Fully Secure and Succinct Attribute Based Encryption for Circuits from Multi-linear Maps

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

Abstract

We propose new fully secure attribute based encryption (ABE) systems for polynomial-size circuits in both key-policy and ciphertext-policy flavors. All the previous ABE systems for circuits were proved only selectively secure. Our schemes are based on asymmetric graded encoding systems in composite-order settings. The assumptions consist of the Subgroup Decision assumptions and two assumptions which are similar to Multi-linear Decisional Diffie-Hellman assumption (but more complex) and are proved to hold in the generic graded encoding model. Both of our systems enjoy succinctness: key and ciphertext sizes are proportional to their corresponding circuit and input string sizes. Our ciphertext-policy ABE for circuits is the first to achieve succinctness, and the first that can deal with unbounded-size circuits (even among selectively secure systems). We develop new techniques for proving co-selective security of key-policy ABE for circuits, which is the main ingredient for the dual-system encryption framework that uses computational arguments for enforcing full security.

Keywords. Attribute-based encryption for circuits, Full security, Multi-linear maps, Dual system encryption, Ciphertext-policy, Key-policy, Succinctness

1 Introduction

Attribute-based encryption (ABE), introduced by Sahai and Waters [25], is an important 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. There are two variants of ABE: Key-Policy [18] and Ciphertext-Policy [4]. In a key-policy ABE system, a ciphertext encrypting message M is associated with attribute x. A secret key, which is issued by an authority, is associated with policy f which is a boolean function from some class $\mathcal{F}$. The decryption on a ciphertext associated with x by a secret key associated with f will succeed if and only if $f(x) = 1$. In a ciphertext-policy ABE system, the roles of f and x are exchanged: they are associated to ciphertext and secret key respectively.

A central theme to ABE has been to expand the class of allowable boolean functions $\mathcal{F}$. Until recently, there were only ABE for simple classes such as boolean formulae [18, 4, 23, 26] and inner product predicate [19, 2, 27]. Only recently, ABE systems that allow any polynomial-size circuits were proposed independently by Garg *et al.* (GGHSW) [9] and Gorbunov *et al.* (GVW) [17]. The former is based on multi-linear maps (more precisely, graded encoding systems) [8, 7], 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.* [12] proposed ABE for circuits based on witness encryption.

The standard security requirement for ABE is *adaptive security* which is also dubbed as *full security*. However, all the available ABE systems for circuits [9, 17, 12] were proved only in a weaker model called *selective security*. Such a notion requires the adversary to announce a target string $x^\star$ upfront before seeing the public key, after then, he can ask for secret keys for f such that $f(x^\star) = 0$. This is in contrast with full security where the adversary receives the public key first and can adaptively ask for secret keys and choose a target string in any order. There is a trivial method to generally bootstrap selective security to full security called *complexity leveraging*. In this approach, the reduction algorithm would guess which string will be chosen as $x^\star$ and simulate the security games from it. Hence, the reduction cost to the underlying hard problem will be reduced by factor 2^{-n}, where n is the length of string x, which the probability that the guess is correct. Constructing ABE for circuits with polynomial reductions in all parameters (including n) to some underlying problems has been an important open problem.

Our Contributions. We propose new fully secure ABE systems for circuits with polynomial reduction (these are the first such schemes, along with concurrent and independent work, see below). We provide both key-policy and ciphertext-policy variants. Both of our ABE systems allow circuits with unbounded size and unbounded fan-out (but bounded depth and bounded input-size), which is exactly the same property as obtained by the KP-ABE of GGHSW [9]. In particular, our CP-ABE is the first scheme (even among selectively-secure ones) that allows unbounded-size circuits. The CP-ABE of GGHSW allows only bounded-size circuits due to their use of universal circuits.

Both of our ABE systems enjoy *succinctness*, meaning that, for the key-policy variant, the size of a key for circuit f is proportional to the size of circuit f, and the size of a ciphertext for boolean string x is proportional to the number of 1's in x. Our CP-ABE system also has analogous efficiency; in particular, to the best of our knowledge, it is the first to achieve succinctness (even among selectively-secure ones). We discuss this in §8.

Our ABE systems are based on multi-linear maps. More precisely, we use composite-order asymmetric graded encoding systems. Such systems was proposed by Coron *et al.* [7] and was recently extended by Gentry *et al.* [13] to the composite-order setting. In our schemes for circuits of bounded depth ℓ, we require 3ℓ-multi-linear maps. We introduce some new assumptions on graded encoding: two subgroup decision assumptions which are extended naturally from the case of bilinear maps, and two assumptions which are similar to the Multi-linear Decisional Diffie-Hellman Assumption (MDDH) [6, 7]. One of the MDDH-related assumptions is parameterized and quite complex, but we prove that they hold in the generic graded encoding model. The parameters for the assumption depend only on the size and depth of a circuit in one query (and not on the number of key queries). The reduction cost to these assumptions is $O(q_1)$ where q_1 is the number of pre-challenge key queries (and hence we obtain polynomial reduction as desired).

Concurrent and Independent Work. Concurrently and independently, Garg *et al.* [11] (GGHZ) recently proposed a fully-secure ABE scheme for circuits. Although, at a high-level view, both of our work and the GGHZ system use similar ingredients, namely composite-order multi-linear maps and dual-system encryption approaches [29], the methods for enforcing full security are completely different. In their scheme, the use of "fixed-once and for all" universal circuits is essential. This has a consequence that the system works for only bounded-size circuits. Moreover, in their scheme, both ciphertexts and keys always have fixed sizes being roughly the size of the fixed universal circuits. This means that they are equal to the worst-case sizes (among circuits and strings on which the fixed universal circuit can process, respectively), *i.e.,* the scheme is not succinct. On the other hand, our ABE systems are succinct and can deal with unbounded-size circuits. Furthermore, their scheme requires multi-linear maps with multi-linearity being roughly also the size of the fixed universal circuits, while our systems require multi-linearity being proportional to the bounded depth (3ℓ). Nevertheless, their scheme enjoys simpler assumptions (but again with larger multi-linearity).

As another independent work, Waters [31] recently proposed fully-secure *functional encryption* (FE) for circuits based on indistinguishability obfuscation (IO) [10, 14]. His result is thus stronger than both ours and GGHZ. However, the current security proof of IO under a polynomial-size set of assumptions requires complexity leveraging and hence exponential loss in reduction [14].

1.1 Difficulties and Our Approaches

Applying Dual System Frameworks. Dual system encryption techniques, introduced by Waters [29], have been successful approaches for constructing fully-secure ABE (for simple classes). Our attempt will be to apply this approach to an existing (selectively-secure) KP-ABE for circuits, namely the GGHSW scheme. Until recently, dual system approaches had been considered as security proof "techniques", where many technical subtle details are buried deep inside the proof, and sometimes this makes it hard to understand. To this end, Wee [32] and Attrapadung [1] independently proposed generic frameworks for dual system approaches. They introduce sufficient primitives (called predicate encoding in [32], and pair encoding in [1]), defined for a predicate, that imply fully-secure ABE for that predicate via generic constructions. Both works describe abstractions of *information-theoretic* arguments which seem to be essential but were implicit in previous dual-system based schemes. Moreover, the framework of [1] describes also an *computational* analog, which generalizes the technique in the ABE (for boolean formulae) of Lewko and Waters [22].

Difficulties in the Case of ABE for Circuits. We examine our canonical scheme, the GGHSW ABE, using the framework of [1]. It turns out that the underlying structure of GGHSW does not possess information-theoretic security required for the dual-system framework. We believe that an intuitive reason for this can be described as follows. As emphasized by Garg *et al.* themselves [9], an essential attack that needs to be prevented when considering general circuits is the so-called "backtracking attack". This attack exploits the distinctive feature of circuits over formulae in that circuits can have gates with fan-out more than one. The attack would occur at an OR gate by somehow "tracking backward". Therefore, to prevent this was the reason why Garg *et al.* essentially used graded multi-linear maps so that the information can flow only forward (towards the output gate). Now, if we would consider their underlying structure in an information-theoretic sense, *i.e.,* without encoding in multi-linear groups, the countermeasure to backtracking will then essentially be lost. Hence, this is the reason why the dual system framework that uses information-theoretic argument would not work for GGHSW. We indeed elaborate checking it concretely in §D.

Applying Computational Approach. The framework of [1] provides a variant of dual system framework that uses computational arguments as ingredients, and was generalized from the ABE for formulae by Lewko and Waters [22]. It was applied to ABE systems in which the information-theoretic structure underlies does not possess required security, such as the ABE for regular languages of Waters [30], the short-ciphertext ABE for formulae of [3], the unbounded ABE for formulae of [21, 24]. Fully secure variants of these primitives were then successfully obtained via the framework of [1]. We would like to do the same to ABE for circuits of GGHSW. To work in this framework, it is quite surprising that the requirement is roughly that the considering ABE is both selectively secure and *co-selectively secure* at the same time. (There is a caveat, see below). Co-selective security is a dual notion to (the more widely-known) selective security. Instead of announcing a target string $x^\star$ for ciphertext upfront as in the selective security game, the adversary $\mathcal{A}$ would announce a circuit f upfront, the challenger then gives the public key and a key for f, after that, $\mathcal{A}$ then asks for a challenge ciphertext for any string $x^\star$ as long as $f(x^\star) = 0$.[1] There is a caveat that it is not exactly this requirement of selective and co-selective security of ABE itself,

[1]The number of key and ciphertext queries can be defined variedly. In our context, we will (implicitly) require one query for each, similarly to [1].

we will actually need the analogous notions but of its underlying pair encoding of ABE. The two analogous notions are called selective and co-selective master-key hiding. Nevertheless, by using techniques similarly to [22, 1], we can convert selective security of ABE to selective master-key hiding of pair encoding almost all the time. It is worth noting that proving co-selective security of KP-ABE is likely to be similar to proving *selective* security of *CP*-ABE due to its dual nature.

The Missing Piece: Co-selectively Secure KP-ABE for Circuits. The GGHSW KP-ABE is already selectively secure. However, the proof of co-selective security for KP-ABE (of GGHSW or other schemes) or selective security proof of CP-ABE (without using universal circuits) is not known. The main technical novelty in this paper is essentially to prove the co-selective security of (a variant of) the GGHSW KP-ABE. (We will indeed prove the full security of our KP-ABE directly, but the co-selective proof structure is essential).

Difficulties for Constructing Co-selectively Secure KP-ABE for Circuits. The first evidence that constructing co-selectively secure KP-ABE for circuits, or somewhat equivalently, selectively secure CP-ABE for circuits, can be hard is that the selectively secure CP-ABE scheme for formulae by Waters [28] is proved under an already more complex assumption than the KP-ABE counterpart [18], namely the Parallel BDHE assumption.[2]

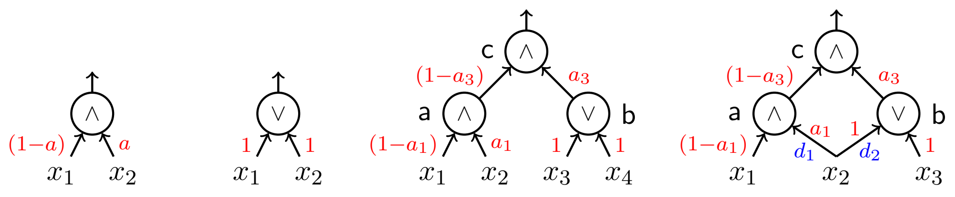
Our goal is to generalize the selective security proof of Waters' CP-ABE (or equivalently, we may think of co-selective security of its dual KP-ABE) to the case of general circuits. This poses two main issues. First, the output of a gate can be wired as an input to another gate (we call this a hierarchy issue). Second, and more essentially, the output of a gate (or a circuit input) can be wired as inputs of 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.

Dealing with Hierarchy. We explain the technical difficulty of this issue via a toy example of circuit f^2 in Figure 1. We would like to apply the simulation for CP-ABE with one gate (*e.g.,* f^{AND} in Figure 1) to any gate of f^2 in a modular way. In the co-selective proof, the reduction simulates a key for circuit f^2 first, gives $\mathsf{SK}_{f^2}, \mathsf{PK}$ to the adversary, and receive a target string x. Suppose that $x^\star = 1100$ (so that $f^2(x^\star) = 0$). The reduction is in the situation where at the AND gate a, the output of a is evaluated to 1 (denoted $f^2_{\mathsf{a}}(x^\star) = 1$). However, if we only use the "stand-alone" proof technique for single gate to this gate a, we will not be able to simulate, since the restriction was that the output of the gate must be 0! We resolve this issue by simulating the key for f^2 in such a way that the key elements for lower gates (*e.g.,* gate a) are embedded with all information of the gates (*e.g.,* gate b) that are on their path to the output. Therefore, we effectively chain all the information on the path aggregated to one element. To make the chaining technique work, we have to have a mechanism that identifies gate w such that $f^2_w(x^\star) = 0$. We do this by providing "individual randomness" from the assumption to each gate. We elaborate more details in §2.

Dealing with Multi-fan-out. We explain the technical difficulty of this issue via a toy example of circuit f^3 in Figure 1. The above discussion suggests to simulate a key element at x_2 by combining all information from all the paths from it to the output gate, *i.e.,* both the path through a and the one through b. The combined information would cease the "stand-alone" proof technique since the information from each path would interfere the simulation of each other. To solve this, we must distinguish each path; we do this by introducing another set of "individual randomness" from the assumption and dedicate each to each outgoing wire. Another problem that arises when considering multi-fan-out is that we may have many paths from a certain gate to the output gate: it can be exponential size in the depth of circuits. Hence, we cannot afford to have the size of assumption to be as large so as to prepare for any possible chains. We resolve this by decomposing any of possible

[2]There are three schemes in [28]. Here we consider his only scheme that is without restriction of bounded repetition.

Figure 1: Toy Examples: circuit $f^{\mathsf{AND}}, f^{\mathsf{OR}}, f^2, f^3$ respectively. (Texts in colors are related to simulation).



chains to multiplicative combinations via the use of multi-linear maps in a confined manner so that only required combinations can be produced and no more. We end up with 3ℓ-multi-linear maps for ABE that allows circuits of bounded depth ℓ. We also must find an algorithm for producing the chain information on the fly since the number of all paths can be exponential. We solve this by providing sophisticated recursive algorithms that run in poly-time in the number of gates.

1.2 Other Related Work

Boneh *et al.* [5] proposed ABE for arithmetic circuits by extending the GVW system. They also proposed KP-ABE for boolean circuits with constant-size ciphertexts. Their systems are selectively secure. Goldwasser *et al.* [15, 16] proposed ABE and FE systems for Turing Machines that are secure against bounded collusions. Our ABE systems are fully secure against unbounded collusions.

2 Toy Examples and Intuition in Technical Details

Before describing formal details, we describe the intuition of our scheme. Readers may skip this in the first read. We illustrate how we solve the difficulties of proving co-selective security of ABE for circuits, mentioned above. We use toy examples for concreteness. We describe these using multi-linear maps in algebraic group setting like in bilinear pairings, which we assume that readers are familiar with. For every system here, we would like prove its co-selective security. In this notion, the adversary $\mathcal{A}$ gives the circuit f that it wants to query, the challenger then gives the $\mathsf{PK}, \mathsf{SK}_f$ to $\mathcal{A}$, who will then ask for the challenge ciphertext for any string x of its choice as long as $f(x) = 0$. In the discussion below, decryption is not important here and we refer to §D.

