{K}\,]^1)$. $\mathcal{B}$ then makes a key query for $X^{(k)}$ to its challenger in the CMH game and obtains

$$\widehat{\mathsf{SK}} = \big([\,\beta^\star\,]^2_{S_{\mathsf{k}}} + [\,\widehat{K_0}\,]^2_{S_{\mathsf{k}}}, [\,\widehat{\boldsymbol{K}}\,]^2\big).$$

This is the challenge for $\mathcal{B}$ to guess if $\beta^\star = 0$ or $\beta^\star \in_R \mathcal{R}$. $\mathcal{B}$ then returns $\mathsf{SK} + \widehat{\mathsf{SK}}$ to $\mathcal{A}$. If $\beta^\star = 0$, then this is a type-1 semi-functional key. If $\beta^\star \in_R \mathcal{R}$, then it is of type-2. We note that this simulated key implicitly defines $[\,\hat{\boldsymbol{h}}_{\mathsf{c}}\,]^2, [\,\hat{\boldsymbol{h}}_{\mathsf{k}}\,]^2$.

(Case $j > k$). $\mathcal{B}$ generates a normal key as $\mathsf{SK}_j \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, X^{(j)})$.

Challenge. The adversary $\mathcal{A}$ outputs messages $M_0, M_1 \in \{0,1\}^\lambda$ along with a target $Y^\star$ such that $R(X_j, Y^\star) = 0$ for all $j \in [1, q_1]$. $\mathcal{B}$ first obtains $[\,\boldsymbol{C}\,]^1$ by running $\mathsf{Encrypt}\big(\mathsf{PK}, Y, M\big)$. $\mathcal{B}$ then makes a ciphertext query for $Y^\star$ to its challenger in the CMH game and receives back $[\,\widehat{\boldsymbol{C}}\,]^2$. This query can be made since $R(X_k, Y^\star) = 0$. $\mathcal{B}$ parses $[\,s_0\,]^1_{S_{\mathsf{c}}}$ from $[\,\boldsymbol{C}\,]^1$, and $[\,\hat{s}_0\,]^2_{S_{\mathsf{c}}}$ from $[\,\widehat{\boldsymbol{C}}\,]^2$. $\mathcal{B}$ then chooses $\mathfrak{b} \xleftarrow{\$} \{0,1\}$ and computes $\widehat{C}_0 = \mathsf{G.Ext}(\mathsf{param}, [\,\alpha\,]_{\emptyset} \cdot [\,1\,]^1_{S_{\mathsf{k}}} \cdot [\,s_0\,]^1_{S_{\mathsf{c}}} + [\,\alpha\,]_{\emptyset} \cdot [\,1\,]^2_{S_{\mathsf{k}}} \cdot [\,\hat{s}_0\,]^2_{S_{\mathsf{c}}}) \oplus M_{\mathfrak{b}}$. $\mathcal{B}$ forms the challenge ciphertext as $\mathsf{CT} = \Big([\,\boldsymbol{C}\,]^1 + [\,\widehat{\boldsymbol{C}}\,]^2,\ \widehat{C}_0\Big)$, which is a properly distributed semi-functional ciphertext as required.

Phase 2. For each query in this phase, $\mathcal{B}$ generates a normal key as usual.

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

Variants of Security Theorems. We also obtain a theorem for the case of $(1,1)$-SMH, instead of $(1,\mathsf{poly})$-SMH. This results in looser reduction. We defer their proofs to the full version, where we also provide some more variants.

Corollary 3. *Suppose that a pair encoding* P *for predicate* R *is* $(1,1)$-CMH, $(1,1)$-SMH *in* G. *Suppose that the* MSD *Assumption holds in* G. *Then,* $\mathsf{ABE}(\mathsf{P},\mathsf{G})$ *is fully secure, with advantage bounded by*

$$\begin{aligned}\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ABE(P,G)}}(\lambda) \le \mathsf{Adv}_{\mathcal{B}_1}^{(\kappa,S_\mathsf{c})\text{-}\mathsf{MSD}}(\lambda) + 2q_{\mathrm{all}}\mathsf{Adv}_{\mathcal{B}_2}^{(\kappa,S_\mathsf{k})\text{-}\mathsf{MSD}}(\lambda)\\ + q_1\mathsf{Adv}_{\mathcal{B}_3}^{(1,1)\text{-}\mathsf{CMH(P)}}(\lambda) + q_2\mathsf{Adv}_{\mathcal{B}_4}^{(1,1)\text{-}\mathsf{SMH(P)}}(\lambda).\end{aligned}$$

On the other hand, we can establish tight reduction from semi-adaptive security to $(1,\mathsf{poly})$-SMH as shown in the following corollary.

Corollary 4. *Suppose that a pair encoding* P *for predicate* R *is* $(1,\mathsf{poly})$-SMH *in* G. *Suppose that the* MSD *Assumption holds in* G. *Then,* $\mathsf{ABE}(\mathsf{P},\mathsf{G})$ *is semi-adaptively secure, with advantage bounded by*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{semi,ABE(P,G)}}(\lambda) \le \mathsf{Adv}_{\mathcal{B}_1}^{(\kappa,S_\mathsf{c})\text{-}\mathsf{MSD}}(\lambda) + 2\mathsf{Adv}_{\mathcal{B}_2}^{(\kappa,S_\mathsf{k})\text{-}\mathsf{MSD}}(\lambda) + \mathsf{Adv}_{\mathcal{B}_3}^{(1,\mathsf{poly})\text{-}\mathsf{SMH(P)}}(\lambda).$$

4 Fully Secure KP-ABE for Circuits

We describe our first KP-ABE via multilinear pair encoding scheme $\mathsf{P}_{\mathsf{KPABE1}}$. It is based on the (selectively-secure) KP-ABE of GGHSW [24], albeit we require 3ℓ-multilinear maps, instead of $(\ell+1)$ as in [24]. More precisely, instead of using all singleton-set levels $\{1\},\dots,\{\ell+1\}$, we implement the scheme on encodings of levels in $\mathcal{S} := \{\,[1,\ell+1],[\ell+2,2\ell+1],\{2\ell+2\},\dots,\{3\ell\}\,\}$. In the construction, each of the first two "bundled" levels will always be used as a whole bundle. We only decompose them in the simulation to accommodate the assumption in the proof. Another difference are some additional terms T_1,T_2,D_1,D_2, for the purpose of proving the $\mathsf{CMH},\mathsf{SMH}$ security using randomizer techniques [3, 36].

Construction $\mathsf{P}_{\mathsf{KPABE1}}$.

- $\mathsf{Init}\big(\lambda,n,\ell\big) \to \big(\kappa,\boldsymbol{h}_\mathsf{c},\boldsymbol{h}_\mathsf{k},\bar{n}\big)$. Set $\kappa = 3\ell$. Set $\bar{n} = n+2$ where we use variables $h_1,\dots,h_n,\phi_1,\phi_2$. Let $\mathcal{S}' := \big\{[\ell+2,2\ell+1],\{2\ell+2\},\dots,\{3\ell\}\big\}$. Define

$$\begin{aligned}\boldsymbol{h}_\mathsf{c} &= \big((\,1\,)_{S_\mathsf{c}},(\,h_1\,)_{S_\mathsf{c}}\dots,(\,h_n\,)_{S_\mathsf{c}},(\,\phi_1\,)_{S_\mathsf{c}},(\,\phi_2\,)_{S_\mathsf{c}}\big),\\ \boldsymbol{h}_\mathsf{k} &= \big(\{(\,1\,)_S\}_{S\in\mathcal{S}'},(\,h_1\,)_{[\ell+2,2\ell+1]}\dots,(\,h_n\,)_{[\ell+2,2\ell+1]},(\,\phi_1\,)_{S_\mathsf{k}},(\,\phi_2\,)_{S_\mathsf{k}}\big),\end{aligned}$$

where $S_\mathsf{c} = [1,\ell+1], S_\mathsf{k} = [\ell+2,3\ell]$.