Toy Example 1: Single-gate Circuit. We first consider a toy ABE system that allows only one gate. Our toy scheme has public key $\mathsf{PK} = (g_1^{h_1}, g_1^{h_2}, e(g_1, g_2)^{\alpha})$, and master key $\mathsf{MSK} = \alpha$. Keys for $\mathsf{AND}, \mathsf{OR}$ are $\mathsf{SK}_{\mathsf{AND}} = (g_2^{\alpha+h_1\ell+h_2 r}, g_2^{\ell}, g_2^{r})$, $\mathsf{SK}_{\mathsf{OR}} = (g_2^{\alpha+h_1\ell}, g_2^{\alpha+h_2 r}, g_2^{\ell}, g_2^{r})$, resp. (The variable ℓ, r is random and specific to each key). For $x \in \{0,1\}^*$, let $A_x = \{\, j \mid x_j = 1 \,\}$. We define a ciphertext for $x \in \{0,1\}^2$ as $\mathsf{CT}_x = (Me(g_1, g_2)^{\alpha s}, g_1^s, \{g_1^{sh_j}\}_{j \in A_x})$. We prove its co-selective security under an assumption stating that given $g_1, g_1^{c_1}, g_1^{c_1 a}, g_1^{z}, g_1^{\frac{z}{a}}, g_1^{\frac{zc_1}{a}}, g_1^{zc_1 a}, g_2, g_2^{c_1}, g_2^{c_2}, g_2^{c_1 a}$, it is hard to decide if $Z = e(g_1, g_2)^{c_1 c_2 z}$ or $Z \in_R \mathbb{G}_T$. This extends the Bilinear-DH assumption (in asymmetric groups) with additional elements involving a. The proof is as follows. We consider the case where $\mathcal{A}$ chooses to obtain $\mathsf{SK}_{\mathsf{AND}}$. The reduction programs $\alpha = c_1 c_2$, $h_1 = c_1(1-a)$, $h_2 = c_1 a$ by setting PK as $g_1^{h_1} = g_1^{c_1} g_1^{-c_1 a}, g_2^{h_2} = g_2^{c_1 a}, e(g_1, g_2)^{\alpha} = e(g_1^{c_1}, g_2^{c_2})$. Here, we neglect re-randomizing parameters, *e.g.*, defining $h_2 = c_1 a + h_2'$ for some known random h_2', for simplicity. The reduction then programs $\ell = -c_2, r = -c_2$ by setting $\mathsf{SK}_{\mathsf{AND}}$ as $g_2^{\alpha+h_1\ell+h_2 r} = g_2^{c_1c_2 - c_1(1-a)c_2 - c_1 a c_2} = 1$ (when parameters are re-randomized, this implies that it is computable), and $g_2^{\ell} = g_2^{r} = g_2^{-c_2}$. Simulating in this manner allows us to produce a challenge ciphertext for every case: $x = 10$ or $x = 01$. If $x = 10$, we programs $s = z(1 + \frac{1}{a})$ by setting $g_1^s = g_1^z g_1^{\frac{z}{a}}$, $g_1^{sh_1} = g_1^{z(1+\frac{1}{a})c_1(1-a)} = g_1^{\frac{zc_1}{a}} g_1^{-zc_1 a}$, and the message mask as $Z \cdot e(g_1^{\frac{zc_1}{a}}, g_2^{c_2})$. If $x = 01$, we simply program $s = z$ by setting $g_1^s = g_1^z$, $g_1^{sh_2} = g_1^{zc_1 a}$, and the message mask as Z. An important point here is that in both cases, an unknown critical term $g_1^{zc_1}$ is canceled out: the $g_1^{sh_j}$ terms only have $g_1^{zc_1 x}$ for some $x \neq 1$. We may say that $(1+\frac{1}{a})$ acts as a "selector" that triggers canceling 1 in $(1-a)$. The case for OR is similar but more straightforward as the "coefficients" for the left child and right child become both 1 (that is, $h_1 = c_1, h_2 = c_1$).

Toy Example 2: Depth-two Circuit. Next, we show how to deal with hierarchy by considering a toy circuit with depth two. The idea is to construct ABE for this class by using the single-circuit scheme at each gate. We use asymmetric graded 3-linear maps $\mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{G}_3 \to \mathbb{G}_T$. Let g_i be a generator in each group. Denote $g_{12} = e(g_1, g_2)$, and so on. This toy scheme has $\mathsf{PK} = (g_1^{h_1}, \ldots, g_1^{h_4}, g_{123}^{\alpha})$, $\mathsf{MSK} = \alpha$. For the circuit f^2 in Figure 1, $\mathsf{SK}_f = (K_\mathsf{a}, K_\mathsf{b}, K_\mathsf{c})$ where $K_\mathsf{a} = (g_2^{\alpha_\mathsf{a} + h_1 \ell_\mathsf{a} + h_2 r_\mathsf{a}}, g_2^{\ell_\mathsf{a}}, g_2^{r_\mathsf{a}})$, $K_\mathsf{b} = (g_2^{\alpha_\mathsf{b} + h_3 \ell_\mathsf{b}}, g_2^{\alpha_\mathsf{b} + h_4 r_\mathsf{b}}, g_2^{\ell_\mathsf{b}}, g_2^{r_\mathsf{b}})$, and $K_\mathsf{c} = (g_{23}^{\alpha + \alpha_\mathsf{a} \ell_\mathsf{c} + \alpha_\mathsf{b} r_\mathsf{c}}, g_3^{\ell_\mathsf{c}}, g_3^{r_\mathsf{c}})$. Note that α_w for gate w is randomness dedicated to only this key. We define $\mathsf{CT}_x = (M g_{123}^{\alpha s}, g_1^s, \{g_1^{sh_j}\}_{j \in A_x})$. In 3-linear map, the problem instance becomes to to distinguish $g_{123}^{zc_1c_2c_3}$ with random. The explanation from §1.1 translates to the following: we will need an information embedded in h_1, h_2 so that the "selector" at gate c can enable canceling even at the different gate a. We resolve this as follows. First, in order to identify which gate where the selector was chosen, we prepare individual randomness a in the assumption dedicated for every AND gate. (OR gates use coefficient 1). In this example, our assumption would involve a_1, a_3 (instead of only one a) dedicated to gate a, c, respectively. Then, the reduction programs h_1, h_2 by "chaining" the information from both gate a and c: it defines $h_1 = c_1(1-a_1)(1-a_3)$ and $h_2 = c_1 a_1 (1-a_3)$. Here, for the left child node, we use $(1-a)$, and for the right child node we use a, as in the first toy example. (We illustrate this with red-colored texts in Fig. 1). Moreover, to simulate SK_{f^2}, the reduction programs $\alpha_\mathsf{a} = c_1c_2(1-a_3)$ and $\ell_\mathsf{a}, r_\mathsf{a} = -c_2$, so that $g_2^{\alpha_\mathsf{a} + h_1 \ell_\mathsf{a} + h_2 r_\mathsf{a}} = g_2^{c_1c_2(1-a_3) - c_1c_2(1-a_1)(1-a_3) - c_1c_2(1-a_1)(a_3)} = 1$. Now to simulate CT_{1100}, the reduction defines $s = z(1+\frac{1}{a_3})$ by setting $g_1^{sh_1} = g_1^{z(1+\frac{1}{a_3})c_1(1-a_1)(1-a_3)}$. Now, we have canceled out an unknown critical term $g_1^{zc_1}$ by the term $(1+\frac{1}{a_3})(1-a_3)$. We can afford to put $g_1^{zc_1x}$ for $x \neq 1$ that is a residue from the "gate mismatch", *e.g.*, $g_1^{zc_1\frac{a_1}{a_3}}, g_1^{zc_1a_1a_3}$ in a new assumption, as long as they do not help distinguish $g_{123}^{zc_1c_2c_3}$ from random.

Toy Example 3: Fan-out-two Circuit. Next, we show how to deal with multi-fan-out. A key SK_{f^3} for circuit f^3 is exactly the same as SK_{f^2} except that now $K_\mathsf{b} = (g_2^{\alpha_\mathsf{b} + h_2 \ell_\mathsf{b}}, g_2^{\alpha_\mathsf{b} + h_3 r_\mathsf{b}}, g_2^{\ell_\mathsf{b}}, g_2^{r_\mathsf{b}})$. Due to the fan-out two of the second input, the difficulty arises as h_2 must contain the "chaining" information for *both* paths from it to the output gate. A flawed attempt is to program h_2 as the sum of both chains: $h_2 = c_1a_1(1-a_3) + c_1 \cdot 1 \cdot a_3$ (1 due to b being OR gate). However, this would fail the cancelation when simulating key as: $g_2^{\alpha_\mathsf{a} + h_1 \ell_\mathsf{a} + h_2 r_\mathsf{a}} = {g_2}^{c_1c_2(1-a_3) - c_1c_2(1-a_1)(1-a_3) - c_1c_2\left(a_1(1-a_3)+a_3\right)} = g_2^{c_1c_2a_3}$ results in a critical term (since it can be used to break the assumption as $e(g_1^{z/a_3}, g_2^{c_1c_2a_3}, g_3^{c_3}) = g_{123}^{zc_1c_2c_3}$). We cannot afford modifying α_a, h_1 altogether to accommodate the path that passes through gate b, since the chaining mechanism would not work anymore. To this end, we introduce a technique so as to distinguish each outgoing wire and makes a mismatch term whenever that wire does not correspond to a gate in consideration. This can be done by preparing individual randomness dedicated for each outgoing wire: in our toy system we use d_1, d_2 for the two wires. We now program $h_2 = c_1a_1(1-a_3)d_1 + c_1a_3d_2$ for PK. Then for gate a, we program $r_\mathsf{a} = \frac{c_2}{d_1}$ and thus set ${g_2}^{\alpha_\mathsf{a} + h_1 \ell_\mathsf{a} + h_2 r_\mathsf{a}} = {g_2}^{c_1c_2(1-a_3) - c_1c_2(1-a_1)(1-a_3) - c_1\frac{c_2}{d_1}\left(a_1(1-a_3)d_1 + a_3d_2\right)} = {g_2}^{c_1c_2a_3\frac{d_2}{d_1}}$. Now we can afford to put the term occurred from the "outgoing-wire mismatch", ${g_2}^{c_1c_2a_3\frac{d_2}{d_1}}$, to a new assumption.

Generalizing to Any Circuits. We list some difficulties when generalizing our ideas in toy systems to any circuits and how we resolve them. We denote by ℓ the depth of the circuit.

First, the chaining mechanism requires terms of form $g_1^{c_1 a_1^{e_1} d_1 a_2^{e_2} d_2 \cdots a_\ell^{e_\ell} d_\ell}$, one term per one path, to be given in the assumption, where each a_i is dedicated to one gate on the path from an input gate to the output gate, $e_i \in \{0,1\}$, and each d_i is dedicated to one outgoing wire on that path. The problem is that since fan-out can be more than one, the number of possible paths from an input gate to the output gate can be of exponential size in the number of circuit depth (ℓ). Hence, the size of the given terms in the assumption would grow exponentially. We resolve this by using multi-linearity inside $\mathbb{G}_1$ also. That is, we define $\mathbb{G}_1 = \mathbb{G}'_1 \times \cdots \times \mathbb{G}'_\ell$, prepare $g_1'^{c_1 a_{j_1}^{e_1} d_{j_1}}, g_2'^{a_{j_2}^{e_2} d_{j_2}}, \ldots, g_\ell'^{a_{j_\ell}^{e_\ell} d_{j_\ell}}$ in separate groups $\mathbb{G}'_i$, and let the reduction computes pairing on the fly. This solves the problem of the assumption size, but then poses another problem that the reduction might run in exponential time due to an exponential number of chains. To this end, we propose a sophisticated polynomial-time recursive algorithm for the reduction to compute chains. See the detail in the proof in §7.3.

Second, the residue from cancellation, *e.g.*, remaining terms in $g_1^{sh_j}$ from "gate mismatch", such as $g_1^{zc_1 a_{j_1}^{e_1} a_{j_2}^{e_2} \frac{1}{a_{j_3}} a_{j_4}^{e_4} \cdots a_{j_\ell}^{e_\ell}}$, must also be decomposed to groups inside $\mathbb{G}_1$ due to the same reason as above. However, this time if we decompose in a "freedom" manner, *e.g.*, to $g_1'^{zc_1 a_{j_1}^{e_1}}, g_2'^{a_{j_2}^{e_2}}, \ldots, g_\ell'^{a_{j_\ell}^{e_\ell}}$, we would end up having a combination that leads to $g_1^{zc_1}$ (by choosing all $e_i = 0$), which is a critical term that leads to break the assumption. (We temporarily neglect d_i terms now for simplicity). We need to decompose in a restricted way so that all allowed combination must have $\frac{1}{a_{j_i}}$ for some j_i. We resolve this by using multi-linearity with one more dimension: $\mathbb{G}_1 = \mathbb{G}'_1 \times \cdots \times \mathbb{G}'_{\ell+1}$, and prepare not only $g_i'^{a_{j_i}^{e_i}}$ for all $i \in [1,\ell]$ but also $e(g'_i, g'_{\ell+1})^{zc_1 \frac{1}{a_{j_i}}}$ for all $i \in [1,\ell]$. In this way, to get an element in $\mathbb{G}_1$, one must use exactly one element from the latter.

In our ABE, it turns out that we use $\mathbb{G}_1, \ldots, \mathbb{G}_{\ell+1}$ for defining the scheme as in [9], and we will decompose $\mathbb{G}_1, \mathbb{G}_2$ respectively to $\ell+1$ and ℓ level of multi-linearity for utilizing the assumption. Hence, we will use 3ℓ level of multi-linearity in total. (Due to lengthy discussions here, we postpone one more technique to §D).

3 Definitions

Circuit Notation. A circuit consists of six tuples $f = (\ell, n, \{m_i\}_{i\in[2,\ell]}, \mathsf{L}, \mathsf{R}, \mathsf{GateType})$. We first note that it is wlog that we consider only *monotone* and *layered* circuits. (We refer to [9] for the discussion). We let ℓ be the number of layers, n be the number of inputs, and m_i be the number of gates in the i-th layer for $i \in [2,\ell]$. We also define $m_1 = n$ for notational consistency. We define $\mathsf{Inputs} = \{w_{1,1}, \ldots, w_{1,n}\}$, and for $i \in [2,\ell]$, $\mathsf{Gates}_i = \{w_{i,1}, \ldots, w_{i,m_i}\}$. We let $\mathsf{Gates} = \bigcup_{i\in[2,n]} \mathsf{Gates}_i$, and let $\mathsf{Nodes} = \mathsf{Inputs} \cup \mathsf{Gates}$. Since we consider only one output gate, we have that $m_\ell = 1$, and that $w_{\ell,1}$ is the output gate, which we also denote it as w_{top}. We define $\mathsf{Depth}(w_{i,j}) = i$ and $\mathsf{Num}(w_{i,j}) = j$. Since we consider a layered circuit, each of the two inputs of gates in Gates_i is wired from the output of gates in Gates_{i-1}. 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 those two 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 $\mathsf{Num}(\mathsf{L}(w_{i,j})) < \mathsf{Num}(\mathsf{R}(w_{i,j}))$. We note that $\mathsf{Depth}(\mathsf{L}(w_{i,j})) = \mathsf{Depth}(\mathsf{R}(w_{i,j})) = i-1$. The function $\mathsf{GateType} : \mathsf{Gates} \to \{\mathsf{OR}, \mathsf{AND}\}$ specifies the type of gate as either OR or AND gate. We also define $\mathsf{GatesOR}$ as the set of all OR gates, and similarly for $\mathsf{GatesAND}$. We abuse notation and denote $f(x)$ as the output of circuit f evaluated with input string x. For $w \in \mathsf{Gates}$, we also denote $f_w(x)$ to be the evaluation of string x at the output of gate w.

ABE for Circuits: Syntax. Consider a circuit family $\mathbb{F}_{n,\ell}$ that consists of all circuits with input length n and bounded depth ℓ. A (key-policy) attribute based encryption scheme for circuits in $\mathbb{F}_{n,\ell}$ with message space $\{0,1\}^\lambda$ consists of the following algorithms.