- $\mathsf{EncC}\big(\boldsymbol{h}_{\mathsf{c}}, x \in \{0,1\}^n\big) \to \big(\boldsymbol{C}, \boldsymbol{s}, \mathcal{P}_x\big)$.[12] Let $A_x = \{\, j \in [1,n] \mid x_j = 1 \,\}$. Output a ciphertext encoding $\boldsymbol{C} = \big(T_1, C, \{C_j\}_{j \in A_x}, T_2\big)$ where

$$T_1 = (\, t \,)_{[1,\ell+1]}, \quad C = (\, s \,)_{[1,\ell+1]}, \quad C_j = (\, h_j s \,)_{[1,\ell+1]}, \quad T_2 = (\, \phi_2 t + \phi_1 s \,)_{[1,\ell+1]}.$$

  The indexed variable vector is $\boldsymbol{s} = \big((\, t \,)_{[1,\ell+1]}, (\, s \,)_{[1,\ell+1]}\big)$. That is, the base randomness term is $(\, t \,)_{[1,\ell+1]}$.[13]
- $\mathsf{EncK}\big(\boldsymbol{h}_{\mathsf{k}}, f \in \mathbb{F}_{n,\ell}\big) \to \big((\, K_0 \,)_{S_{\mathsf{k}}}, \boldsymbol{K}, \boldsymbol{r}, \mathcal{P}_f\big)$.[12] Set the indexed variable vector:

$$\boldsymbol{r} = \Big((\, r \,)_{[\ell+2,3\ell]}, \big\{(\, \alpha_w \,)_{[\ell+2,2\ell+i_w]}\big\}_{w \in \mathsf{Nodes}}, \big\{(\, v_w \,)_{[\ell+2,2\ell+1]}\big\}_{w \in \mathsf{Inputs}}, \big\{(\, \ell_w \,)_{\{2\ell+i_w\}}, (\, r_w \,)_{\{2\ell+i_w\}}\big\}_{w \in \mathsf{Gates}}\Big)$$

  where we denote $i_w := \mathsf{Depth}(w)$. Define

$$D_1 = (\, \phi_2 r \,)_{[\ell+2,3\ell]}, \qquad D_2 = (\, r \,)_{[\ell+2,3\ell]}, \qquad D_3 = (\, \phi_1 r - \alpha_{w_{\mathsf{top}}} \,)_{[\ell+2,3\ell]}.$$

  Define the key element $\boldsymbol{K}_w$ for each $w \in \mathsf{Nodes}$ as follows.
  1. For each input node $w \in \mathsf{Inputs}$ (*i.e.,* $\mathsf{Depth}(w) = 1$), let $j = \mathsf{Num}(w)$. Define $\boldsymbol{K}_w = (U_w, K_w)$ as

$$U_w = (\, v_w \,)_{[\ell+2,2\ell+1]}, \qquad\qquad K_w = (\, \alpha_w + h_j v_w \,)_{[\ell+2,2\ell+1]}.$$

  2. For each gate $w \in \mathsf{Gates}$ (*i.e.,* $\mathsf{Depth}(w) > 1$), define $L_w = (\, \ell_w \,)_{\{2\ell+i_w\}}$, $R_w = (\, r_w \,)_{\{2\ell+i_w\}}$, and do as follows.
     - If $\mathsf{Type}(w) = \mathsf{OR}$, then set $\boldsymbol{K}_w = (L_w, R_w, K_{w,1}, K_{w,2})$, where we let

$$K_{w,1} = (\, \alpha_w + \alpha_{\mathsf{L}(w)} \ell_w \,)_{[\ell+2,2\ell+i_w]}, \quad K_{w,2} = (\, \alpha_w + \alpha_{\mathsf{R}(w)} r_w \,)_{[\ell+2,2\ell+i_w]}.$$

     - If $\mathsf{Type}(w) = \mathsf{AND}$, then we set $\boldsymbol{K}_w = (L_w, R_w, K_w)$, where we let

$$K_w = (\, \alpha_w + \alpha_{\mathsf{L}(w)} \ell_w + \alpha_{\mathsf{R}(w)} r_w \,)_{[\ell+2,2\ell+i_w]}.$$

  Output the key encoding as $((\, K_0 \,)_{S_{\mathsf{k}}}, \boldsymbol{K})$ where the master-key masking term is $(\, K_0 \,)_{S_{\mathsf{k}}} = D_1$ and the rest is $\boldsymbol{K} = \big(D_2, D_3, \{\boldsymbol{K}_w\}_{w \in \mathsf{Nodes}}\big)$.
- $\mathsf{Pair}(f, x) \to \mathcal{P}_{f,x}$. Assume $f(x) = 1$. We describe multilinear program $\mathcal{P}_{f,x}$ that takes $(\boldsymbol{C}, \boldsymbol{K})$ as an input, and outputs $(\, K_0 t \,)_{[1,3\ell]}$. It computes at each node w such that $f_w(x) = 1$ in the bottom-up manner. It will derive $E_w := (\, \alpha_w s \,)_{[1,2\ell+i]}$, where $i = \mathsf{Depth}(w)$. We show this by induction on i (1 to ℓ).
  1. For each input node $w \in \mathsf{Inputs} = [1,n]$ such that $f_w(x) = 1$, we have $x_w = 1$ and $j := \mathsf{Num}(w) \in A_x$. Compute

$$E_w = C \cdot K_w - C_j \cdot U_w = (\, \alpha_w s \,)_{[1,2\ell+1]}.$$

     This effectively proves the base case of the induction.

[12]The multilinear programs $\mathcal{P}_x$ output from EncC and $\mathcal{P}_f$ output from EncK are straightforwardly deducible from the respective encodings.

[13]That is, we use variable t in place of s_0 of the generic construction.

2. For each gate $w \in \mathsf{Gates}$ such that $f_w(x) = 1$, we have two cases.
   - If $\mathsf{Type}(w) = \mathsf{OR}$, then $f_{\mathsf{L}(w)}(x) = 1$ or $f_{\mathsf{R}(w)}(x) = 1$. Wlog, we can assume that $f_{\mathsf{L}(w)}(x) = 1$. Hence, $E_{\mathsf{L}(w)} = (\,\alpha_{\mathsf{L}(w)}s\,)_{[1,2\ell+i-1]}$ by the induction hypothesis, as $\mathsf{Depth}(\,\mathsf{L}(w)) = i-1$. Then, compute
   $$E_w = C \cdot K_{w,1} - E_{\mathsf{L}(w)} \cdot L_w = (\,\alpha_w s\,)_{[1,2\ell+i]}.$$
   - If $\mathsf{Type}(w) = \mathsf{AND}$, then $f_{\mathsf{L}(w)}(x) = 1$ and $f_{\mathsf{R}(w)}(x) = 1$. Hence, $E_{\mathsf{L}(w)} = (\,\alpha_{\mathsf{L}(w)}s\,)_{[1,2\ell+i-1]}$, $E_{\mathsf{R}(w)} = (\,\alpha_{\mathsf{R}(w)}s\,)_{[1,2\ell+i-1]}$, by the induction hypothesis. Then, compute
   $$E_w = C \cdot K_w - \big(E_{\mathsf{L}(w)} \cdot L_w + E_{\mathsf{R}(w)} \cdot R_w\big) = (\,\alpha_w s\,)_{[1,2\ell+i]}.$$

   This concludes the induction. Finally, at the top gate w_{top}, where $\mathsf{Depth}(w_{\mathsf{top}}) = \ell$, we obtain $E_{w_{\mathsf{top}}} = (\,\alpha_{w_{\mathsf{top}}}s\,)_{[1,3\ell]}$. Compute and obtain
   $$T_2 \cdot D_2 - E_{w_{\mathsf{top}}} - C \cdot D_3 = (\,K_0 t\,)_{[1,3\ell]},$$
   as required.

Properties. We can see that the key encoding for circuit f contains (at most) $2n + 4g' + 3$ elements, where g' is the number of internal gates. Hence it admits succinctness (the size is $O(g)$, where $g = n + g'$ is the size of a circuit). The ciphertext encoding for x contains $|A_x| + 3 \le n + 3$ elements. Moreover, it has no bound on circuit size and fan-out. We only require bounds on input length n and depth ℓ.

Assumptions. We describe two new assumptions, $\mathsf{SMDDH1}$ and $\mathsf{EMDDH1}$, which extend the regular Multi-linear DDH assumption (MDDH) [15, 23, 20] in asymmetric setting. (S, E is for Simple/Esoteric extension, resp.) For assumption X, we define the advantage $\mathsf{Adv}^{\mathsf{X}}_{\mathcal{A}}(\lambda) := |\Pr[\mathcal{A}(D,Z) = 1] - \Pr[\mathcal{A}(D,Z') = 1]|$, for adversary $\mathcal{A}$, where D, Z, Z' are specified in each assumption.

Definition 7 (ℓ-$\mathsf{SMDDH1}$). *Let* $\mathsf{InstGen}(1^\lambda, 3\ell, 2) \to (\mathsf{param}, \mathsf{esk})$. *Sample* $\zeta, z, c_1, \ldots, c_{\ell+1}$, *from* $\mathcal{R}$. *The* ℓ-$\mathsf{SMDDH1}$ *Assumption states that the following distributions are computationally indistinguishable:*
$$\Big(D, Z = [\,c_1 \cdots c_{\ell+1} z\,]^2_{[\ell+2,3\ell]}\Big) \qquad \textit{and} \qquad \Big(D, Z' = [\,\zeta\,]^2_{[\ell+2,3\ell]}\Big),$$
where D consists of: param, $\Big\{[\,1\,]^1_{\{i\}}, [\,1\,]^2_{\{i\}}\Big\}_{i\in[1,3\ell]}$, $[\,z\,]^2_{[1,\ell+1]}, [\,c_1 z\,]^2_{[\ell+2,3\ell]}$,
$$[\,c_1\,]^2_{[1,\ell+1]}, [\,c_1\,]^2_{[\ell+2,2\ell+1]}, [\,c_1\,]^2_{\{2\ell+2\}}, \ldots, [\,c_1\,]^2_{\{3\ell\}},$$
$$[\,c_2\,]^2_{[\ell+2,2\ell+1]}, [\,c_3\,]^2_{\{2\ell+2\}}, \ldots, [\,c_{\ell+1}\,]^2_{\{3\ell\}}.$$

$\mathsf{SMDDH1}$ differs from MDDH (in asymmetric settings) in two aspects. First, the target element is in the level $[\ell+2, 3\ell]$, instead of the whole, which is $[1, 3\ell]$. Second, it gives out one more element $[\,c_1 z\,]^2_{[\ell+2,3\ell]}$. We can see that this would not help attacking since it cannot be multiplied with available $c_2, \ldots, c_\ell$ as they are all encoded in levels that are subsets of $[\ell+2, 3\ell]$.

Definition 8 **((ℓ, m)-$\mathsf{EMDDH1}$).** *Let* $\mathsf{InstGen}(1^\lambda, 3\ell, 2) \to (\mathsf{param}, \mathsf{esk})$. *Sample* $b, z, v, c_1, \cdots, c_{\ell+1}, \mu_1, \cdots, \mu_\ell, \nu_1, \cdots, \nu_\ell, \omega_1, \cdots, \omega_\ell, \{a_{i,j}, d_{i,j}\}_{i\in[1,\ell], j\in[1,m]}$, *and* ζ *from* $\mathcal{R}$. *Denote* $\mu = \mu_1 \cdots \mu_\ell, \nu = \nu_1 \cdots \nu_\ell, \omega = \omega_1 \cdots \omega_\ell$. *The* (ℓ, m)-$\mathsf{EMDDH1}$ *Assumption states that the following distributions are computationally indistinguishable:*

$$\Big(D, Z = [\, c_1 \cdots c_{\ell+1} b \,]^2_{[\ell+2, 3\ell]}\Big) \qquad \text{and} \qquad \Big(D, Z' = [\, \zeta \,]^2_{[\ell+2, 3\ell]}\Big),$$

where D *consists of*[14]: param, $\big\{[\,1\,]^1_S, [\,1\,]^2_S\big\}_{S\in\mathcal{S}}$, $[\,\frac{z}{b}\,]^2_{[1,\ell+1]}, [\,v\,]^2_{[1,\ell+1]}, [\,v\,]^2_{[\ell+2,3\ell]}$, $[\,vb\,]^2_{[\ell+2,3\ell]}, [\,\frac{c_1\cdots c_{\ell+1}}{v}\,]^2_{[\ell+2,3\ell]}$, *and*

$$\begin{array}{rlrl}
\forall_{e\in\{0,-1\}} & [\,\mu_i a^e_{i,j}\,]^2_{\{i\}}, & & [\,\dfrac{z}{\mu}\,]^2_{\{\ell+1\}}, \\
\forall_{e\in\{0,1\}} & [\,\nu_i a^e_{i,j} d_{i,j}\,]^2_{\{i\}}, & & [\,\dfrac{c_1}{\nu}\,]^2_{\{\ell+1\}}, \\
\forall_{e\in\{0,-1\}} & [\,\omega_i a^e_{i,j}\,]^2_{\{i\}}, & & [\,\dfrac{\omega_i}{\omega} zv \dfrac{1}{a_{i,j}}\,]^2_{\{i,\ell+1\}}, \\
\forall_{(e,e')\in\mathsf{E}} & [\,\dfrac{a^e_{i,j}}{a^{e'}_{i,j'}} d_{i,j}\,]^2_{\{i\}}, & \forall_{(e,e')\in\mathsf{E}^\star} & [\,zc_1\dfrac{a^e_{i,j}}{a^{e'}_{i,j'}} d_{i,j}\,]^2_{\{i,\ell+1\}}, \\
 & [\,\dfrac{c_2}{d_{1,j}}\,]^2_{[\ell+2,2\ell+1]}, & \forall_{i\in[2,\ell]} & [\,\dfrac{c_{i+1}}{d_{i,j}}\,]^2_{\{2\ell+i\}}, \\
\forall_{i\in[2,\ell]}\forall_{e\in\{0,1\}} & [\,a^e_{i,j} d_{i,j}\,]^2_{\{\ell+1+i\}}, & & \\
\forall_{e\in\{0,1\}} & [\,c_1\cdots c_i a^e_{i,j} d_{i,j}\,]^2_{S_i}, & \forall_{e\in\{0,1\}} & [\,c_1\cdots c_{i+1} a^e_{i,j}\dfrac{d_{i,j}}{d_{i,j'}}\,]^2_{S_i},
\end{array}$$

where, unless stated above, subscripts range for all $i \in [1, \ell]$, $j, j' \in [1, m]$ *such that* $j' \neq j$. *Denote* $\mathsf{E} = \{(0, 0), (0, 1), (1, 0), (1, 1), (-1, 0)\}$; $\mathsf{E}^\star = \mathsf{E} \setminus \{(0, 0)\}$. *Denote* $S_1 = \{\ell + 2\}$ *and* $S_i = [\ell + 2, \ell + 1 + i] \cup [2\ell + 2, 2\ell + i]$ *for* $i \geq 2$.