- $\mathsf{Setup}(1^\lambda, n, \ell) \to (\mathsf{PK}, \mathsf{MSK})$: takes as input a security parameter 1^λ, the length n of input strings to circuits, and the bound ℓ on the circuit depth, and outputs a public key PK and a master secret key MSK.
- $\mathsf{Encrypt}(\mathsf{PK}, x, M) \to \mathsf{CT}$: takes as input the public key PK, a string $x \in \{0,1\}^n$, and a message $M \in \{0,1\}^\lambda$. It outputs a ciphertext CT.
- $\mathsf{KeyGen}(\mathsf{MSK}, f) \to \mathsf{SK}$: takes as input the master key MSK and a circuit description $f \in \mathbb{F}_{n,\ell}$. It outputs a secret key SK.
- $\mathsf{Decrypt}(\mathsf{CT}, \mathsf{SK}) \to M$: takes as input a ciphertext CT and the decryption key SK. It attempts to decrypt and outputs a message M if successful; otherwise, it outputs a special symbol $\perp$.

Correctness. Consider all messages $M \in \{0,1\}^\lambda$, string $x \in \{0,1\}^n$, and circuit $f \in \mathbb{F}_{n,\ell}$ such that $f(x) = 1$. If $\mathsf{Encrypt}(\mathsf{PK}, x, M) \to \mathsf{CT}$ and $\mathsf{KeyGen}(\mathsf{MSK}, f) \to \mathsf{SK}$ where $(\mathsf{PK}, \mathsf{MSK})$ is generated from $\mathsf{Setup}(1^\lambda, n, \ell)$, then $\mathsf{Decrypt}(\mathsf{CT}, \mathsf{SK}) \to M$.

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

1 **Setup**: $\mathcal{C}$ runs $^{(0)}\boxed{\mathsf{Setup}(1^\lambda, n, \ell) \to (\mathsf{PK}, \mathsf{MSK})}$ and hands the public key PK to $\mathcal{A}$.

2 **Phase 1**: $\mathcal{A}$ makes a j-th key query for a circuit $f^{(j)} \in \mathbb{F}_{n,\ell}$. $\mathcal{C}$ returns $^{(1)}\boxed{\mathsf{SK}_j \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, f^{(j)})}$.

3 **Challenge**: $\mathcal{A}$ submits equal-length messages M_0, M_1 and a target string $x^\star \in \{0,1\}^n$ with the restriction that $f^{(j)}(x^\star) = 0$ for all $j \in [1, q_1]$. $\mathcal{C}$ flips a bit $\mathfrak{b} \xleftarrow{\$} \{0,1\}$ and returns the challenge ciphertext $^{(2)}\boxed{\mathsf{CT}^\star \leftarrow \mathsf{Encrypt}(\mathsf{PK}, x^\star, M_{\mathfrak{b}})}$.

4 **Phase 2**: $\mathcal{A}$ continues to make a j-th private key query for $f^{(j)} \in \mathbb{F}_{n,\ell}$, under the restriction $f^{(j)}(x^\star) = 0$. $\mathcal{C}$ returns $^{(3)}\boxed{\mathsf{SK}_j \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, f^{(j)})}$.

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

4 Graded Encoding

Our scheme will use asymmetric graded encoding systems in composite-order setting. Graded encoding systems were proposed by Garg *et al.* [8] (GGH) and subsequently by Coron *et al.* [7] (CLT) as cryptographically secure multi-linear maps. The main difference with bilinear pairings in algebraic groups is that the encoding of a (which is g^a in algebraic group setting) is not deterministic. We work on the CLT system since it was shown to extend to composite-order settings by Gentry *et al.* [13]. We recall their abstraction in this section and introduce some notations for it.

Definition 1 (Asymmetric Graded Encoding System)**.** A κ-Asymmetric Graded Encoding System for a ring $\mathcal{R}$ is a set system $\left\{ \mathcal{E}_S^{(a)} \subset \{0,1\}^* \;\middle|\; a \in \mathcal{R},\, S \subseteq [1,\kappa] \right\}$, with the following properties:

1. For every set $S \subseteq [1,\kappa]$, every distinct $a_1, a_2 \in \mathcal{R}$, we have $\mathcal{E}_S^{(a_1)} \cap \mathcal{E}_S^{(a_2)} = \emptyset$.
2. There are a binary operation '$+$' and an unary operation '$-$' on $\{0,1\}^*$ such that for every $a_1, a_2 \in \mathcal{R}$, every $S \subseteq [1,\kappa]$, every $u_1 \in \mathcal{E}_S^{(a_1)}, u_2 \in \mathcal{E}_S^{(a_2)}$, it holds that $u_1 + u_2 \in \mathcal{E}_S^{(a_1+a_2)}$ and that $-u_1 \in \mathcal{E}_S^{(-a_1)}$, where $a_1 + a_2$ and $-a_1$ are addition and negation in $\mathcal{R}$.

3. There is an associative binary operation '$\cdot$' on $\{0,1\}^*$ such that for every $a_1, a_2 \in \mathcal{R}$, every $S_1, S_2 \subseteq [1,\kappa]$ such that $S_1 \cap S_2 = \emptyset$, every $u_1 \in \mathcal{E}_{S_1}^{(a_1)}, u_2 \in \mathcal{E}_{S_2}^{(a_2)}$, it holds that $u_1 \cdot u_2 \in \mathcal{E}_{S_1 \cup S_2}^{(a_1 \cdot a_2)}$, where $a_1 \cdot a_2$ is multiplication in $\mathcal{R}$.

Bracket Notation. For $a \in \mathcal{R}$ and $S \subseteq [1,\kappa]$, we call an element in $\mathcal{E}_S^{(a)}$ as a *level-S encoding* of a and denote it as $[\,a; r\,]_S$, where r is the internal randomness (or *noise*) of this encoding. Let $\mathcal{N}$ be the set of all possible noises. That is, $\mathcal{E}_S^{(a)} = \{\,[\,a;r\,]_S \mid r \in \mathcal{N}\,\}$. From the definition, we have that for every $a_1, a_2 \in \mathcal{R}$, $r_1, r_2 \in \mathcal{N}$, there exists $r', r'' \in \mathcal{N}$ such that

$$[\,a_1; r_1\,]_S + [\,a_2; r_2\,]_S = [\,a_1 + a_2; r'\,]_S, \qquad [\,a_1; r_1\,]_{S_1} \cdot [\,a_2; r_2\,]_{S_2} = [\,a_1 \cdot a_2; r''\,]_{S_1 \cup S_2}. \tag{1}$$

For simplicity, from now on, unless noises are significant in the context, we abuse the notation by suppressing implicit noises and denoting an encoding simply by $[\,a\,]_S$. Hence we have

$$[\,a_1\,]_S + [\,a_2\,]_S = [\,a_1 + a_2\,]_S, \qquad [\,a_1\,]_{S_1} \cdot [\,a_2\,]_{S_2} = [\,a_1 \cdot a_2\,]_{S_1 \cup S_2}. \tag{2}$$

Composite-Order Setting. We will use a composite-order variant of graded encoding system where the encoding space is a direct product of subrings: $\mathcal{R} = \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_\nu}$, where each N_i is a composite number with (non-public) large prime factors. In our ABE, we use $\nu = 3$.

For $a \in \mathcal{R}$ and $V \subseteq [1,\nu]$, we denote $[\,a\,]_S^V := [\,a'\,]_S$, where we set $a' \in \mathcal{R}$ to be such that $a' \equiv a \pmod{N_i}$ for all $i \in V$ and $a' \equiv 0 \pmod{N_{i'}}$ for all $i' \notin V$. Hence we have $[\,a\,]_S^V = [\,a \bmod N_V\,]_S^V$, where $N_V := \prod_{i \in V} N_i$. We sometimes abuse the notation as $[\,a\,]_S^{\{i\}} = [\,a\,]_S^i$ and $[\,a\,]_S^{\{i,j\}} = [\,a\,]_S^{i,j}$. In the composite setting we have some useful properties. First, due to Chinese-Remainder Theorem (CRT), we have that for $V_1 \cap V_2 = \emptyset$ and every $a \in \mathcal{R}$, $[\,a\,]_S^{V_1 \cup V_2}$ can be decomposed uniquely as $[\,a\,]_S^{V_1 \cup V_2} = [\,a \bmod N_{V_1}\,]_S^{V_1} + [\,a \bmod N_{V_2}\,]_S^{V_2}$. We call $[\,a \bmod N_{V_1}\,]_S^{V_1}$ the N_{V_1} component of $[\,a\,]_S^{V_1 \cup V_2}$. Second, we have orthogonality: for $S_1 \cap S_2 = \emptyset$, $V_1, V_2 \subseteq [1,\nu]$, $a_1, a_2 \in \mathcal{R}$, it holds that

$$[\,a_1\,]_{S_1}^{V_1} \cdot [\,a_2\,]_{S_2}^{V_2} = [\,a_1 \cdot a_2\,]_{S_1 \cup S_2}^{V_1 \cap V_2}.$$

In particular, we have $[\,a_1\,]_{S_1}^i \cdot [\,a_2\,]_{S_2} = [\,a_1 \cdot a_2\,]_{S_1 \cup S_2}^i$, $[\,1\,]_S^i \cdot [\,a\,]_\emptyset = [\,a\,]_S^i$, $[\,a_1\,]_\emptyset \cdot [\,a_2\,]_S^V = [\,a_1 \cdot a_2\,]_S^V$.

Noise. In all the current schemes of graded encoding systems, noise will grow after operations are done on encodings, *e.g.,* in Equation (1), we have $r', r'' > r_1, r_2$. To deal with noises abstractly, we use a notion of *noise level*. Namely, we consider a sequence $\mathcal{N}_1 \subset \mathcal{N}_2 \subset \cdots \subset \mathcal{N}_\sigma = \mathcal{N}$ for some $\sigma \in \mathbb{N}$ that progressively contains larger elements. We call $\mathcal{N}_j$ the set of possible noises of level j. The noise level will be utilized in the re-randomization algorithm below.

Procedures. A graded encoding system comes equipped with some further procedures:

- $\mathsf{InstGen}(1^\lambda, \kappa, \nu) \to (\mathsf{param}, \mathsf{esk})$. Instance Generation algorithm takes a security parameter λ, a multi-linearity level κ, and a subring dimension ν as inputs, and outputs public parameter param and encoding secret key esk.

- $\mathsf{PrivEncode}(\mathsf{param}, \mathsf{esk}, S, V, a) \to [\,a\,]_S^V$. Private Encoding algorithm takes $\mathsf{param}, \mathsf{esk}$, a level-index set $S \subseteq [1,\kappa]$, a subring-index set $V \in [1,\nu]$, and $a \in \mathcal{R}$ as inputs, and outputs $[\,a\,]_S^V$.

- $\mathsf{Samp}(\mathsf{param}) \to [\,a\,]_\emptyset$. Ring Sampling algorithm takes param as an input, and outputs a level-$\emptyset$ encoding of a uniformly random element $a \in_R \mathcal{R}$.

- $\mathsf{Encode}(\mathsf{param}, [\,1\,]_S^V, [\,a\,]_\emptyset) \to [\,a\,]_S^V$. Encoding algorithm simply computes $[\,1\,]_S^V \cdot [\,a\,]_\emptyset = [\,a\,]_S^V$.

- $\mathsf{ReRand}(\mathsf{param}, \eta, S, [\,a; r\,]_S) \to [\,a; r'\,]_S$. Re-randomization algorithm takes param, a noise level $\eta \in [1, \sigma]$, a level-index set $S \subseteq [1, \kappa]$, and an encoding $[\,a; r\,]_S$, where $r \in \mathcal{N}_j$ for some $j < \eta$, as inputs. It outputs another encoding $[\,a; r'\,]_S$ of the same element, where $r' \in \mathcal{N}_\eta$. We require that if $\mathsf{ReRand}(\mathsf{param}, \eta, S, [\,a; r_1\,]_S) \to [\,a; r'_1\,]_S$ and $\mathsf{ReRand}(\mathsf{param}, \eta, S, [\,a; r_2\,]_S) \to [\,a; r'_2\,]_S$, then r'_1 and r'_2 distribute statistically identically.

- $\mathsf{IsZero}(\mathsf{param}, [\,a\,]_{[1,\kappa]}) \to \{0, 1\}$. Zero-Test algorithm takes param and a level-$[1, \kappa]$ encoding $[\,a\,]_{[1,\kappa]}$, and outputs 1 if and only if $a = 0$.

- $\mathsf{Ext}(\mathsf{param}, [\,a\,]_{[1,\kappa]}) \to K \in \{0, 1\}^\lambda$. Extraction algorithm takes param and a level-$[1, \kappa]$ encoding $[\,a\,]_{[1,\kappa]}$, and outputs a uniformly random string $K \in \{0, 1\}^\lambda$. We have two properties. First, for any $r_1, r_2 \in \mathcal{N}$, it holds that $\mathsf{Ext}(\mathsf{param}, [\,a; r_1\,]_{[1,\kappa]}) = \mathsf{Ext}(\mathsf{param}, [\,a; r_2\,]_{[1,\kappa]})$. Second, for any $a \in \mathcal{R}$ and $i \in [1, \nu]$, it holds that if $b \in_R \mathbb{Z}_{N_i}$ then $\mathsf{Ext}(\mathsf{param}, [\,a\,]_{[1,\kappa]} + [\,b\,]^i_{[1,\kappa]})$ is almost uniform.

Remark on the Encoding Procedure. We note that, unless possessing esk, without $[\,1\,]^V_S$, one cannot encode to level-S for subring element modulo N_V. We will explicitly publish these elements exactly for required levels and subrings in the public key for our ABE schemes.

Remark on the Re-randomization Procedure. We also remark that ReRand can re-randomize encodings for any level S. This is due to "re-randomizers" $\xi_i = \left\{ \xi_{i,j} = [\,0; r_{i,j}\,]_{\{i\}} \;\middle|\; j \in [1, \upsilon] \right\}$ for all $i \in [1, \kappa]$ internally presented in param, for some sufficiently large υ. To re-randomize $[\,a; r\,]_S$, one simply adds it with the product of random subset-sum for each singleton level in S: $\prod_{i \in S} \sum_{j \in [1,\upsilon]} b_{i,j} \xi_{i,j}$. But this might be inefficient and may require more sophisticated distribution of the random coefficients $b_{i,j}$ than the original CLT construction. A more efficient and ready-to-use method is to directly publish also a "pre-computed" re-randomizer set for level-S: $\xi_S = \{\, \xi_{S,j} = [\,0; r_{S,j}\,]_S \mid j \in [1, \upsilon] \,\}$. In our schemes and assumptions, we assume that whenever $[\,1\,]^V_S$ appears, we also implicitly have a re-randomizer set for level S (*e.g.,* in PK).

Simplifying Noise Levels. In every algorithm of ABE, we will re-randomize the encodings of elements by artificially adding noise to a certain pre-defined level. This is to enforce the distributions of elements in the scheme and the simulation to be the same. For simplicity of exposition, we use only two levels: $\mathcal{N}_1$ for $\mathsf{PK}, \mathsf{MSK}$ and $\mathcal{N}_2$ for CT, SK.

5 Assumptions

In this section, we introduce new assumptions. All are non-interactive and falsifiable assumptions.

Definition 2 (SD1)**.** Let $\mathsf{InstGen}(1^\lambda, 3\ell, 3) \to (\mathsf{param}, \mathsf{esk})$. Let $z, c \xleftarrow{\$} \mathcal{R}$. The Subgroup Decision Assumption 1 states that the following distributions are computationally indistinguishable:

$$\left(D, Z = [\,z\,]^1_{[1,\ell+1]}\right) \qquad \text{and} \qquad \left(D, Z = [\,z\,]^{1,2}_{[1,\ell+1]}\right),$$

where $D = \left(\mathsf{param}, \left\{[\,1\,]^1_{\{i\}}, [\,1\,]^3_{\{i\}}\right\}_{i \in [1,3\ell]}, B = [\,b\,]^{1,2}_{[\ell+2,3\ell]}\right)$.