Due to the lack of space, we defer the intuition, some remark, and its generic hardness for $\mathsf{EMDDH1}$ to the full version. We provide some discussions regarding $\mathsf{EMDDH1}$ as follows.

On Assumption Simplicity. To compare simplicity of assumptions *quantitatively*, we measure their sizes. The size of $\mathsf{EMDDH1}$ is $O(\ell m^2)$. In bilinear groups, we already have the Expanded m-BDHE [49] assumption, or the one in [43], of which size is $O(m^2)$. The expansion factor of $O(\ell)$ in ours is somewhat natural since we extend to 3ℓ-linear maps. Indeed, the most basic assumption for ℓ-linear maps, namely, the normal ℓ-MDDH [15, 23, 20], already has size $\Omega(\ell)$.

Comparing to Uber Assumption. The Uber Assumptions in multilinear settings (Uber) are introduced in [42, 38], for proving their IO schemes. Intuitively, Uber assumes the indistinguishability of (D, Z) and (D, Z') for *all non-trivial triples* of (D, Z, Z'). We compare $\mathsf{EMDDH1}$ to Uber as they share this similar

[14]We refer the definition of $\mathcal{S}$ to the beginning of this section (§4).

intuition. However, contrastingly to Uber, $\mathsf{EMDDH1}$ requires only *one such specific triple*, parameterized by (ℓ, m). Our scheme could possibly be proved as well under Uber, so that new assumptions would not be needed. However, this would be undesirable since Uber is *not* efficiently falsifiable [42]; while, on the other hand, our assumptions are. In other words, we believe that it is important to come up with such a specific triple, even if it might look complex. Indeed, our novelty exactly lies in identifying such an explicit triple (D, Z, Z') defined for $\mathsf{EMDDH1}$.

Security. We now state the security theorems for our encoding $\mathsf{P}_{\mathsf{KPABE1}}$. Their proofs are deferred to the full version. From these and Theorem 3, Corollary 4, we also obtain the full and semi-adaptive security of our first KP-ABE below.

Theorem 4. $\mathsf{P}_{\mathsf{KPABE1}}$ *is* $(1,1)$-CMH *under the* (ℓ, m)-$\mathsf{EMDDH1}$ *assumption with tight reduction, where* ℓ, m *is the bounded depth and the width of queried circuit.*

Theorem 5. $\mathsf{P}_{\mathsf{KPABE1}}$ *is* $(1, \mathsf{poly})$-SMH *under the* ℓ-$\mathsf{SMDDH1}$ *assumption with tight reduction, where* ℓ *is the bounded depth.*

Corollary 5. $\mathsf{ABE}(\mathsf{P}_{\mathsf{KPABE1}}, \mathsf{G})$ *is fully secure under* $\mathsf{EMDDH1}, \mathsf{SMDDH1}, \mathsf{MSD}$*, and semi-adaptively secure under* $\mathsf{SMDDH1}, \mathsf{MSD}$*, with advantage bounded by*

$$\begin{aligned}\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ABE}(\mathsf{P}_{\mathsf{KPABE1}},\mathsf{G})}(\lambda) \leq & \mathsf{Adv}_{\mathcal{B}_1}^{(\kappa,S_\mathsf{c})\text{-}\mathsf{MSD}}(\lambda) + (2q_1+2)\mathsf{Adv}_{\mathcal{B}_2}^{(\kappa,S_\mathsf{k})\text{-}\mathsf{MSD}}(\lambda) \\ & + q_1 \mathsf{Adv}_{\mathcal{B}_3}^{(\ell,m)\text{-}\mathsf{EMDDH1}}(\lambda) + \mathsf{Adv}_{\mathcal{B}_4}^{\ell\text{-}\mathsf{SMDDH1}}(\lambda), \\ \mathsf{Adv}_{\mathcal{A}}^{\mathsf{semi},\mathsf{ABE}(\mathsf{P}_{\mathsf{KPABE1}},\mathsf{G})}(\lambda) \leq & \mathsf{Adv}_{\mathcal{B}'_1}^{(\kappa,S_\mathsf{c})\text{-}\mathsf{MSD}}(\lambda) + 2\mathsf{Adv}_{\mathcal{B}'_2}^{(\kappa,S_\mathsf{k})\text{-}\mathsf{MSD}}(\lambda) + \mathsf{Adv}_{\mathcal{B}'_3}^{\ell\text{-}\mathsf{SMDDH1}}(\lambda),\end{aligned}$$

where $\kappa = 3\ell$*,* $S_\mathsf{c} = [1, \ell+1]$*,*$S_\mathsf{k} = [\ell+2, 3\ell]$.

5 Fully Secure KP-ABE with Short Ciphertext

We describe our KP-ABE for circuits with short ciphertexts. We use similar techniques from compact ABE for formulae of [8, 3], which are also similar to [13, 19], for designing elements related to the input layer of circuits. The mechanism regarding internal gates of circuits are exactly the same as our first KP-ABE.

Construction $\mathsf{P}_{\mathsf{KPABE2}}$.

- $\mathsf{Init}\big(\lambda, n, \ell\big) \to \big(\kappa, \boldsymbol{h}_\mathsf{c}, \boldsymbol{h}_\mathsf{k}, \bar{n}\big)$. Set $\kappa = 3\ell$. Set $\bar{n} = n+4$ where we use variables $h_0, h_1, \ldots, h_n, \phi_1, \phi_2, \phi_3$. Define $\boldsymbol{h}_\mathsf{c}$ and $\boldsymbol{h}_\mathsf{k}$ as in our first KP-ABE except that we have one additional term for each: $(\,\phi_3\,)_{[1,\ell+1]}$ in $\boldsymbol{h}_\mathsf{c}$; $(\,\phi_3\,)_{[\ell+2,2\ell+1]}$ in $\boldsymbol{h}_\mathsf{k}$.
- $\mathsf{EncC}\big(\boldsymbol{h}_\mathsf{c}, x \in \{0,1\}^n\big) \to \big(\boldsymbol{C}, \boldsymbol{s}, \mathcal{P}_x\big)$. Let $A_x = \{\, j \in [1,n] \mid x_j = 1 \,\}$. Output a ciphertext encoding $\boldsymbol{C} = \big(C, C_1, C_2, T_1, T_2\big)$ where

$$C = (\,s\,)_{[1,\ell+1]}, \quad C_1 = (\,\phi_3 s + (h_0 + \sum_{j \in A_x} h_j) u\,)_{[1,\ell+1]}, \quad C_2 = (\,u\,)_{[1,\ell+1]}$$

$$T_1 = (\,t\,)_{[1,\ell+1]}, \quad T_2 = (\,\phi_2 t + \phi_1 s\,)_{[1,\ell+1]}.$$

The indexed variable vector is $\boldsymbol{s} = \Big((\,t\,)_{[1,\ell+1]}, (\,s\,)_{[1,\ell+1]}, (\,u\,)_{[1,\ell+1]}\Big)$.

- $\mathsf{EncK}\big(\boldsymbol{h}_{\mathsf{k}}, f \in \mathbb{F}_{n,\ell}\big) \to \big((\, K_0\,)_{S_{\mathsf{k}}}, \boldsymbol{K}, \boldsymbol{r}, \mathcal{P}_f\big)$. All the elements are the same as our first KP-ABE except $\{\boldsymbol{K}_w\}_{w\in\mathsf{Inputs}}$. Let $j = \mathsf{Num}(w)$. We define $\boldsymbol{K}_w = (U_w, K_w, F_w, \{G_{w,i}\}_{i\in[1,n]\smallsetminus\{j\}})$ as

$$U_w = (\, v_w\,)_{[\ell+2,2\ell+1]}, \qquad K_w = (\, \alpha_w + \phi_3 v_w\,)_{[\ell+2,2\ell+1]},$$
$$F_w = (\, (h_0 + h_j)v_w\,)_{[\ell+2,2\ell+1]}, \qquad G_{w,i} = (\, h_i v_w\,)_{[\ell+2,2\ell+1]}.$$

- $\mathsf{Pair}(f,x) \to \mathcal{P}_{f,x}$. Assume that $f(x) = 1$. We describe the multilinear program $\mathcal{P}_{f,x}$ that takes $(\boldsymbol{C}, \boldsymbol{K})$ as an input and outputs $(\, K_0 t\,)_{[1,3\ell]}$. It computes exactly as in our first KP-ABE except the computation regarding input nodes. For each input node $w \in \mathsf{Inputs} = [1,n]$ such that $f_w(x) = 1$, we have $x_w = 1$. Let $j = \mathsf{Num}(w)$. We have $j \in A_x$. We compute:

$$E_w = C \cdot K_w - C_1 \cdot U_w + C_2 \cdot F_w + C_2 \cdot \sum_{i\in A_x\smallsetminus\{j\}} G_{w,i} = (\, \alpha_w s\,)_{[1,2\ell+1]}$$

The rest of algorithm is defined as in $\mathsf{P}_{\mathsf{KPABE1}}$.

Properties. We can see that the ciphertext encoding for string x always contains 5 elements (hence constant-size relative to n). The key encoding for circuit f contains $n(n-1) + 4g' + 6$ elements, where g' is the number of internal gates.

Assumptions. We use new assumptions SMDDH2, EMDDH2, which are similar to SMDDH1, EMDDH1 respectively, albeit with some additional terms that will be used for simulating the new input layer. In particular, SMDDH2 consists of terms that are similar to the BDHE [14] and the Multi-linear BDHE [13] assumptions (the terms of the form $g, g^a, \ldots, g^{a^n}, g^{a^{n+2}}, \ldots, g^{a^{2n}}$), depicted in the last line of SMDDH2 below. Again, we prove their generic hardness in the full version.

Definition 9 **((ℓ,n)-SMDDH2).** *Let* $\mathsf{InstGen}(1^\lambda, 3\ell, 2) \to (\mathsf{param}, \mathsf{esk})$. *Sample* $\zeta, z, c_1, \ldots, c_{\ell+1}, b$ *from* $\mathcal{R}$. *Let* $S = [\ell{+}2, 2\ell{+}1]$. *The* (ℓ,n)-SMDDH2 *Assumption states that the following distributions are computationally indistinguishable:*

$$\Big(D, Z = [\, c_1^{n+1} c_2 \cdots c_{\ell+1} b\,]^2_{[\ell+2,3\ell]}\Big) \qquad \text{and} \qquad \Big(D, Z' = [\, \zeta\,]^2_{[\ell+2,3\ell]}\Big),$$

where D consists of: param, $\Big\{[\,1\,]^1_{\{i\}}, [\,1\,]^2_{\{i\}}\Big\}_{i\in[1,3\ell]}$, *and*

$$\begin{aligned}
&[\, z\,]^2_{[1,\ell+1]}, [\, \tfrac{z}{b}\,]^2_{[1,\ell+1]}, [\, c_1^{n+1} b\,]^2_{[\ell+2,3\ell]},\\
&[\, c_1^{n+1}\,]^2_{\{2\ell+2\}}, \ldots, [\, c_1^{n+1}\,]^2_{\{3\ell\}},\\
&[\, c_2\,]^2_{[\ell+2,2\ell+1]}, [\, c_3\,]^2_{\{2\ell+2\}}, \ldots, [\, c_{\ell+1}\,]^2_{\{3\ell\}},\\
&[\, c_1\,]^2_{[1,\ell+1]}, \ldots, [\, c_1^{n+1}\,]^2_{[1,\ell+1]},\ [\, c_1\,]^2_S, \ldots, [\, c_1^{n+1}\,]^2_S\\
&[\, c_1 c_2\,]^2_S, \ldots, [\, c_1^n c_2\,]^2_S, [\, c_1^{n+2} c_2\,]^2_S, \ldots, [\, c_1^{2n+1} c_2\,]^2_S.
\end{aligned}$$

Definition 10 **((ℓ, m, n)-$\mathsf{EMDDH2}$).** *The (ℓ, m, n)-$\mathsf{EMDDH2}$ is defined in exactly the same manner as (ℓ, m)-$\mathsf{EMDDH1}$ except that the given part D contains also additional elements as follows. The problem instance additionally samples b_j for $j \in [1, n]$. It augments D to also contain, for $j, j' \in [1, n]$ such that $j \neq j'$,*

$$[\,\mu_1 b_j\,]^2_{\{1\}},\ [\,\nu_1 b_j^2\,]^2_{\{1\}},\ [\,\frac{1}{b_j}\,]^2_{\{1\}},\ [\,\frac{z c_1 b_j}{b_{j'}^2}\,]^2_{\{1,\ell+1\}},$$

$$[\,b_j c_2\,]^2_{[\ell+2,2\ell+1]},\ [\,\frac{c_1}{b_j}\,]^2_{\{\ell+2\}},\ [\,\frac{c_1}{b_j^2}\,]^2_{\{\ell+2\}},\ [\,\frac{c_1 c_2 b_j}{b_{j'}}\,]^2_{\{\ell+2\}},\ [\,\frac{c_1 c_2 b_j}{b_{j'}^2}\,]^2_{\{\ell+2\}}.$$

Security. We now state the security theorems for $\mathsf{P}_{\mathsf{KPABE2}}$. We prove them in the full version. The full/semi-adaptive security of the resulting ABE is also given below.

Theorem 6. *$\mathsf{P}_{\mathsf{KPABE2}}$ is $(1, 1)$-CMH under the (ℓ, m, n)-$\mathsf{EMDDH2}$ assumption with tight reduction, where ℓ is the bounded depth, n is the input length, and m is the width of the queried circuit.*

Theorem 7. *$\mathsf{P}_{\mathsf{KPABE2}}$ is $(1, \mathsf{poly})$-SMH under the (ℓ, n)-$\mathsf{SMDDH2}$ assumption with tight reduction, where ℓ is the bounded depth and n is the input length.*

Corollary 6. *$\mathsf{ABE}(\mathsf{P}_{\mathsf{KPABE2}}, \mathsf{G})$ is fully secure under $\mathsf{EMDDH2}, \mathsf{SMDDH2}, \mathsf{MSD}$, and semi-adaptively secure under $\mathsf{SMDDH2}, \mathsf{MSD}$, with advantage bounded by*

$$\begin{aligned}\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ABE}(\mathsf{P}_{\mathsf{KPABE2}},\mathsf{G})}(\lambda) \leq & \mathsf{Adv}_{\mathcal{B}_1}^{(\kappa,S_\mathsf{c})\text{-}\mathsf{MSD}}(\lambda) + (2q_1+2)\mathsf{Adv}_{\mathcal{B}_2}^{(\kappa,S_\mathsf{k})\text{-}\mathsf{MSD}}(\lambda) \\ & + q_1 \mathsf{Adv}_{\mathcal{B}_3}^{(\ell,m,n)\text{-}\mathsf{EMDDH2}}(\lambda) + \mathsf{Adv}_{\mathcal{B}_4}^{(\ell,n)\text{-}\mathsf{SMDDH2}}(\lambda), \\ \mathsf{Adv}_{\mathcal{A}}^{\mathsf{semi},\mathsf{ABE}(\mathsf{P}_{\mathsf{KPABE2}},\mathsf{G})}(\lambda) \leq & \mathsf{Adv}_{\mathcal{B}'_1}^{(\kappa,S_\mathsf{c})\text{-}\mathsf{MSD}}(\lambda) + 2\mathsf{Adv}_{\mathcal{B}'_2}^{(\kappa,S_\mathsf{k})\text{-}\mathsf{MSD}}(\lambda) + \mathsf{Adv}_{\mathcal{B}'_3}^{(\ell,n)\text{-}\mathsf{SMDDH2}}(\lambda),\end{aligned}$$

where $\kappa = 3\ell$, $S_\mathsf{c} = [1, \ell+1]$,$S_\mathsf{k} = [\ell+2, 3\ell]$.