Definition 3 (SD2)**.** Let $\mathsf{InstGen}(1^\lambda, 3\ell, 3) \to (\mathsf{param}, \mathsf{esk})$. Let $a, b, c \xleftarrow{\$} \mathcal{R}$. For $i \in [\ell + 2, 3\ell]$, let $z_i \xleftarrow{\$} \mathcal{R}$. The Subgroup Decision Assumption 2 states that the following distributions are computationally indistinguishable:

$$\left(D, Z = \left\{[\,z_i\,]^{1,2}_{\{i\}}\right\}_{i \in [\ell+2,3\ell]}\right) \qquad \text{and} \qquad \left(D, Z = \left\{[\,z_i\,]^{1,2,3}_{\{i\}}\right\}_{i \in [\ell+2,3\ell]}\right),$$

where $D = \left(\mathsf{param}, \left\{[\,1\,]^1_{\{i\}}, [\,1\,]^3_{\{i\}}\right\}_{i \in [1,3\ell]}, A = [\,a\,]^{1,2}_{[1,\ell+1]}, B = [\,b\,]^{1,2}_{[\ell+2,3\ell]}, C = [\,c\,]^{2,3}_{[\ell+2,3\ell]}\right)$.

The above two assumptions are naturally generalized from Subgroup Decision assumptions in bilinear groups (the First and Second assumptions in [20]). Its generic hardness should be immediate. The next two assumptions are similar to the Multi-linear DDH assumption (MDDH) [6, 8, 7] but with more given elements. We first describe the simpler of the two:

Definition 4 (ℓ-$\mathsf{EMDDH2}$)**.** Let $\mathsf{InstGen}(1^\lambda, 3\ell, 3) \to (\mathsf{param}, \mathsf{esk})$. Sample $z, c_1, \cdots, c_{\ell+1}$, and ζ from $\mathcal{R}$. The Expanded Multi-linear Decisional Diffie-Hellman Assumption 2 states that the following distributions are computationally indistinguishable:

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

where D consists of: param, $\Big\{[\,1\,]^1_{\{i\}}, [\,1\,]^2_{\{i\}}, [\,1\,]^3_{\{i\}}\Big\}_{i\in[1,3\ell]}$ and $[\,z\,]^2_{[1,\ell+1]}, [\,c_1 z\,]^2_{[\ell+2,3\ell]}$,

$$[\,c_1\,]^2_{[1,\ell+1]}, [\,c_1\,]^2_{[\ell+2,2\ell+1]}, [\,c_1\,]^2_{\{2\ell+2\}}, \ldots, [\,c_1\,]^2_{\{3\ell\}}, \qquad [\,c_2\,]^2_{[\ell+2,2\ell+1]}, [\,c_3\,]^2_{\{2\ell+2\}}, \ldots, [\,c_{\ell+1}\,]^2_{\{3\ell\}}.$$

The $\mathsf{EMDDH2}$ assumption extends the regular Multi-linear DDH (in asymmetric settings)[3] by giving 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 level $\subset [\ell+2, 3\ell]$.

Definition 5 ((ℓ,m)-$\mathsf{EMDDH1}$)**.** Let $\mathsf{InstGen}(1^\lambda, 3\ell, 3) \to (\mathsf{param}, \mathsf{esk})$. 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 $\mathcal{R}$.[4] The (ℓ,m)-Expanded Multi-linear Decisional Diffie-Hellman Assumption 1 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: param, $\big\{[\,1\,]^1_S, [\,1\,]^2_S, [\,1\,]^3_S, \big\}_{S\in\mathcal{S}=\big\{[1,\ell+1],[\ell+2,2\ell+1],\{2\ell+2\},\{2\ell+3\},\ldots,\{3\ell\}\big\}}$ (also with re-randomizers for level index 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

$$\begin{array}{rlrl}
\forall_{e\in\{0,-1\}} & [\,\mu_i a^e_{i,j}\,]^2_{\{i\}}, & & [\,\dfrac{1}{\mu_1\cdots\mu_\ell} z\,]^2_{\{\ell+1\}}, \\
\forall_{e\in\{0,1\}} & [\,\nu_i a^e_{i,j} d_{i,j}\,]^2_{\{i\}}, & & [\,\dfrac{1}{\nu_1\cdots\nu_\ell} c_1\,]^2_{\{\ell+1\}}, \\
\forall_{e\in\{0,-1\}} & [\,\omega_i a^e_{i,j}\,]^2_{\{i\}}, & & [\,\dfrac{1}{\omega_1\cdots\omega_{i-1}\omega_{i+1}\cdots\omega_\ell} zv\dfrac{1}{a_{i,j}}\,]^2_{\{i,\ell+1\}}, \\
\forall_{(e,e')\in\mathsf{E}} & [\,\dfrac{a^e_{i,j}}{a^{e'}_{i,j'}} d_{i,j}\,]^2_{\{i\}}, & \forall_{(e,e')\in\mathsf{E}^\star} & [\,zc_1\dfrac{a^e_{i,j}}{a^{e'}_{i,j'}} d_{i,j}\,]^2_{\{i,\ell+1\}}, \\
\forall_{i\in[2,\ell]}\forall_{e\in\{0,1\}} & [\,a^e_{i,j} d_{i,j}\,]^2_{\{\ell+1+i\}}, & & \\
\forall_{e\in\{0,1\}} & [\,c_1\cdots c_i a^e_{i,j} d_{i,j}\,]^2_{\substack{[\ell+2,\ell+1+i]\cup\\ [2\ell+2,2\ell+i]}}, & \forall_{e\in\{0,1\}} & [\,c_1\cdots c_{i+1} a^e_{i,j}\dfrac{d_{i,j}}{d_{i,j'}}\,]^2_{\substack{[\ell+2,\ell+1+i]\cup\\ [2\ell+2,2\ell+i]}}, \\
 & [\,\dfrac{c_2}{d_{1,j}}\,]^2_{[\ell+2,2\ell+1]}, & \forall_{i\in[2,\ell]} & [\,\dfrac{c_{i+1}}{d_{i,j}}\,]^2_{\{2\ell+i\}},
\end{array}$$

where the range for all subscripts (when appears and is not stated otherwise) is: for all $i \in [1,\ell]$, $j \in [1,m]$, and $j' \in [1,m]$ such that $j' \neq j$ (note that if j' appears, j also appears). We denote $\mathsf{E} = \{(0,0),(0,1),(1,0),(1,1),(-1,0)\}$ and $\mathsf{E}^\star = \mathsf{E} \setminus \{(0,0)\} = \{(0,1),(1,0),(1,1),(-1,0)\}$. [5]

[3]More precisely, we should say a variant of MDDH since the target element is in level $[\ell+2,3\ell]$, not the whole $[1,3\ell]$.

[4]The probability that a random element has no multiplicative inverse is negligible since $\mathcal{R}$ has large primes factors.

[5]We abuse notation and define $[2\ell+2, 2\ell+1] = \emptyset$, so that in the second line from the last of D, when $i = 1$, the level index is simply the singleton set $\{\ell+2\}$.

We prove the generic hardness of the EMDDH1 Assumption in Lemma 10 in §A. We note that our assumption (EMDDH1) is complex, but this is in the same vein as assumptions already used for *selectively secure* ABE for *simpler classes* (such as boolean formulae [24] or regular languages [30]). Contrastingly, we will use our assumption for *fully-secure* ABE for the class of *any poly-size circuits.*

An Interpretation of the EMDDH1 Assumption. The EMDDH1 assumption is defined in such a way that only confined multiplicative combinations would be "useful". We observe first that the "generators" (the encodings of 1) are not given out for all singleton levels; only "bundled" levels $[1, \ell+1], [\ell+2, 2\ell+1]$ are given out among level 1 to $2\ell+1$. Therefore, intuitively, encodings of which levels are subset of these must be combined to the whole bundle to make them useful. Such useful combinations are completely listed as follows, where $i \in [1, \ell]$, $j_1, \dots, j_\ell, j'_1, \dots, j'_\ell \in [1, m]$ such that $j'_1 \neq j_1, \dots, j'_\ell \neq j_\ell$:

$$\begin{aligned}
\forall_{e_1,\dots,e_\ell \in \{0,-1\}} \quad & [\, z a_{1,j_1}^{e_1} \cdots a_{\ell,j_\ell}^{e_\ell} \,]^2_{[1,\ell+1]}, \\
\forall_{e_1,\dots,e_\ell \in \{0,1\}} \quad & [\, c_1 a_{1,j_1}^{e_1} \cdots a_{\ell,j_\ell}^{e_\ell} d_{1,j_1} \cdots d_{\ell,j_\ell} \,]^2_{[1,\ell+1]}, \\
\forall_{e_1,\dots,e_{i-1},e_{i+1},\dots,e_\ell \in \{0,-1\}} \quad & [\, z v a_{1,j_1}^{e_1} \cdots a_{i-1,j_{i-1}}^{e_{i-1}} \frac{1}{a_{i,j_i}} a_{i+1,j_{i+1}}^{e_{i+1}} \cdots a_{\ell,j_\ell}^{e_\ell} \,]^2_{[1,\ell+1]}, \\
\forall_{(e_i,e'_i) \in \mathsf{E}^\star} \forall_{\{(e_k,e'_k)\}_{k \in [1,\ell] \smallsetminus \{i\}} \in \mathsf{E}^{\ell-1}} \quad & [\, z c_1 \frac{a_{1,j_1}^{e_1}}{a_{1,j'_1}^{e'_1}} \cdots \frac{a_{\ell,j_\ell}^{e_\ell}}{a_{\ell,j'_\ell}^{e'_\ell}} d_{1,j_1} \cdots d_{\ell,j_\ell} \,]^2_{[1,\ell+1]}, \\
\forall_{e_i,\dots,e_\ell \in \{0,1\}} \quad & [\, c_1 \cdots c_i a_{i,j_i}^{e_i} \cdots a_{\ell,j_\ell}^{e_\ell} d_{i,j_i} \cdots d_{\ell,j_\ell} \,]^2_{[\ell+2,2\ell+i]}, \\
\forall_{e_i,\dots,e_\ell \in \{0,1\}} \quad & [\, c_1 \cdots c_{i+1} a_{i,j_i}^{e_i} \cdots a_{\ell,j_\ell}^{e_\ell} \frac{1}{d_{i,j'_i}} d_{i,j_i} \cdots d_{\ell,j_\ell} \,]^2_{[\ell+2,2\ell+i]}
\end{aligned} \tag{3}$$

Now, to grasp a quick intuition for its hardness, we may see that

$$Z \cdot [\, \frac{z}{b} \,]^2_{[1,\ell+1]}, \; [\, z \,]^2_{[1,\ell+1]}, [\, c_1 d_{1,j_1} \cdots d_{\ell,j_\ell} \,]^2_{[\ell+2,2\ell+1]}, [\, \frac{c_2}{d_{1,j_1}} \,]^2_{[\ell+2,2\ell+1]}, [\, \frac{c_3}{d_{2,j_2}} \,]^2_{\{2\ell+2\}}, \dots, [\, \frac{c_{\ell+1}}{d_{\ell,j_\ell}} \,]^2_{\{3\ell\}}$$

forms a variant of the ℓ-MDDH assumption. Here, the second and third element are from the first and fifth line of (3), respectively. There are also other tuples that form the MDDH-like assumption, hence we may think of it as a parallel version of MDDH (*à la* the Decisional Parallel BDHE [28]).

The combinations in List (3) come separately from each line of the assumption, except that both line 5 and 6 come from combinations of line 5 and 6 of the assumption. For example, from the first line of the assumption, we have $[\, \mu_1 a_{1,j_1}^{e_1} \,]^2_{\{1\}} \cdots [\, \mu_\ell a_{\ell,j_\ell}^{e_\ell} \,]^2_{\{\ell\}} \cdot [\, \frac{1}{\mu_1 \cdots \mu_\ell} z \,]^2_{\{\ell+1\}} = [\, z a_{1,j_1}^{e_1} \cdots a_{\ell,j_\ell}^{e_\ell} \,]^2_{[1,\ell+1]}$. The first four lines are essentially the only combinations that take place in level $[1, \ell+1]$. For these first four lines, it is clear that we can only combine terms from the same line, since we must cancel all "local" variables, such as μ_i on the first line, by the last element on the same line. On the fourth line, we do not have any local variable but they must be combined together anyway since we do not have $[\, 1 \,]^2_{\{i\}}$ for $i \in [1, \ell+1]$ available. (We only have the bundle $[\, 1 \,]^2_{[1,\ell+1]}$). For the third and the fourth line, we must pick exactly one level-$\{i, \ell+1\}$ term (on the right) to obtain level $\ell+1$. This forcefully prevents further picking another level-$\{i\}$ term (on the left), for the same i. Effectively, this enforces including a "hard" term, such as $a_{i,j}^{e}/a_{i,j'}^{e'}$ where $(e, e') \neq (0, 0)$ on the fourth line, into all combinations (of this line). Intuitively, this kind of hard term will be useful for the "mismatch" techniques for the security proof of ABE (see for intuition in §2). For combinations that contain the second bundled level, $[\ell+2, 2\ell+1]$, the assumption defines the elements on the sixth line in such a way that there is a "hole" set of levels that must be filled in so as to obtain the bundle.

The main point is that it is exactly these kind of "aggregated" elements that we would like to utilize in the first place, *e.g.,* for simulating keys relating to each path of the circuits (since intuitively, a path contains much information to be aggregated). However, we cannot afford putting this list as the given part of the assumption, since they consist of *exponential* number of elements.

Definition 6. For an assumption X, we define the advantage function $\mathsf{Adv}^{\mathsf{X}}_{\mathcal{A}}(\lambda)$ for adversary $\mathcal{A}$ in the standard manner: it is the distance $\mathsf{Adv}^{\mathsf{X}}_{\mathcal{A}}(\lambda) := |\Pr[\mathcal{A}(D,Z)=1] - \Pr[\mathcal{A}(D,Z')=1]|$, where Z, Z' refer to the term Z in each distribution in the definition of the assumption X.

6 A Fully Secure Key-Policy ABE Scheme for Circuits

We describe our KP-ABE for circuits in this section. It is based on the selectively-secure KP-ABE of GGHSW [9]. Our scheme can be thought of their variant that is implemented in *asymmetric* and *composite-order* multi-linear maps (graded encoding systems) with some additional terms. Moreover, while the GGHSW implements on $(\ell+1)$-multilinear maps, ours requires 3ℓ-multi-linearity: instead of using all singleton levels $\{1\}, \dots, \{\ell+1\}$, we implement the scheme on encodings of level $[1,\ell+1], [\ell+2, 2\ell+1], \{2\ell+2\}, \dots, \{3\ell\}$. In the scheme description, each of the first two "bundled" levels will be always used as a whole bundle. We only decompose them in the simulation to accommodate the assumption. Additional terms or modified terms that are different from GGHSW are those "head" elements: in a ciphertext, these comprise T_1, T_2, while in a key, these comprise D_1, D_2, defined below. These are for accommodating the embedding of (implicit) master-key hiding in the semi-functional space in the proof. Our scheme description is as follows.