6 Dual Conversion and CP-ABE

In this section, we provide a generic dual conversion for multilinear pair encoding. It uses essentially the same idea as the dual conversion for bilinear pair encoding of [9]. We then apply it to our KP-ABE and obtain CP-ABE for circuits.

6.1 Generic Dual Conversion

Given a multi-linear pair encoding scheme P for predicate R, we construct a scheme $\textsc{Con}(\mathsf{P})$ for its dual predicate $\bar{R}$ as follows. We also denote $\overline{\mathsf{P}} = \textsc{Con}(\mathsf{P})$.

- $\overline{\mathsf{P}}.\mathsf{Init}(\Lambda)$: Run $\mathsf{P}.\mathsf{Init}(\Lambda) \to \big(\kappa, \boldsymbol{h}_\mathsf{c}, \boldsymbol{h}_\mathsf{k}, n\big)$. Parse $S_\mathsf{c}, S_\mathsf{k}$ from $\boldsymbol{h}_\mathsf{c}, \boldsymbol{h}_\mathsf{k}$. Let

  $$\bar{S}_\mathsf{c} := S_\mathsf{k}, \qquad \bar{S}_\mathsf{k} := S_\mathsf{c}, \qquad \overline{\boldsymbol{h}}_\mathsf{c} := (\boldsymbol{h}_\mathsf{k}, (\,\phi\,)_{\bar{S}_\mathsf{c}}), \qquad \overline{\boldsymbol{h}}_\mathsf{k} := (\boldsymbol{h}_\mathsf{c}, (\,\phi\,)_{\bar{S}_\mathsf{k}}),$$

  where ϕ is a fresh variable. Output $\big(\kappa, \overline{\boldsymbol{h}}_\mathsf{c}, \overline{\boldsymbol{h}}_\mathsf{k}, n+1\big)$.

- $\overline{\mathsf{P}}.\mathsf{EncK}(Y, \overline{\boldsymbol{h}}_{\mathsf{k}})$: Parse $\boldsymbol{h}_{\mathsf{c}}$ from $\overline{\boldsymbol{h}}_{\mathsf{k}}$. Run $\mathsf{P}.\mathsf{EncC}(Y, \boldsymbol{h}_{\mathsf{c}}) \to \big(\boldsymbol{C}, \boldsymbol{s}, \mathcal{P}_Y\big)$. Define
$$\bar{K}_0 := \phi s_0, \qquad \overline{\boldsymbol{K}} := \boldsymbol{C}, \qquad \bar{\boldsymbol{r}} := \boldsymbol{s}.$$
Define $\overline{\mathcal{P}}_Y$ exactly as $\mathcal{P}_Y$ (which outputs $\boldsymbol{C}$) but with an additional input $(\, \phi s_0 \,)_{\bar{S}_{\mathsf{k}}}$, which is trivially wired to output $(\, \bar{K}_0 \,)_{\bar{S}_{\mathsf{k}}}$. Output $\big((\, \bar{K}_0 \,)_{\bar{S}_{\mathsf{k}}}, \overline{\boldsymbol{K}}, \bar{\boldsymbol{r}}, \overline{\mathcal{P}}_Y\big)$.
- $\overline{\mathsf{P}}.\mathsf{EncC}(X, \overline{\boldsymbol{h}}_{\mathsf{c}})$: Parse $\boldsymbol{h}_{\mathsf{k}}$ from $\overline{\boldsymbol{h}}_{\mathsf{c}}$. Run $\mathsf{P}.\mathsf{EncK}(X, \boldsymbol{h}_{\mathsf{k}}) \to \big((\, K_0 \,)_{S_{\mathsf{k}}}, \boldsymbol{K}, \boldsymbol{r}, \mathcal{P}_X\big)$. Define
$$\overline{\boldsymbol{C}} := \big((\, \bar{s}_0 \,)_{\bar{S}_{\mathsf{c}}}, (\, \phi\bar{s}_0 + K_0 \,)_{\bar{S}_{\mathsf{c}}}, \boldsymbol{K}\big), \qquad \bar{\boldsymbol{s}} := \big((\, \bar{s}_0 \,)_{\bar{S}_{\mathsf{c}}}, \boldsymbol{r}\big).$$
where $\bar{s}_0$ is a fresh variable. Define $\overline{\mathcal{P}}_X$ exactly as $\mathcal{P}_X$ (which is a program that outputs $((\, K_0 \,)_{S_{\mathsf{k}}}, \boldsymbol{K})$) but with additional inputs $(\, \bar{s}_0 \,)_{\bar{S}_{\mathsf{c}}}, (\, \phi\bar{s}_0 \,)_{\bar{S}_{\mathsf{c}}}$, which is used for the two new output elements in $\overline{\boldsymbol{C}}$. Output $\big(\overline{\boldsymbol{C}}, \bar{\boldsymbol{s}}, \overline{\mathcal{P}}_X\big)$.
- $\overline{\mathsf{P}}.\mathsf{Pair}(Y, X)$: Run $\mathsf{P}.\mathsf{Pair}(X, Y) \to \mathcal{P}_{X,Y}$. Define program $\overline{\mathcal{P}}_{Y,X}$ as:
$$\overline{\mathcal{P}}_{Y,X}(\overline{\boldsymbol{K}}, \overline{\boldsymbol{C}}): \text{ Output } (\, \phi\bar{s}_0 + K_0 \,)_{\bar{S}_{\mathsf{c}}}(\, s_0 \,)_{\bar{S}_{\mathsf{k}}} - \mathcal{P}_{X,Y}(\boldsymbol{K}, \boldsymbol{C}).$$
Note that $(\, \phi\bar{s}_0 + K_0 \,)_{\bar{S}_{\mathsf{c}}}$ and $\boldsymbol{K}$ are parsed from $\overline{\boldsymbol{C}}$, while $(\, s_0 \,)_{\bar{S}_{\mathsf{k}}}$ is parsed from the first element of $\overline{\boldsymbol{K}} = \boldsymbol{C}$. Outputs the description of $\overline{\mathcal{P}}_{Y,X}$.

Correctness. Assume $\bar{R}(Y, X) = 1$. Hence, $R(X, Y) = 1$. From the correctness of P, we have $\mathcal{P}_{X,Y}(\boldsymbol{K}, \boldsymbol{C}) = (\, K_0 s_0 \,)_{[1,\kappa]}$. Hence
$$\begin{aligned}\overline{\mathcal{P}}_{Y,X}(\overline{\boldsymbol{K}}, \overline{\boldsymbol{C}}) &= (\, \phi\bar{s}_0 + K_0 \,)_{\bar{S}_{\mathsf{c}}}(\, s_0 \,)_{\bar{S}_{\mathsf{k}}} - (\, K_0 s_0 \,)_{[1,\kappa]} \\ &= (\, \phi\bar{s}_0 \,)_{\bar{S}_{\mathsf{c}}}(\, s_0 \,)_{\bar{S}_{\mathsf{k}}} = (\, \bar{s}_0 \,)_{\bar{S}_{\mathsf{c}}}(\, \phi s_0 \,)_{\bar{S}_{\mathsf{k}}} = (\, \bar{K}_0 \bar{s}_0 \,)_{[1,\kappa]},\end{aligned}$$
as required. We must also verify the associativity of $\overline{\mathcal{P}}_Y$ over $(\bar{\boldsymbol{r}}, \overline{\boldsymbol{h}}_{\mathsf{k}})$, and of $\overline{\mathcal{P}}_X$ over $(\bar{\boldsymbol{s}}, \overline{\boldsymbol{h}}_{\mathsf{c}})$. But these are straightforward due to the associativity of $\mathcal{P}_Y$ over $(\boldsymbol{s}, \boldsymbol{h}_{\mathsf{c}})$, and of $\mathcal{P}_X$ over $(\boldsymbol{r}, \boldsymbol{h}_{\mathsf{k}})$, and the new elements can be easily inspected, in particular, $(\, \bar{s}_0 \,)_{\bar{S}_{\mathsf{c}}}(\, 1 \,)_{\emptyset} = (\, \bar{s}_0 \,)_{\emptyset}(\, 1 \,)_{\bar{S}_{\mathsf{c}}}$, and we have $(\, 1 \,)_{\bar{S}_{\mathsf{c}}} \in \boldsymbol{h}_{\mathsf{k}}$.