- $\mathsf{Setup}\big(1^\lambda, n, \ell\big) \to (\mathsf{PK}, \mathsf{MSK})$. Set $\kappa = 3\ell$ and $\nu = 3$. Run $\mathsf{InstGen}(1^\lambda, \kappa, \nu) \to (\mathsf{param}, \mathsf{esk})$. Sample $\alpha, h_1, \dots, h_n, \phi_1, \phi_2 \xleftarrow{\$} \mathcal{R}$. By using $\mathsf{PrivEncode}(\mathsf{esk}, \cdot, \cdot, \cdot)$ and then re-randomizing by $\mathsf{ReRand}(\mathsf{param}, 1, \cdot, \cdot)$, it outputs:

$$\mathsf{PK} = \Big(\mathsf{param}, [\,1\,]^1_{[1,\ell+1]}, [\,\alpha\,]^1_{[1,3\ell]}, [\,h_1\,]^1_{[1,\ell+1]} \dots, [\,h_n\,]^1_{[1,\ell+1]}, [\,\phi_1\,]^1_{[1,\ell+1]}, [\,\phi_2\,]^1_{[1,\ell+1]}\Big),$$

$$\mathsf{MSK} = \Big(\mathsf{param}, \big\{[\,1\,]^1_S, [\,1\,]^3_S\big\}_{S \in \mathcal{S} = \big\{[1,\ell+1],[\ell+2,2\ell+1],\{2\ell+2\},\dots,\{3\ell\}\big\}},$$
$$[\,\alpha\,]^{1,2}_{[\ell+2,3\ell]}, [\,h_1\,]^1_{[\ell+2,2\ell+1]} \dots, [\,h_n\,]^1_{[\ell+2,2\ell+1]}, [\,\phi_1\,]^1_{[\ell+2,3\ell]}, [\,\phi_2\,]^1_{[\ell+2,3\ell]}\Big).$$

- $\mathsf{Encrypt}\big(\mathsf{PK}, x \in \{0,1\}^n, M \in \{0,1\}^\lambda\big) \to \mathsf{CT}$. Let $A_x = \{\, j \in [1,n] \mid x_j = 1 \,\}$. Sample $[\,t\,]_\emptyset, [\,s\,]_\emptyset \leftarrow \mathsf{Samp}(\mathsf{param})$. Output a ciphertext $\mathsf{CT} = \big(C_0, C, \{C_j\}_{j \in A_x}, T_1, T_2\big)$ where the message is masked as $C_0 = \mathsf{Ext}(\mathsf{param}, [\,\alpha t\,]^1_{[1,3\ell]}) \oplus M$, and

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

  All these encodings are re-randomized via $\mathsf{ReRand}(\mathsf{param}, 2, \cdot, \cdot)$.

- $\mathsf{KeyGen}\big(\mathsf{MSK}, f \in \mathbb{F}_{n,\ell}\big) \to \mathsf{SK}$. Sample $[\,r\,]_\emptyset$ and $[\,\alpha_w\,]_\emptyset$ for all $w \in \mathsf{Nodes}$ from $\mathsf{Samp}(\mathsf{param})$, and set

$$D_1' = [\,\alpha\,]^{1,2}_{[\ell+2,3\ell]} + [\,\phi_2 r\,]^1_{[\ell+2,3\ell]}, \qquad D_2' = [\,r\,]^1_{[\ell+2,3\ell]}, \qquad D_3' = [\,\phi_1 r - \alpha_{w_{\mathsf{top}}}\,]^1_{[\ell+2,3\ell]}.$$

  Compute the key element $\mathcal{K}_w$ for each $w \in \mathsf{Nodes}$ as follows.

  1. For each input node $w \in \mathsf{Inputs}$ (*i.e.*, $\mathsf{Depth}(w) = 1$), sample $[\,v_w\,]_\emptyset \leftarrow \mathsf{Samp}(\mathsf{param})$. Let $j = \mathsf{Num}(w)$. Compute $\mathcal{K}_w' = (U_w', K_w')$ as

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

  2. For each gate $w \in \mathsf{Gates}$ (*i.e.*, $\mathsf{Depth}(w) > 1$), sample $[\,\ell_w\,]_\emptyset, [\,r_w\,]_\emptyset \leftarrow \mathsf{Samp}(\mathsf{param})$. Let $i = \mathsf{Depth}(w)$. Compute

$$L_w' = [\,\ell_w\,]^1_{\{2\ell+i\}}, \qquad R_w' = [\,r_w\,]^1_{\{2\ell+i\}}.$$

  It then computes:

– If $\mathsf{GateType}(w) = \mathsf{OR}$, then set $\mathcal{K}'_w = (L'_w, R'_w, K'_{w,1}, K'_{w,2})$ where

$$K'_{w,1} = [\,\alpha_w + \alpha_{\mathsf{L}(w)}\ell_w\,]^1_{[\ell+2,2\ell+i]}, \qquad K'_{w,2} = [\,\alpha_w + \alpha_{\mathsf{R}(w)}r_w\,]^1_{[\ell+2,2\ell+i]}.$$

– If $\mathsf{GateType}(w) = \mathsf{AND}$, then set $\mathcal{K}'_w = (L'_w, R'_w, K'_w)$ where

$$K'_w = [\,\alpha_w + \alpha_{\mathsf{L}(w)}\ell_w + \alpha_{\mathsf{R}(w)}r_w\,]^1_{[\ell+2,2\ell+i]}.$$

Then for each element X' in $\mathsf{SK}' := \big(\{\mathcal{K}'_w\}_{w\in\mathsf{Nodes}}, D'_1, D'_2, D'_3\big)$, the algorithm adds a random mask from the subring $\mathbb{Z}_{N_3}$ as follows. Let $S_{X'}$ be the set index of X'. It samples $[\,\gamma_X\,]_\emptyset \leftarrow \mathsf{Samp}(\mathsf{param})$ and sets $X := X' + [\,\gamma_X\,]^3_{S_{X'}}$. All these encodings are re-randomized via $\mathsf{ReRand}(\mathsf{param}, 2, \cdot, \cdot)$. Output the key as $\mathsf{SK} := \big(\{\mathcal{K}_w\}_{w\in\mathsf{Nodes}}, D_1, D_2, D_3\big)$.

- **Decrypt**$(\mathsf{SK}, \mathsf{CT}) \to M$. Assume that $f(x) = 1$, so that the decryption is possible. Compute at each node w such that $f_w(x) = 1$ in the bottom-up manner as follows. More precisely, we show how to compute $E_w := [\,\alpha_w s\,]^1_{[1,2\ell+i]}$, where $i = \mathsf{Depth}(w)$, by induction on i from 1 to ℓ.

1. For each input node $w \in \mathsf{Inputs} = [1, n]$ such that $f_w(x) = 1$, we have $x_w = 1$. Compute[6]

$$\begin{aligned} E_w &= C \cdot K_w - C_w \cdot U_w \\ &= [\,s\,]^1_{[1,\ell+1]} \cdot [\,\alpha_w + h_j v_w\,]^1_{[\ell+2,2\ell+1]} - [\,h_j s\,]^1_{[1,\ell+1]} \cdot [\,v_w\,]^1_{[\ell+2,2\ell+1]} = [\,\alpha_w s\,]^1_{[1,2\ell+1]}, \end{aligned}$$

where $j = \mathsf{Num}(w)$. This effectively proves the base case of the induction statement.

2. For each gate $w \in \mathsf{Gates}$ such that $f_w(x) = 1$, consider the following two cases.

– If $\mathsf{GateType}(w) = \mathsf{OR}$, 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, $E_{\mathsf{L}(w)} = [\,\alpha_{\mathsf{L}(w)}s\,]^1_{[1,2\ell+i-1]}$ by the induction hypothesis, where we note that $\mathsf{Depth}(\mathsf{L}(w)) = i - 1$. Then, compute

$$\begin{aligned} E_w &= C \cdot K_{w,1} - E_{\mathsf{L}(w)} \cdot L_w \\ &= [\,s\,]^1_{[1,\ell+1]} \cdot [\,\alpha_w + \alpha_{\mathsf{L}(w)}\ell_w\,]^1_{[\ell+2,2\ell+i]} - [\,\alpha_{\mathsf{L}(w)}s\,]^1_{[1,2\ell+i-1]} \cdot [\,\ell_w\,]^1_{\{2\ell+i\}} = [\,\alpha_w s\,]^1_{[1,2\ell+i]}. \end{aligned}$$

– If $\mathsf{GateType}(w) = \mathsf{AND}$, then $f_{\mathsf{L}(w)}(x) = 1$ and $f_{\mathsf{R}(w)}(x) = 1$. Hence, we have $E_{\mathsf{L}(w)} = [\,\alpha_{\mathsf{L}(w)}s\,]^1_{[1,2\ell+i-1]}$ and $E_{\mathsf{R}(w)} = [\,\alpha_{\mathsf{R}(w)}s\,]^1_{[1,2\ell+i-1]}$, by the induction hypothesis. Then compute

$$\begin{aligned} E_w &= C \cdot K_w - \big(E_{\mathsf{L}(w)} \cdot L_w + E_{\mathsf{R}(w)} \cdot R_w\big) \\ &= [\,s\,]^1_{[1,\ell+1]} \cdot [\,\alpha_w + \alpha_{\mathsf{L}(w)}\ell_w + \alpha_{\mathsf{R}(w)}r_w\,]^1_{[\ell+2,2\ell+i]} \\ &\quad - \Big([\,\alpha_{\mathsf{L}(w)}s\,]^1_{[1,2\ell+i-1]} \cdot [\,\ell_w\,]^1_{\{2\ell+i\}} + [\,\alpha_{\mathsf{R}(w)}s\,]^1_{[1,2\ell+i-1]} \cdot [\,r_w\,]^1_{\{2\ell+i\}}\Big) \\ &= [\,\alpha_w s\,]^1_{[1,2\ell+i]}. \end{aligned}$$

This concludes the induction. Finally, at the top gate w_{top}, in which $\mathsf{Depth}(w_{\mathsf{top}}) = \ell$, we obtain $E_{w_{\mathsf{top}}} = [\,\alpha_{w_{\mathsf{top}}}s\,]^1_{[1,3\ell]}$. From this, it computes

$$\begin{aligned} &E_{w_{\mathsf{top}}} + C \cdot D_3 + T_1 \cdot D_1 - T_2 \cdot D_2 \\ &= [\,\alpha_{w_{\mathsf{top}}}s\,]^1_{[1,3\ell]} + [\,s\,]^1_{[1,\ell+1]} \cdot [\,\phi_1 r - \alpha_{w_{\mathsf{top}}}\,]^1_{[\ell+2,3\ell]} + [\,t\,]^1_{[1,\ell+1]} \cdot ([\,\alpha\,]^{1,2}_{[\ell+2,3\ell]} + [\,\phi_2 r\,]^1_{[\ell+2,3\ell]}) \\ &\quad - [\,\phi_2 t + \phi_1 s\,]^1_{[1,\ell+1]} \cdot [\,r\,]^1_{[\ell+2,3\ell]} \\ &= [\,\alpha t\,]^1_{[1,3\ell]}, \end{aligned}$$

and computes the message mask $\mathsf{Ext}(\mathsf{param}, [\,\alpha t\,]^1_{[1,3\ell]})$ and obtains M from C_0.

[6] In these following computations, the N_3 components of elements in SK will be all canceled out by orthogonality; and hence we do not write them explicitly for simplicity.

Properties of Our KP-ABE for Circuits. In our KP-ABE, the size of a ciphertext for string x is proportional to the number of 1's in x, the size of a key for circuit f is proportional to the size of circuit f (the number of nodes). Hence, the scheme is said to be succinct. Moreover, it has no bound on circuit size and fan-out, *i.e.,* we can setup a fixed system, and keys for circuits of any size and fan-out can be constructed (as long as they are polynomial sizes, since the key size is linear to the circuit size). Putting it in other words, the public key PK does not depend on circuit size and fan-out. We only require bounds on input length n and depth ℓ. We remark that, however, the assumption will be parameterized by the maximum number of (internal) gates per layer (and hence the circuit size and the maximum fan-out) of circuits for which the adversary issues key queries. We emphasize that this number is not bounded at the system setup. This is analogous to previous ABE systems with unbounded nature [30, 24, 1], of which underlying assumptions are parameterized by sizes of attributes related to the challenge or key queries made by the adversary.

7 Security Proof

We define semi-functional algorithms to be used in the security proof as follows.

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

$$\begin{aligned}
\widehat{\mathsf{PK}} &= \Big([\,1\,]^2_{[1,\ell+1]}, [\,\alpha\,]^2_{[1,3\ell]}, [\,\hat{h}_1\,]^2_{[1,\ell+1]} \ldots, [\,\hat{h}_n\,]^2_{[1,\ell+1]}, [\,\hat{\phi}_1\,]^2_{[1,\ell+1]}, [\,\hat{\phi}_2\,]^2_{[1,\ell+1]}\Big),\\
\widehat{\mathsf{MSK}}_{\mathsf{base}} &= \Big(\, \big\{[\,1\,]^2_S\big\}_{S \in \mathcal{S} = \big\{[1,\ell+1],[\ell+2,2\ell+1],\{2\ell+2\},\ldots,\{3\ell\}\big\}} \,\Big),\\
\widehat{\mathsf{MSK}}_{\mathsf{aux}} &= \Big([\,\hat{h}_1\,]^2_{[\ell+2,2\ell+1]} \ldots, [\,\hat{h}_n\,]^2_{[\ell+2,2\ell+1]}, [\,\hat{\phi}_1\,]^2_{[\ell+2,3\ell]}, [\,\hat{\phi}_2\,]^2_{[\ell+2,3\ell]}\Big),
\end{aligned}$$

  where it samples $\hat{h}_1, \ldots, \hat{h}_n, \hat{\phi}_1, \hat{\phi}_2 \xleftarrow{\$} \mathcal{R}$. Again, all these encodings can be obtained using $\mathsf{PrivEncode}(\mathsf{esk}, \cdot, \cdot, \cdot)$ and then re-randomized using $\mathsf{ReRand}(\mathsf{param}, 1, \cdot, \cdot)$.

- $\mathsf{SFEncrypt}\big(\mathsf{PK}, x \in \{0,1\}^n, M \in \{0,1\}^\lambda, \widehat{\mathsf{PK}}\big) \to \mathsf{CT}$. Let $A_x = \{\, j \in [1,n] \mid x_j = 1 \,\}$. Sample $[\,t\,]_\emptyset, [\,s\,]_\emptyset, [\,\hat{t}\,]_\emptyset, [\,\hat{s}\,]_\emptyset \leftarrow \mathsf{Samp}(\mathsf{param})$. Output a semi-functional ciphertext $\mathsf{CT} = \big(C_0, C, \{C_j\}_{j \in A_x}, T_1, T_2\big)$ where $C_0 = \mathsf{Ext}(\mathsf{param}, [\,\alpha t\,]^1_{[1,3\ell]} + [\,\alpha \hat{t}\,]^2_{[1,3\ell]}) \oplus M$, and

$$\begin{aligned}
C &= [\,s\,]^1_{[1,\ell+1]} + [\,\hat{s}\,]^2_{[1,\ell+1]}, & C_j &= [\,h_j s\,]^1_{[1,\ell+1]} + [\,\hat{h}_j \hat{s}\,]^2_{[1,\ell+1]},\\
T_1 &= [\,t\,]^1_{[1,\ell+1]} + [\,\hat{t}\,]^2_{[1,\ell+1]}, & T_2 &= [\,\phi_2 t + \phi_1 s\,]^1_{[1,\ell+1]} + [\,\hat{\phi}_2 \hat{t} + \hat{\phi}_1 \hat{s}\,]^2_{[1,\ell+1]}.
\end{aligned}$$

  All these encodings are re-randomized via $\mathsf{ReRand}(\mathsf{param}, 2, \cdot, \cdot)$.