The following lemma shows that the conversion preserves security (in an alternating manner). The proof is similar to [9], and is given in the full version.

Lemma 5. $(1,1)$-CMH *security of* P *implies* $(1,1)$-SMH *security of* $\overline{\mathsf{P}}$. *Oppositely,* $(1,1)$-SMH *security of* P *implies* $(1,1)$-CMH *security of* $\overline{\mathsf{P}}$.

6.2 Fully-Secure CP-ABE for Circuits

We obtain multi-linear pair encoding schemes for CP-ABE by applying the dual conversion to our two encoding schemes for KP-ABE. In particular, we obtain two schemes: $\mathsf{P}_{\mathsf{CPABE1}} := \textsc{Con}(\mathsf{P}_{\mathsf{KPABE1}})$ and $\mathsf{P}_{\mathsf{CPABE2}} := \textsc{Con}(\mathsf{P}_{\mathsf{KPABE2}})$. The efficiency is obtained by swapping the key encoding size and the ciphertext encoding size of the original KP-ABE schemes, plus one element for each encoding due to the conversion. Therefore, both resulting CP-ABE schemes admit succinctness, and the second CP-ABE achieves constant-size keys. The functionality is also preserved, hence they can deal with unbounded-size circuits.

From Lemma 5 and the security of $\mathsf{P}_{\mathsf{KPABE1}}$ and $\mathsf{P}_{\mathsf{KPABE2}}$ (Theorem 4,5 and 6,7), and the fact that $(1, \mathsf{poly})$-SMH trivially implies $(1,1)$-SMH, we have the following corollaries. Recall that ℓ is the bounded depth, n is the input length, while m is the width of the queried circuit.

Corollary 7. $\mathsf{P}_{\mathsf{CPABE1}}$ *is* $(1,1)$-CMH *under the* ℓ-$\mathsf{SMDDH1}$ *assumption.*

Corollary 8. $\mathsf{P}_{\mathsf{CPABE1}}$ *is* $(1,1)$-SMH *under the* (ℓ,m)-$\mathsf{EMDDH1}$ *assumption.*

Corollary 9. $\mathsf{P}_{\mathsf{CPABE2}}$ *is* $(1,1)$-CMH *under the* (ℓ,n)-$\mathsf{SMDDH2}$ *assumption.*

Corollary 10. $\mathsf{P}_{\mathsf{CPABE2}}$ *is* $(1,1)$-SMH *under the* (ℓ,m,n)-$\mathsf{EMDDH2}$ *assumption.*

All the above corollaries admit tight reductions. From these and Corollary 3, we obtain fully secure CP-ABE schemes with $O(q_{\text{all}})$ reduction as follows.

Corollary 11. $\mathsf{ABE}(\mathsf{P}_{\mathsf{CPABE1}}, \mathsf{G})$ *is fully secure under* $\mathsf{EMDDH1}, \mathsf{SMDDH1}, \mathsf{MSD}$. $\mathsf{ABE}(\mathsf{P}_{\mathsf{CPABE2}}, \mathsf{G})$ *is fully secure under* $\mathsf{EMDDH2}, \mathsf{SMDDH2}, \mathsf{MSD}$. *We have*

$$\begin{aligned}
\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ABE}(\mathsf{P}_{\mathsf{CPABE1}},\mathsf{G})}(\lambda) \leq & \mathsf{Adv}_{\mathcal{B}_1}^{(\kappa,\bar{S}_{\mathsf{c}})\text{-}\mathsf{MSD}}(\lambda) + 2q_{\text{all}}\mathsf{Adv}_{\mathcal{B}_2}^{(\kappa,\bar{S}_{\mathsf{k}})\text{-}\mathsf{MSD}}(\lambda) \\
& + q_1\mathsf{Adv}_{\mathcal{B}_3}^{\ell\text{-}\mathsf{SMDDH1}}(\lambda) + q_2\mathsf{Adv}_{\mathcal{B}_4}^{(\ell,m)\text{-}\mathsf{EMDDH1}}(\lambda). \\
\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ABE}(\mathsf{P}_{\mathsf{CPABE2}},\mathsf{G})}(\lambda) \leq & \mathsf{Adv}_{\mathcal{B}_1}^{(\kappa,\bar{S}_{\mathsf{c}})\text{-}\mathsf{MSD}}(\lambda) + 2q_{\text{all}}\mathsf{Adv}_{\mathcal{B}_2}^{(\kappa,\bar{S}_{\mathsf{k}})\text{-}\mathsf{MSD}}(\lambda) \\
& + q_1\mathsf{Adv}_{\mathcal{B}_3}^{(\ell,n)\text{-}\mathsf{SMDDH2}}(\lambda) + q_2\mathsf{Adv}_{\mathcal{B}_4}^{(\ell,m,n)\text{-}\mathsf{EMDDH2}}(\lambda).
\end{aligned}$$

Here, $\kappa = 3\ell$, $\bar{S}_{\mathsf{c}} = [\ell+2, 3\ell]$,$\bar{S}_{\mathsf{k}} = [1, \ell+1]$.

References

1. M. Albrecht, P. Farshim, D. Hofheinz, E. Larraia, K. Paterson. Multilinear Maps from Obfuscation. In *TCC 2016-A*, *LNCS*, pp. 446–473, 2016.
2. P. Ananth, Z. Brakerski, G. Segev, V. Vaikuntanathan. From Selective to Adaptive Security in Functional Encryption. In *Crypto (2) 2015*, *LNCS*, pp. 657–677, 2015.
3. N. Attrapadung. Dual System Encryption via Doubly Selective Security: Framework, Fully-secure Functional Encryption for Regular Languages, and More. In *Eurocrypt 2014*, *LNCS*, pp. 557-577, 2014.
4. N. Attrapadung. Fully Secure and Succinct Attribute Based Encryption for Circuits from Multi-linear Maps, 2014. Cryptology ePrint Archive: Report 2014/772.
5. N. Attrapadung. Dual System Encryption Framework in Prime-Order Groups via Computational Pair Encodings. In *Asiacrypt 2016*, *LNCS*, pp. 591–623, 2016.
6. N. Attrapadung. Dual System Framework in Multilinear Settings and Applications to Fully Secure (Compact) ABE for Unbounded-Size Circuits. (The full version of this paper). Cryptology ePrint Archive, 2017.
7. N. Attrapadung, B. Libert. Functional Encryption for Inner Product: Achieving Constant-Size Ciphertexts with Adaptive Security or Support for Negation. In *PKC 2010*, *LNCS*, pp. 384–402, 2010.
8. N. Attrapadung, B. Libert, E. Panafieu. Expressive Key-Policy Attribute-Based Encryption with Constant-Size Ciphertexts. In *PKC 2011*, *LNCS*, pp. 90–108, 2010.