- $\mathsf{SFKeyGen}\big(\mathsf{MSK}, f \in \mathbb{F}_{n,\ell}, \widehat{\mathsf{MSK}}_{\mathsf{base}}, \widehat{\mathsf{MSK}}_{\mathsf{aux}}, \mathsf{type} \in \{1,2,3\}, [\,\beta\,]_\emptyset\big) \to \mathsf{SK}$. It takes additional inputs $\widehat{\mathsf{MSK}}_{\mathsf{base}}, \widehat{\mathsf{MSK}}_{\mathsf{aux}}, \mathsf{type}$ and $[\,\beta\,]_\emptyset$, which is a level-$\emptyset$ encoding of some $\beta \in \mathcal{R}$. Sample $[\,r\,]_\emptyset, [\,\hat{r}\,]_\emptyset \leftarrow \mathsf{Samp}(\mathsf{param})$. It then sets

$$D_1' = \begin{cases}
[\,\alpha\,]^{1,2}_{[\ell+2,3\ell]} + [\,\phi_2 r\,]^1_{[\ell+2,3\ell]} + [\qquad \hat{\phi}_2 \hat{r}\,]^2_{[\ell+2,3\ell]} & \text{if } \mathsf{type} = 1,\\
[\,\alpha\,]^{1,2}_{[\ell+2,3\ell]} + [\,\phi_2 r\,]^1_{[\ell+2,3\ell]} + [\,\beta + \hat{\phi}_2 \hat{r}\,]^2_{[\ell+2,3\ell]} & \text{if } \mathsf{type} = 2,\\
[\,\alpha\,]^{1,2}_{[\ell+2,3\ell]} + [\,\phi_2 r\,]^1_{[\ell+2,3\ell]} + [\,\beta \qquad\,]^2_{[\ell+2,3\ell]} & \text{if } \mathsf{type} = 3.
\end{cases}$$

  Next, we consider the two following cases:

  – If $\mathsf{type} = 3$, then we compute the remaining elements as in KeyGen.

– If $\mathsf{type} = 1$ or 2, then we compute the remaining elements as follows. It samples $[\,\alpha_w\,]_\emptyset, [\,\hat{\alpha}_w\,]_\emptyset \leftarrow \mathsf{Samp}(\mathsf{param})$ for all $w \in \mathsf{Nodes}$. It sets

$$D'_2 = [\,r\,]^1_{[\ell+2,3\ell]} + [\,\hat{r}\,]^2_{[\ell+2,3\ell]}, \qquad D'_3 = [\,\phi_1 r - \alpha_{w_{\mathsf{top}}}\,]^1_{[\ell+2,3\ell]} + [\,\hat{\phi}_1 \hat{r} - \hat{\alpha}_{w_{\mathsf{top}}}\,]^2_{[\ell+2,3\ell]}.$$

Compute the key element $\mathcal{K}_w$ for each $w \in \mathsf{Nodes}$ as follows.

1. For each input node $w \in \mathsf{Inputs}$ (*i.e.*, $\mathsf{Depth}(w) = 1$), sample $[\,v_w\,]_\emptyset, [\,\hat{v}_w\,]_\emptyset \leftarrow \mathsf{Samp}(\mathsf{param})$. Let $j = \mathsf{Num}(w)$. Compute $\mathcal{K}'_w = (U'_w, K'_w)$ as

$$U'_w = [\,v_w\,]^1_{[\ell+2,2\ell+1]} + [\,\hat{v}_w\,]^2_{[\ell+2,2\ell+1]}, \quad K'_w = [\,\alpha_w + h_j v_w\,]^1_{[\ell+2,2\ell+1]} + [\,\hat{\alpha}_w + \hat{h}_j \hat{v}_w\,]^2_{[\ell+2,2\ell+1]}.$$

2. For each gate $w \in \mathsf{Gates}$ (*i.e.*, $\mathsf{Depth}(w) > 1$), sample $[\,\ell_w\,]_\emptyset, [\,r_w\,]_\emptyset, [\,\hat{\ell}_w\,]_\emptyset, [\,\hat{r}_w\,]_\emptyset$ from $\mathsf{Samp}(\mathsf{param})$. Let $i = \mathsf{Depth}(w)$. Compute

$$L'_w = [\,\ell_w\,]^1_{\{2\ell+i\}} + [\,\hat{\ell}_w\,]^2_{\{2\ell+i\}}, \qquad R'_w = [\,r_w\,]^1_{\{2\ell+i\}} + [\,\hat{r}_w\,]^2_{\{2\ell+i\}}.$$

– If $\mathsf{GateType}(w) = \mathsf{OR}$, then set $\mathcal{K}'_w = (L'_w, R'_w, K'_{w,1}, K'_{w,2})$ where

$$K'_{w,1} = [\,\alpha_w + \alpha_{\,\mathsf{L}(w)} \ell_w\,]^1_{[\ell+2,2\ell+i]} + [\,\hat{\alpha}_w + \hat{\alpha}_{\,\mathsf{L}(w)} \hat{\ell}_w\,]^2_{[\ell+2,2\ell+i]},$$
$$K'_{w,2} = [\,\alpha_w + \alpha_{\,\mathsf{R}(w)} r_w\,]^1_{[\ell+2,2\ell+i]} + [\,\hat{\alpha}_w + \hat{\alpha}_{\,\mathsf{R}(w)} \hat{r}_w\,]^2_{[\ell+2,2\ell+i]}.$$

– If $\mathsf{GateType}(w) = \mathsf{AND}$, then set $\mathcal{K}'_w = (L'_w, R'_w, K'_w)$ where

$$K'_w = [\,\alpha_w + \alpha_{\,\mathsf{L}(w)} \ell_w + \alpha_{\,\mathsf{R}(w)} r_w\,]^1_{[\ell+2,2\ell+i]} + [\,\hat{\alpha}_w + \hat{\alpha}_{\,\mathsf{L}(w)} \hat{\ell}_w + \hat{\alpha}_{\,\mathsf{R}(w)} \hat{r}_w\,]^2_{[\ell+2,2\ell+i]}.$$

Then for each element, the algorithm does exactly the same as in KeyGen: it adds a random mask from the subring $\mathbb{Z}_{N_3}$ and re-randomizes using $\mathsf{ReRand}(\mathsf{param}, 2, \cdot, \cdot)$.

We say that semi-functional keys with $\beta = 0$ are of *correlated* type, and those with random β are of *uncorrelated* type. Here, we consider the correlation to the message mask element of semi-functional ciphertext: if β is random, it masks $\alpha \bmod N_2$ in the keys. Consequently, type-1 is classified as correlated, while type-2, 3 is uncorrelated.

We note that in computing type 3 semi-functional keys, $\widehat{\mathsf{MSK}}_{\mathsf{aux}}$ is not required as input and that in computing type 1 semi-functional keys, $[\,\beta\,]_\emptyset$ is not required as inputs. In the proof, we often refer to the additional part in semi-functional elements as semi-functional components and denoted the element with hat, *e.g.*, $\hat{C} = [\,\hat{s}\,]^2_{[1,\ell+1]}$ is semi-functional component of C.

Remark on Relations to the Framework of [1]. The scheme and the semi-functional element structures mostly follow the framework of [1], which shows a generic construction of fully-secure schemes (for a predicate) from underlying primitives called pair encodings (of the same predicate). One advantage of using this framework is that the reduction is tighter than normal dual system proofs: the cost is only $O(q_1)$, instead of $O(q_{\text{all}})$, where q_1, q_{all} are the number of pre-challenge and all key queries respectively. We remark that although the framework of [1] was defined for the case of bilinear maps, we can generalize to work with multi-linear maps. In particular, we use our new Subgroup Decision assumptions that are generalized from those used in [20, 1]. Nevertheless, we do not build a whole new framework for multi-linear maps; instead, we construct our ABE directly.

One deviation from the framework of [1] is that we use a technique from [32] related to the information-theoretic argument for the final transition (to the final game). This has an advantage over [1] in that we can eliminate one assumption (namely, the Third Subgroup Decision assumption in [20, 1]). To enable this, we define D_1 to contain $[\,\alpha\,]^{1,2}_{[\ell+2,3\ell]}$, instead of simply $[\,\alpha\,]^1_{[\ell+2,3\ell]}$ (as would be defined if [1] is used), and define the message mask element of semi-functional ciphertext to be $[\alpha t]^1_{[1,3\ell]} + [\alpha \hat{t}]^2_{[1,3\ell]}$, instead of only $[\alpha t]^1_{[1,3\ell]}$.

Figure 2: The sequence of games in the security proof.

$$\mathsf{G}_0 \quad \text{:Modify}^{(0)} \; \boxed{\mathsf{SFsetup}(1^\lambda, n, \ell) \to (\mathsf{PK}, \mathsf{MSK}, \widehat{\mathsf{PK}}, \widehat{\mathsf{MSK}}_{\mathsf{base}}, \widehat{\mathsf{MSK}}_{\mathsf{aux}})}.$$

$$\text{Modify}^{(2)} \; \boxed{\mathsf{CT}^\star \leftarrow \mathsf{SFEncrypt}(\mathsf{PK}, x^\star, M_{\mathfrak{b}}, \widehat{\mathsf{PK}})}.$$

$$\mathsf{G}_{k,1} \quad \text{:Modify}^{(1)} \; \boxed{[\,\beta_j\,]_\emptyset \leftarrow \mathsf{Samp}(\mathsf{param}),\ \mathsf{SK}_j \leftarrow \begin{cases} \mathsf{SFKeyGen}(\mathsf{MSK}, f^{(j)}, \widehat{\mathsf{MSK}}_{\mathsf{base}}, \quad - \quad, 3, [\,\beta_j\,]_\emptyset) & \text{if } j < k \\ \mathsf{SFKeyGen}(\mathsf{MSK}, f^{(j)}, \widehat{\mathsf{MSK}}_{\mathsf{base}}, \widehat{\mathsf{MSK}}_{\mathsf{aux}}, 1, \quad - \quad) & \text{if } j = k \\ \quad \mathsf{KeyGen}(\mathsf{MSK}, f^{(j)}) & \text{if } j > k \end{cases}}$$

$$\mathsf{G}_{k,2} \quad \text{:Modify}^{(1)} \; \boxed{[\,\beta_j\,]_\emptyset \leftarrow \mathsf{Samp}(\mathsf{param}),\ \mathsf{SK}_j \leftarrow \begin{cases} \mathsf{SFKeyGen}(\mathsf{MSK}, f^{(j)}, \widehat{\mathsf{MSK}}_{\mathsf{base}}, \quad - \quad, 3, [\,\beta_j\,]_\emptyset) & \text{if } j < k \\ \mathsf{SFKeyGen}(\mathsf{MSK}, f^{(j)}, \widehat{\mathsf{MSK}}_{\mathsf{base}}, \widehat{\mathsf{MSK}}_{\mathsf{aux}}, 2, [\,\beta_j\,]_\emptyset) & \text{if } j = k \\ \quad \mathsf{KeyGen}(\mathsf{MSK}, f^{(j)}) & \text{if } j > k \end{cases}}$$

$$\mathsf{G}_{k,3} \quad \text{:Modify}^{(1)} \; \boxed{[\,\beta_j\,]_\emptyset \leftarrow \mathsf{Samp}(\mathsf{param}),\ \mathsf{SK}_j \leftarrow \begin{cases} \mathsf{SFKeyGen}(\mathsf{MSK}, f^{(j)}, \widehat{\mathsf{MSK}}_{\mathsf{base}}, \quad - \quad, 3, [\,\beta_j\,]_\emptyset) & \text{if } j \le k \\ \quad \mathsf{KeyGen}(\mathsf{MSK}, f^{(j)}) & \text{if } j > k \end{cases}}$$

$$\mathsf{G}_{q_1+1}\text{:Modify}^{(3)} \; \boxed{\mathsf{SK}_j \leftarrow \quad \mathsf{SFKeyGen}(\mathsf{MSK}, f^{(j)}, \widehat{\mathsf{MSK}}_{\mathsf{base}}, \widehat{\mathsf{MSK}}_{\mathsf{aux}}, 1, \quad - \quad)}$$

$$\mathsf{G}_{q_1+2}\text{:Insert} \; \boxed{[\,\beta\,]_\emptyset \quad \leftarrow \mathsf{Samp}(\mathsf{param})} \text{ at the begin of Phase 2.}$$

$$\text{Modify}^{(3)} \; \boxed{\mathsf{SK}_j \leftarrow \quad \mathsf{SFKeyGen}(\mathsf{MSK}, f^{(j)}, \widehat{\mathsf{MSK}}_{\mathsf{base}}, \widehat{\mathsf{MSK}}_{\mathsf{aux}}, 2, [\,\beta\,]_\emptyset\ )}$$

$$\mathsf{G}_{q_1+3}\text{:Modify}^{(3)} \; \boxed{\mathsf{SK}_j \leftarrow \quad \mathsf{SFKeyGen}(\mathsf{MSK}, f^{(j)}, \widehat{\mathsf{MSK}}_{\mathsf{base}}, \quad - \quad, 3, [\,\beta\,]_\emptyset\ )}$$

$$\mathsf{G}_{\mathrm{final}} \text{ :Modify}^{(2)} \; \boxed{M \xleftarrow{\$} \{0,1\}^\lambda,\ \mathsf{CT}^\star \leftarrow \mathsf{SFEncrypt}(\mathsf{PK}, x^\star, M, \widehat{\mathsf{PK}})}.$$

7.1 Security Theorem and Proof Overview

We obtain the following security theorem for our KP-ABE for circuits. We recall that $\mathbb{F}_{n,\ell}$ is the class of polynomial size circuits with bounded input-size n and bounded depth ℓ.

Theorem 1. *Suppose that the* SD1, SD2, EMDDH1, EMDDH2 *Assumptions hold. Then our KP-ABE for circuits is fully secure. More precisely, for any PPT adversary $\mathcal{A}$ that attacks our KP-ABE for circuits in $\mathbb{F}_{n,\ell}$, there exist PPT algorithms $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3, \mathcal{B}_4$, whose running times are that of $\mathcal{A}$ plus some polynomial times, such that for any λ,*

$$\mathsf{Adv}_{\mathcal{A}}^{(n,\ell)\text{-}\mathsf{KPABE}}(\lambda) \le \mathsf{Adv}_{\mathcal{B}_1}^{\mathsf{SD1}}(\lambda) + (2q_1+2)\mathsf{Adv}_{\mathcal{B}_2}^{\mathsf{SD2}}(\lambda) + q_1\mathsf{Adv}_{\mathcal{B}_3}^{(\ell,m)\text{-}\mathsf{EMDDH1}}(\lambda) + \mathsf{Adv}_{\mathcal{B}_4}^{\ell\text{-}\mathsf{EMDDH2}}(\lambda),$$

where q_1 is the number of key queries by $\mathcal{A}$ in phase 1, and m is the maximum number of (internal) gates per layer of circuits for which $\mathcal{A}$ issues key queries in phase 1.

Security Proof Structure for Theorem 1. We use a sequence of games in the following order:

$$\begin{tikzcd}
\mathsf{G}_{\mathrm{real}} \arrow[r, "\substack{\mathsf{SD1}\\12}"'] & \mathsf{G}_0 \arrow[r] & \mathsf{G}_{1,1} \arrow[r] & \cdots \arrow[r] & \mathsf{G}_{k-1,3} \arrow[r, "\substack{\mathsf{SD2}\\13}"'] & \mathsf{G}_{k,1} \arrow[r, "\substack{\mathsf{EMDDH1}\\3}"'] & \mathsf{G}_{k,2} \arrow[r, "\substack{\mathsf{SD2}\\14}"'] & \mathsf{G}_{k,3} \arrow[r] & \cdots \arrow[r] & \mathsf{G}_{q_1,3} \arrow[r, "\substack{\mathsf{SD2}\\15}"'] & \mathsf{G}_{q_1+1} \arrow[r, "\substack{\mathsf{EMDDH2}\\16}"'] & \mathsf{G}_{q_1+2} \arrow[r, "\substack{\mathsf{SD2}\\17}"'] & \mathsf{G}_{q_1+3} \arrow[r, "\substack{=\\18}"'] & \mathsf{G}_{\mathrm{final}}
\end{tikzcd}$$

where each game is defined as follows. $\mathsf{G}_{\mathrm{real}}$ is the actual security game, and each of the following game is defined exactly as *its previous game* in the sequence except the specified modification that is defined in Fig. 2. For notational purpose, let $\mathsf{G}_{0,3} := \mathsf{G}_0$. In the final game, the advantage of $\mathcal{A}$ is trivially 0. We prove the indistinguishability between all these adjacent games (under the underlying assumptions as written in the diagram). These comprise Lemma 12,13,3,14,15,16,17,18 (also depicted in the diagram). From these lemmata, we obtain Theorem 1.