9. N. Attrapadung, S. Yamada. Duality in ABE: Converting Attribute Based Encryption for Dual Predicate and Dual Policy via Computational Encodings. In *CT-RSA 2015*, *LNCS*, pp. 87–105, 2015.
10. J. Bethencourt, A. Sahai, B. Waters. Ciphertext-Policy Attribute-Based Encryption. In *IEEE S&P 2007*, pp. 321–334, 2007.
11. Z. Brakerski, V. Vaikuntanathan. Circuit-ABE from LWE: Unbounded Attributes and Semi-Adaptive Security. In *Crypto 2016*, *LNCS*, pp. 363–384, 2016.
12. D. Boneh, X. Boyen. Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles. In *Journal of Cryptology*, 24 (4), pp. 659–693, 2011. Extended abstract in *Eurocrypt 2004*, *LNCS* 3027, pp. 223–238, 2004.
13. 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.
14. D. Boneh, C. Gentry, B. Waters. Collusion Resistant Broadcast Encryption with Short Ciphertexts and Private Keys. In *Crypto 2005*, *LNCS*, pp. 258–275, 2005.
15. D. Boneh, A. Silverberg. Applications of multilinear forms to cryptography. Contemporary Mathematics Vol. 324, pp. 71–90, 2003.
16. X. Boyen, X. Fan, E. Shi. Adaptively Secure Fully Homomorphic Signatures Based on Lattices . Cryptology ePrint Archive, Report 2014/916.
17. J. H. Cheon, K. Han, C. Lee, H. Ryu, D. Stehle. Cryptanalysis of the multilinear map over the integers. In *Eurocrypt 2015*, *LNCS*, pp. 3–12, 2015.
18. J. H. Cheon, P-A. Fouque, C. Lee, B. Minaud, H. Ryu. Cryptanalysis of the New CLT Multilinear Map over the Integers. In *Eurocrypt 2016*, *LNCS*, pp. 509–536, 2016.
19. J. Chen, H. Wee. Semi-adaptive Attribute-Based Encryption and Improved Delegation for Boolean Formula. In *SCN 2014*, *LNCS*, pp. 277–297, 2014.
20. J. Coron, T. Lepoint, M. Tibouchi. Practical Multilinear Maps over the Integers. In *Crypto 2013*, *LNCS*, pp. 476–493, 2013.
21. J.-S. Coron, C. Gentry, S. Halevi, T. Lepoint, H.K. Maji, E. Miles, M. Raykova, A. Sahai, M. Tibouchi. Zeroizing without low-level zeroes: new attacks on multilinear maps and their limitations. In *Crypto 2015 (1)*, *LNCS*, pp. 247-266.
22. J. Coron, T. Lepoint, M. Tibouchi. New Multilinear Maps over the Integers. In *Crypto 2015 (1)*, *LNCS*, pp. 267-286.
23. S. Garg, C. Gentry, S. Halevi. Candidate multilinear maps from ideal lattices In *Eurocrypt 2013*, *LNCS*, pp. 1–17, 2013.
24. S. Garg, C. Gentry, S. Halevi, A. Sahai, B. Waters. Attribute-based encryption for circuits from multilinear maps. In *Crypto 2013*, *LNCS*, pp. 479–499, 2013.
25. S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, B. Waters. Candidate Indistinguishability Obfuscation and Functional Encryption for all circuits. In *FOCS 2013*, pp. 40–49, 2013.
26. S. Garg, C. Gentry, S. Halevi, M. Zhandry. Fully Secure Attribute Based Encryption from Multilinear Maps. Cryptology ePrint Archive: Report 2014/622, 2014.
27. S. Garg, C. Gentry, S. Halevi, M. Zhandry. Fully secure functional encryption without obfuscation. In *TCC 2016-A*, *LNCS*, pp 480–511. Cryptology ePrint Archive, Report 2014/666.
28. S. Garg, C. Gentry, A. Sahai, B. Waters. Witness encryption and its applications. In *STOC 2013*, pp. 467–476, 2013.
29. C. Gentry, A. Lewko, B. Waters. Witness Encryption from Instance Independent Assumptions. In *Crypto 2014*, *LNCS*, pp. 426–443, 2014.

30. C. Gentry, A. Lewko, A. Sahai, B. Waters. Indistinguishability Obfuscation from the Multilinear Subgroup Elimination Assumption. In *FOCS 2015*, pp. 151–170, 2105.
31. S. Gorbunov, V. Vaikuntanathan, H. Wee. Attribute-based encryption for circuits. In *STOC 2013*, pp. 545–554, 2013.
32. R. Goyal, V. Koppula, B. Waters. Semi-Adaptive Security and Bundling Functionalities Made Generic and Easy. In *TCC 2016-B*, *LNCS*, to appear.
33. V. Goyal, O. Pandey, A. Sahai, B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM CCS 2006*, pp. 89–98, 2006.
34. J. Katz, A. Sahai, B. Waters. Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products. In *Eurocrypt 2008*, *LNCS*, pp. 146–162, 2008.
35. A. Lewko, B. Waters. New Techniques for Dual System Encryption and Fully Secure HIBE with Short Ciphertexts. In *TCC 2010*, *LNCS*, pp. 455–479, 2010.
36. A. Lewko, B. Waters. New Proof Methods for Attribute-Based Encryption: Achieving Full Security through Selective Techniques. In *Crypto 2012*, *LNCS*, pp. 180–198, 2012.
37. 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*, *LNCS*, pp. 62–91, 2010.
38. H. Lin. Indistinguishability Obfuscation from Constant-Degree Graded Encoding Schemes. In *Eurocrypt 2016*, *LNCS*, pp. 28–57, 2016.
39. M. Naor, On Cryptographic Assumptions and Challenges. In *Crypto 2003*, *LNCS*, pp. 96–109, 2003.
40. T. Okamoto, K. Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In *Crypto 2010*, *LNCS*, pp. 191–208, 2010.
41. T. Okamoto, K. Takashima. Adaptively Attribute-Hiding (Hierarchical) Inner Product Encryption. In *Eurocrypt 2012*, *LNCS*, pp. 591–608, 2012.
42. R. Pass, K. Seth, S. Telang. Obfuscation from Semantically-Secure Multi-linear Encodings. In *Crypto 2014*, *LNCS*, pp. 500–517, 2014.
43. Y. Rouselakis, B. Waters Practical constructions and new proof methods for large universe attribute-based encryption. In *ACM CCS 2013*, pp. 463–474, 2013.
44. A. Sahai, B. Waters. Fuzzy Identity-Based Encryption In *Eurocrypt 2005*, *LNCS*, pp. 457–473, 2005.
45. K. Takashima. Expressive Attribute-Based Encryption with Constant-Size Ciphertexts from the Decisional Linear Assumption. In *SCN 2014*, *LNCS*, pp. 298–317, 2014.
46. L. G. Valiant. Universal circuits (preliminary report). In *STOC 1976*, pp. 196–203.
47. B. Waters. Ciphertext-Policy Attribute-Based Encryption: An Expressive, Efficient, and Provably Secure Realization. In *PKC 2011*, *LNCS*, pp. 53–70, 2011.
48. B. Waters. Dual System Encryption: Realizing Fully Secure IBE and HIBE under Simple Assumptions. In *Crypto 2009*, *LNCS*, pp. 619–636, 2009.
49. B. Waters. Functional Encryption for Regular Languages. In *Crypto 2012*, *LNCS*, pp. 218–235, 2012.
50. B. Waters. A Punctured Programming Approach to Adaptively Secure Functional Encryption. In *Crypto (2) 2015*, *LNCS*, pp. 678–697, 2015.
51. H. Wee. Dual System Encryption via Predicate Encodings. In *TCC 2014*, *LNCS*, pp. 616–637, 2014.