Outline for the Proof of Each Lemma. We first consider the proofs for game transitions that are based on subgroup decision assumptions ($\mathsf{SD1}, \mathsf{SD2}$): which are the game pair $\mathsf{G}_{\mathrm{real}}/\mathsf{G}_0$, $\mathsf{G}_{k-1,3}/\mathsf{G}_{k,1}$, $\mathsf{G}_{k,2}/\mathsf{G}_{k,3}$, $\mathsf{G}_{q_1,3}/\mathsf{G}_{q_1+1}$, $\mathsf{G}_{q_1+2}/\mathsf{G}_{q_1+3}$. Although some of these proofs are lengthy, they all share the same idea. We thus summarize their proof sketch together below (§7.2) and postpone each of the full proofs to §B (Lemma 12,13,14,15,17 respectively).

The most non-trivial proofs are indistinguishability of $\mathsf{G}_{k,1}/\mathsf{G}_{k,2}$ and of $\mathsf{G}_{q_1+1}/\mathsf{G}_{q_1+2}$. Both transitions switch semi-functional keys from type-1 (the correlated type) to type 2 (an uncorrelated type). When the switching occurs in the phase 1 (corresponding to game $\mathsf{G}_{k,1}/\mathsf{G}_{k,2}$), we will essentially use the *co-selective* security techniques. We will not, however, explicitly show it in a modular manner here. Instead, we reduce the indistinguishability directly to the assumption ($\mathsf{EMDDH1}$). Nevertheless, the proof will exhibit clearly the nature of co-selective type of proof. That is, the adversary $\mathcal{A}$ first announces circuit f, the reduction algorithm $\mathcal{B}$ then simulates parameters (in semi-functional space) and SK_f. The key is then returned to $\mathcal{A}$. After then, $\mathcal{A}$ in the challenge phase will ask for ciphertext for $x^\star$, and $\mathcal{B}$ simulates it in the consistent way with SK_f based on the simulated parameters. The proof for co-selective security of ABE for circuits is the main novelty in this paper. Hence, we show the full proof in the paper body below in §7.3.

On the other hand, if the switching occurs in the phase 2 (corresponding to game $\mathsf{G}_{q_1+1}/\mathsf{G}_{q_1+2}$), we will essentially, but again implicitly, use the *selective* security techniques, and hence we can follow from the selective security proof of the GGHSW KP-ABE [9], upon which our scheme is constructed. We prove it directly to the ℓ-EMDDH2 assumption. We postpone this proof to §B.5.

Finally, it is easy to see that G_{q_1+3} is the same as $\mathsf{G}_{\mathrm{final}}$, due to the fact that all the semi-functional keys become those of uncorrelated type. We clarify the detail in Lemma 18 in §B.7.

We define $\mathsf{G}_j\mathsf{Adv}_{\mathcal{A}}^{(n,\ell)\text{-}\mathsf{KPABE}}(\lambda)$ to be the advantage of $\mathcal{A}$ in the game G_j.

7.2 Sketch of Proofs for Subgroup-Decision Based Transitions

Lemma 2 (informal)**.** *Consider pairs of games:* $\mathsf{G}_{\mathrm{real}}/\mathsf{G}_0$, $\mathsf{G}_{k-1,3}/\mathsf{G}_{k,1}$, $\mathsf{G}_{k,2}/\mathsf{G}_{k,3}$, $\mathsf{G}_{q_1,3}/\mathsf{G}_{q_1+1}$, $\mathsf{G}_{q_1+2}/\mathsf{G}_{q_1+3}$. *If there exists an adversary $\mathcal{A}$ that has non-negligible difference in advantage between the two games in question, then we can construct an algorithm $\mathcal{B}$ that breaks* $\mathsf{SD1}$ *in the first case, and* $\mathsf{SD2}$ *in all the other four cases.*

Proof Sketch. The algorithm $\mathcal{B}$ obtains an input (D, Z) from the underlying assumption ($\mathsf{SD1}$ in the first case, $\mathsf{SD2}$ in all the other cases). $\mathcal{B}$ will use the adversary $\mathcal{A}$ against either of the two games in question as subroutines to answer whether Z has $\mathbb{Z}_{N_2}$ component being zero or random. All the proofs simulate the setup phase in exactly the same way. First, $\mathcal{B}$ samples $[\,\tilde{\alpha}\,]_\emptyset \leftarrow \mathsf{Samp}(\mathsf{param})$ and sets $[\,\alpha\,]^{1,2}_{[\ell+2,3\ell]} = [\,\tilde{\alpha}\,]_\emptyset \cdot [\,b\,]^{1,2}_{[\ell+2,3\ell]}$ for MSK, and $[\,\alpha\,]^1_{[1,3\ell]} = [\,\alpha\,]^{1,2}_{[\ell+2,3\ell]} \cdot [\,1\,]^1_{[1,\ell+1]}$ for PK. The other terms in PK are trivially computed: for each variable y in $H := \{h_1, \ldots, h_n, \phi_1, \phi_2\}$, $\mathcal{B}$ samples $[\,\tilde{y}\,]_\emptyset \leftarrow \mathsf{Samp}(\mathsf{param})$ and computes $[\,y\,]^1_S = [\,\tilde{y}\,]_\emptyset \cdot [\,1\,]^1_S$ for every needed term in $\mathsf{PK}, \mathsf{MSK}$. Note that $[\,1\,]^1_S$ is computable from $[\,1\,]^1_{\{i\}}, [\,1\,]^3_{\{i\}}$ for all $i \in [1, 3\ell]$. On the other hand, $\mathcal{B}$ cannot simulate $\widehat{\mathsf{PK}}, \widehat{\mathsf{MSK}}_{\mathsf{base}}, \widehat{\mathsf{MSK}}_{\mathsf{aux}}$ since it does not possess any $[\,1\,]^2_S$. However, we *implicitly* define $[\,\hat{y}\,]^2_S = [\,\tilde{y}\,]_\emptyset \cdot [\,1\,]^2_S$ for each variable $\hat{y} \in \hat{H} := \{\hat{h}_1, \ldots, \hat{h}_n, \hat{\phi}_1, \hat{\phi}_2\}$. Due to CRT, $y = \tilde{y} \bmod N_1$ and $\hat{y} = \tilde{y} \bmod N_2$ is independent as required. This allows $\mathcal{B}$ to simulate ciphertexts and keys by multiplying $[\,\tilde{y}\,]_\emptyset$ with, say F being $[\,X\,]^1_S$ or $[\,X\,]^{1,2}_S$ (for some X). If $F = [\,X\,]^1_S$, then $[\,\tilde{y}\,]_\emptyset \cdot F = [\,yX\,]^1_S$ produces a *normal* element (whether ciphertext or key depends on contexts), while if $F = [\,X\,]^{1,2}_S$, then $[\,\tilde{y}\,]_\emptyset \cdot F = [\,yX\,]^1_S + [\,\hat{y}\hat{X}\,]^2_S$ produces a *semi-functional* element, with $\hat{X} = X \bmod N_2$. Hence, if F is the problem instance, which is the term Z from $\mathsf{SD1}, \mathsf{SD2}$, we can use it to simulate the game transitions which modify normal components to semi-functional ones. On the other hand, if F is an given known term, which is A, B, C in the assumptions, we can use it to simulate corresponding element types (whether normal or semi-functional depends on contexts). In Table 1, we summarize

Table 1: Summary of proofs for subgroup-decision based transitions (for Lemma 2).

Between	Difference	Assumption	MSK	SK_j in phase 1			CT	SK_j in phase 2
				$(j<k)$	$(j=k)$	$(j>k)$		$(\forall j)$
$\mathsf{G}_{\mathrm{real}}$ / G_0	CT = normal/semi	$\mathsf{SD1}$	B		✓		Z	✓
$\mathsf{G}_{k-1,3}$/$\mathsf{G}_{k,1}$	SK_k = normal/semi-1	$\mathsf{SD2}$	B	C	Z	✓	A	✓
$\mathsf{G}_{k,2}$ / $\mathsf{G}_{k,3}$	SK_k = semi-2 /semi-3	$\mathsf{SD2}$	B	C	Z, C	✓	A	✓
$\mathsf{G}_{q_1,3}$ / G_{q_1+1}	$\mathsf{SK}_{\forall j}$= normal/semi-1	$\mathsf{SD2}$	B		C		A	Z
G_{q_1+2} / G_{q_1+3}	$\mathsf{SK}_{\forall j}$= semi-2 /semi-3	$\mathsf{SD2}$	B		C		A	Z, C

Table 2: Overview for Simulation of Elements in Lemma 3

Element	For	Used Terms from the $\mathsf{EMDDH1}$ Assumption	Note
$[\,\hat{h}_j\,]^2_{[\ell+2,2\ell+1]}$	SK	$[\,c_1 a^{e_1}_{1,j_1}\cdots a^{e_\ell}_{\ell,j_\ell} d_{1,j_1}\cdots d_{\ell,j_\ell}\,]^2_{[\ell+2,2\ell+1]}$	
$[\,\hat{\alpha}_w\,]^2_{[\ell+2,2\ell+i]}$	SK	$[\,c_1\cdots c_i a^{e_i}_{i,j_i}\cdots a^{e_\ell}_{\ell,j_\ell} d_{i,j_i}\cdots d_{\ell,j_\ell}\,]^2_{[\ell+2,2\ell+i]}$	$\mathsf{Depth}(w) = i-1$
$[\,\hat{\alpha}_w + \hat{\alpha}_{\mathsf{L}(w)}\hat{\ell}_w\,]^2_{[\ell+2,2\ell+i]}$	SK	$[\,c_1\cdots c_{i+1} a^{e_i}_{i,j_i}\cdots a^{e_\ell}_{\ell,j_\ell}\frac{1}{d_{i,j'_i}} d_{i,j_i}\cdots d_{\ell,j_\ell}\,]^2_{[\ell+2,2\ell+i]}$	$\mathsf{Depth}(w) = i$
$[\,\hat{\ell}_w\,]^2_{\{2\ell+i\}}$	SK	$[\,\frac{c_{i+1}}{d_{i,j}}\,]^2_{\{2\ell+i\}}$	$\mathsf{Depth}(w) = i$
$[\,\hat{s}\,]^2_{[1,\ell+1]}$	CT	$[\,z a^{e_1}_{1,j_1}\cdots a^{e_\ell}_{\ell,j_\ell}\,]^2_{[1,\ell+1]}$	
$[\,\hat{h}_j\,]^2_{[1,\ell+1]}$	CT	$[\,c_1 a^{e_1}_{1,j_1}\cdots a^{e_\ell}_{\ell,j_\ell} d_{1,j_1}\cdots d_{\ell,j_\ell}\,]^2_{[1,\ell+1]}$	
$[\,\hat{\phi}_2\hat{t} + \hat{\phi}_1\hat{s}\,]^2_{[1,\ell+1]}$	CT	$[\,z v a^{e_1}_{1,j_1}\cdots a^{e_{i-1}}_{i-1,j_{i-1}}\frac{1}{a_{i,j_i}} a^{e_{i+1}}_{i+1,j_{i+1}}\cdots a^{e_\ell}_{\ell,j_\ell}\,]^2_{[1,\ell+1]}$	
$[\,s\hat{h}_j\,]^2_{[1,\ell+1]}$	CT	$[\,z c_1 \frac{a^{e_1}_{1,j_1}}{a^{e'_1}_{1,j'_1}}\cdots\frac{a^{e_\ell}_{\ell,j_\ell}}{a^{e'_\ell}_{\ell,j'_\ell}} d_{1,j_1}\cdots d_{\ell,j_\ell}\,]^2_{[1,\ell+1]}$	$\exists i : (e_i, e'_i) \neq (0,0)$

how we use elements Z, A, B, C from the assumption $\mathsf{SD1}, \mathsf{SD2}$ to simulate what elements in the five game transitions that is based on $\mathsf{SD1}, \mathsf{SD2}$. ✓ means that we can trivially compute. We note that type-3 keys are exactly like normal keys but with additional $[\,\beta\,]^2_{\ell+2,3\ell}$ in D_1; this can be simulated using C. Also note that transitions of keys in phase 2 differ from those of phase 1 in that instead switching one key at a time, all key queries (in phase 2) are changed altogether at once. $\square$

7.3 Proof for Transition of Type-1 to Type-2 Semi-functional Key in Phase 1

Lemma 3 ($\mathsf{G}_{k,1}$ to $\mathsf{G}_{k,2}$). *For any adversary $\mathcal{A}$ that makes the k-th key query for a circuit of which the maximum number of (internal) gates per layer is m, there exists an algorithm $\mathcal{B}$ that breaks the (ℓ, m)-$\mathsf{EMDDH1}$ with* $|\mathsf{G}_{k,1}\mathsf{Adv}^{(n,\ell)\text{-}\mathsf{KPABE}}_{\mathcal{A}}(\lambda) - \mathsf{G}_{k,2}\mathsf{Adv}^{(n,\ell)\text{-}\mathsf{KPABE}}_{\mathcal{A}}(\lambda)| \leq \mathsf{Adv}^{(\ell,m)\text{-}\mathsf{EMDDH1}}_{\mathcal{B}}(\lambda)$.

Proof Overview. The algorithm $\mathcal{B}$ obtains an input (D, Z) from the (ℓ, m)-$\mathsf{EMDDH1}$ Assumption. Denote $Z = [\,\delta + c_1\cdots c_{\ell+1}b\,]^2_{[\ell+2,3\ell]}$. Its task is to guess whether $\delta = 0$ or $\delta \in_R \mathcal{R}$. The idea is that $\mathcal{B}$ will define $\beta_k = \delta$ in the simulation for the k-th key, so that if $\delta = 0$, it will be a type-1 semi-functional key, otherwise δ is random, and it will be a type-2 semi-functional key. $\mathcal{B}$ will construct the normal component as in the scheme while it will simulate all the semi-functional components by using the assumption. $\mathcal{B}$ will not define all the hatted variables (and hence, $\widehat{\mathsf{PK}}, \widehat{\mathsf{MSK}_{\mathsf{aux}}}$) until the first query that requires using them, which is the k-th key. This resembles to the co-selective security notion. As an overview, we highlight which element in the scheme can be simulated by which term from the assumption (in the form of List (3)). We summarize important terms in Table 2. The intuition basically follows from explanation in §2. We review the main ideas as follows.

- **Chaining Information on Paths.** First, for any input node $j \in [1, n]$, the element $[\,\hat{h}_j\,]^2_{[1,\ell+1]}$ (for ciphertext) and $[\,\hat{h}_j\,]^2_{[\ell+2,2\ell+1]}$ (for key) are simulated by chaining information on all paths from the output gate to it. We use a_{i,j_i} for AND gate w_{i,j_i}. For its left child we use $(1 - a_{i,j_i})$, while for its right child we use a_{i,j_i} as the information to aggregate.

- **Gate Mismatch.** The variable a_{i,j_i} will be also served as "individual randomness" for identifying a gate in the circuit. The purpose is to enforce cancellation of non-trivial terms in $[\,s\hat{h}_j\,]^2_{[1,\ell+1]}$ so that only terms with "gate mismatch" will remain. Intuitively, this is represented by the term $a^{e_i}_{i,j_i}/a^{e'_i}_{i,j'_i}$, where $j_i \neq j'_i, (e_i, e'_i) \neq (0, 0)$, which appears when two gates mismatch.

- **Outgoing-wire Mismatch.** The variable d_{i,j_i} is served as "individual randomness" for identifying an outgoing wire when we gather many paths for chaining. The purpose is to enforce cancellation of non-trivial terms in $[\,\hat{\alpha}_w + \hat{\alpha}_{\mathsf{L}(w)}\hat{\ell}_w\,]^2_{[\ell+2,2\ell+i]}$ so that only terms with "outgoing-wire mismatch" will remain. Intuitively, this is represented by the term $d_{i,j_i}/d_{i,j'_i}$, where $j_i \neq j'_i$, which appears when out-going wires mismatch.

In the proof, we will indeed not use the elements as in the form of List (3), instead we will use more sophisticated recursive algorithms that will multiply the original elements from the assumption on the fly so that the running time are kept efficient.

We are now ready to describe the full proof.

Proof (of Lemma 3). The algorithm $\mathcal{B}$ obtains an input (D, Z) from the (ℓ, m)-$\mathsf{EMDDH1}$ Assumption. Denote $Z = [\,\delta + c_1 \cdots c_{\ell+1} b\,]^2_{[\ell+2,3\ell]}$. Its task is to guess whether $\delta = 0$ or $\delta \in_R \mathcal{R}$. $\mathcal{B}$ will implicitly define $\beta_k = \delta$ in the simulation for the k-th key below.

Setup. The algorithm $\mathcal{B}$ simulates $\mathsf{SFSetup}(1^\lambda, 3\ell, 3)$ as follows. $\mathcal{B}$ will compute $\mathsf{PK}, \mathsf{MSK}, \widehat{\mathsf{MSK}}_{\mathsf{base}}$. We first note that $\mathcal{B}$ can compute $[\,1\,]^1_S, [\,1\,]^2_S, [\,1\,]^3_S$ for any S that is a union of sets from $\mathcal{S}$, where $S \in \mathcal{S} = \{[1, \ell+1], [\ell+2, 2\ell+1], \{2\ell+2\}, \ldots, \{3\ell\}\}$. $\mathcal{B}$ begins by sampling $[\,\alpha\,]_\emptyset \leftarrow \mathsf{Samp}(\mathsf{param})$ and computing $[\,\alpha\,]^1_{[1,3\ell]}, [\,\alpha\,]^{1,2}_{[\ell+2,3\ell]}$, for $\mathsf{PK}, \mathsf{MSK}$, respectively. For each variable y in $H := \{h_1, \ldots, h_n, \phi_1, \phi_2\}$, $\mathcal{B}$ samples $[\,y\,]_\emptyset \leftarrow \mathsf{Samp}(\mathsf{param})$. $\mathcal{B}$ then computes each $[\,y\,]^1_S$ term that appears in $\mathsf{PK}, \mathsf{MSK}$. All these terms are computable and completely define $\mathsf{PK}, \mathsf{MSK}$. PK is given to $\mathcal{A}$. Moreover, $\widehat{\mathsf{MSK}}_{\mathsf{base}}$ is computable from $[\,1\,]^2_S$, for $S \in \mathcal{S}$. We remark that $\mathcal{B}$ will not define all the hatted variables (and hence, $\widehat{\mathsf{PK}}, \widehat{\mathsf{MSK}}_{\mathsf{aux}}$) 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 $f^{(j)}$, $\mathcal{B}$ generates a key as follows

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

[Case $j < k$]. $\mathcal{B}$ samples $[\,\beta_j\,]_\emptyset \leftarrow \mathsf{Samp}(\mathsf{param})$. $\mathcal{B}$ generates a type-3 semi-functional key by setting $\mathsf{SK}_j \leftarrow \mathsf{SFKeyGen}(\mathsf{MSK}, f^{(j)}, \widehat{\mathsf{MSK}}_{\mathsf{base}}, -, 3, [\,\beta_j\,]_\emptyset)$.

[Case $j = k$]. $\mathcal{B}$ generates a type-1 or type-2 semi-functional key as follows. First it generates a normal key $\mathsf{SK}_k \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, f^{(k)})$. A type-1 or type-2 semi-functional key is different from a normal key by having additional N_2 components. To define these components, we begin with programming hatted variables in $\widehat{\mathsf{PK}}, \widehat{\mathsf{MSK}}_{\mathsf{aux}}$ and in semi-functional key components, using the information on the circuit $f^{(k)}$.

(Programming Parameters and Randomness in Key). $\mathcal{B}$ first samples $[\,h'_1\,]_\emptyset, \ldots, [\,h'_n\,]_\emptyset$, $[\,\phi'_1\,]_\emptyset, [\,\phi'_2\,]_\emptyset \leftarrow \mathsf{Samp}(\mathsf{param})$. We implicitly set parameters for $\widehat{\mathsf{PK}}, \widehat{\mathsf{MSK}}_{\mathsf{aux}}$:

$$\hat{\phi}_1 = v + \phi'_1, \qquad \hat{\phi}_2 = vb + \phi'_2. \tag{4}$$

The remaining parameters $\hat{h}_j$'s depend on the key query, which is the circuit $f^{(k)}$. We define them along with programming randomness for simulating the key for the circuit. For $i \in [1, \ell]$, let m_i be the number of nodes of depth i. Note that $m_1 = n$. Let m be the maximum among $m_2, \ldots, m_\ell$. We recall that each node is indexed by its depth and its number in the lexicographical order in that layer: node $w_{i,j}$ has depth i and is the j-th node in the depth-i layer.

We implicitly set $\hat{\alpha}_w$ for all $w \in \mathsf{Nodes}$ and $\hat{h}_j$ for all $j \in [1, n]$ as

$$\hat{\alpha}_{w_{i,j}} = y_{i,j} c_1 \cdots c_{i+1}, \qquad \hat{h}_j = d_{1,1} y_{1,j} c_1 + h'_j. \tag{5}$$

where $y_{i,j}$ is defined recursively from the output gate (of which depth is ℓ) to the input gates (of which depth is 1) as follows. First at the output gate $w_{\mathsf{top}} = w_{\ell,1}$, we set $y_{\ell,1} = 1$.[7] Then, for i ranges from ℓ to 2, and for $j' \in [1, m_{i-1}]$, we set

$$y_{i-1,j'} = \sum_{\iota \in [1,m_i]} \rho_{(i-1,j') \to (i,\iota)} d_{i,\iota} y_{i,\iota} = \rho_{(i-1,j') \to (i,1)} d_{i,1} y_{i,1} + \cdots + \rho_{(i-1,j') \to (i,m_i)} d_{i,m_i} y_{i,m_i}, \tag{6}$$

where for $j \in [1, m_i]$, we also define

$$\rho_{(i-1,j') \to (i,j)} = \begin{cases} 1 & \text{if } \mathsf{GateType}(w_{i,j}) = \mathsf{OR} \text{ and } w_{i-1,j'} \in \{\mathsf{L}(w_{i,j}), \mathsf{R}(w_{i,j})\}, \\ 1 - a_{i,j} & \text{if } \mathsf{GateType}(w_{i,j}) = \mathsf{AND} \text{ and } w_{i-1,j'} = \mathsf{L}(w_{i,j}), \\ a_{i,j} & \text{if } \mathsf{GateType}(w_{i,j}) = \mathsf{AND} \text{ and } w_{i-1,j'} = \mathsf{R}(w_{i,j}), \\ 0 & \text{if } w_{i-1,j'} \notin \{\mathsf{L}(w_{i,j}), \mathsf{R}(w_{i,j})\}. \end{cases} \tag{7}$$

For $w_{i,j} \in \mathsf{Nodes}$, we denote $\mathsf{path}_{(i,j)}$ to be the set of all sequences of node numbers in the path in the circuit between $w_{i,j}$ and $w_{\ell,1}$ (the output gate) inclusively. That is, we say $(j, j_{i+1}, \ldots, j_{\ell-1}, 1) \in \mathsf{path}_{(i,j)}$ iff $(w_{i,j}, w_{i+1,j_{i+1}}, \ldots, w_{\ell-1,j_{\ell-1}}, w_{\ell,1})$ is a path from $w_{i,j}$ to $w_{\ell,1}$. From this, we can derive an explicit form of $y_{i,j}$ for $i \in [1, \ell-1], j \in [1, m_i]$ as

$$y_{i,j} = \sum_{(j,j_{i+1},\ldots,j_{\ell-1},1) \in \mathsf{path}_{(i,j)}} (\rho_{(i,j) \to (i+1,j_{i+1})} d_{i+1,j_{i+1}}) \cdots (\rho_{(\ell-1,j_{\ell-1}) \to (\ell,1)} d_{\ell,1}) \tag{8}$$

For the remaining randomness in key, we set $\beta_k = \delta$, $\hat{r} = \frac{c_1 \cdots c_{\ell+1}}{v}$, and

$$\forall_{w_{1,j} \in \mathsf{Inputs}} : \hat{v}_{w_{1,j}} = -\frac{c_2}{d_{1,1}}, \qquad \forall_{w_{i,j} \in \mathsf{Gates}} : \hat{\ell}_{w_{i,j}} = -\frac{c_{i+1}}{d_{i,j}}, \quad \hat{r}_{w_{i,j}} = -\frac{c_{i+1}}{d_{i,j}}. \tag{9}$$

Note that these values are not properly randomly distributed yet. We separately describe the re-randomization process later for simplicity.

(Simulating Key). From the above (implicit) definitions, the semi-functional components for key are well defined. $\mathcal{B}$ computes them as follows. First, $\mathcal{B}$ trivially sets $\hat{D}_2 = [\,\frac{c_1 \cdots c_{\ell+1}}{v}\,]^2_{[\ell+2,3\ell]}$, $\hat{U}_{w_{1,j}} = -[\,\frac{c_2}{d_{1,1}}\,]^2_{[\ell+2,2\ell+1]}$ for all $w_{1,j} \in \mathsf{Inputs}$, and $\hat{L}_{w_{i,j}} = \hat{R}_{w_{i,j}} = -[\,\frac{c_{i+1}}{d_{i,j}}\,]^2_{\{2\ell+i\}}$ for all $w_{i,j} \in \mathsf{Gates}$. Then $\mathcal{B}$ computes

$$\hat{D}_1 = Z + [\,\phi'_2\,]_\emptyset \cdot [\,\frac{c_1 \cdots c_{\ell+1}}{v}\,]^2_{[\ell+2,3\ell]} = [\,\beta_k + \hat{\phi}_2 \hat{r}\,]^2_{[\ell+2,3\ell]},$$

where we observe that in $Z = [\,\beta + c_1 \cdots c_{\ell+1} b\,]^2_{[\ell+2,3\ell]}$, the last term can be expressed as $(vb)(\frac{c_1 \cdots c_{\ell+1}}{v})$. vb is then combined with ϕ'_2 to produce $\hat{\phi}_2$. Next, $\mathcal{B}$ computes

$$\begin{aligned} \hat{D}_3 \quad &= [\,\phi'_1\,]_\emptyset \cdot [\,\frac{c_1 \cdots c_{\ell+1}}{v}\,]^2_{[\ell+2,3\ell]} = [\,\hat{\phi}_1 \hat{r} - \hat{\alpha}_{w_{\ell,1}}\,]^2_{[\ell+2,3\ell]}, \\ \forall_{w_{1,j} \in \mathsf{Inputs}} \ \hat{K}_{w_{1,j}} &= -[\,h'_j\,]_\emptyset \cdot [\,\frac{c_2}{d_{1,1}}\,]^2_{[\ell+2,2\ell+1]} \quad = [\,\hat{\alpha}_{w_{1,j}} + \hat{h}_j \hat{v}_{w_{1,j}}\,]^2_{[\ell+2,2\ell+1]}, \end{aligned}$$

[7]Note that in this layer, there is only one gate w_{top} but we write its number, 1, for notation consistency.

where in the former, $c_1 \cdots c_\ell$ is canceled out, while in the latter c_1c_2 is canceled out. To keep notation compact in the next computation, for every $i \in [2, \ell]$, $j' \in [1, m_{i-1}]$, $j \in [1, m_i]$, we define

$$\sigma_{(i-1,j'),(i,j)} := \sum_{\substack{\iota \in [1,m_i] \\ \text{s.t. } \iota \neq j}} \rho_{(i-1,j') \to (i,\iota)} \frac{d_{i,\iota}}{d_{i,j}} y_{i,\iota} c_1 \cdots c_{i+1}. \tag{10}$$

For all $w_{i,j} \in \mathsf{Gates}$, we compute corresponding elements as follows. Let $w_{i-1,j_L} = \mathsf{L}(w_{i,j})$, $w_{i-1,j_R} = \mathsf{R}(w_{i,j})$. If $w_{i,j}$ is an OR gate, then $\mathcal{B}$ computes

$$\begin{aligned} \hat{K}_{w_{i,j},1} &= -[\,\sigma_{(i-1,j_L),(i,j)}\,]^2_{[\ell+2,2\ell+i]} = [\,\hat{\alpha}_{w_{i,j}} + \hat{\alpha}_{\mathsf{L}(w_{i,j})} \hat{\ell}_{w_{i,j}}\,]^2_{[\ell+2,2\ell+i]}, \\ \hat{K}_{w_{i,j},2} &= -[\,\sigma_{(i-1,j_R),(i,j)}\,]^2_{[\ell+2,2\ell+i]} = [\,\hat{\alpha}_{w_{i,j}} + \hat{\alpha}_{\mathsf{R}(w_{i,j})} \hat{r}_{w_{i,j}}\,]^2_{[\ell+2,2\ell+i]}. \end{aligned} \tag{11}$$

We verify this by substituting $\hat{\alpha}_{w_{i,j}} = y_{i,j} c_1 \cdots c_{i+1}$, $\hat{\alpha}_{\mathsf{L}(w_{i,j})} = (y_{i-1,j_L} c_1 \cdots c_i)$, $\hat{\ell}_{w_{i,j}} = -\frac{c_{i+1}}{d_{i,j}}$, and seeing $\hat{\alpha}_{w_{i,j}} + \hat{\alpha}_{\mathsf{L}(w_{i,j})} \hat{\ell}_{w_{i,j}} = y_{i,j} c_1 \cdots c_{i+1} - \Big(1 d_{i,j} y_{i,j} + \sum_{\substack{\iota \in [1,m_i] \\ \text{s.t. } \iota \neq j}} \rho_{(i-1,j_L) \to (i,\iota)} d_{i,\iota} y_{i,\iota}\Big) c_1 \cdots c_i (\frac{c_{i+1}}{d_{i,j}})$, where we use Eq.(6),(10), and the fact that $w_{i,j}$ is an OR gate, hence $\rho_{(i-1,j_L) \to (i,j)} = 1$ from Eq.(7). The last equation cancels out $y_{i,j} c_1 \cdots c_{i+1}$ and we obtain Eq.(11).

If $w_{i,j}$ is an AND gate, $\mathcal{B}$ computes

$$\begin{aligned} \hat{K}_{w_{i,j}} &= -[\,\sigma_{(i-1,j_L),(i,j)}\,]^2_{[\ell+2,2\ell+i]} - [\,\sigma_{(i-1,j_R),(i,j)}\,]^2_{[\ell+2,2\ell+i]} \\ &= [\,\hat{\alpha}_{w_{i,j}} + \hat{\alpha}_{\mathsf{L}(w_{i,j})} \hat{\ell}_{w_{i,j}} + \hat{\alpha}_{\mathsf{R}(w_{i,j})} \hat{r}_{w_{i,j}}\,]^2_{[\ell+2,2\ell+i]}. \end{aligned}$$

This can be verified similarly as above, but this time, $w_{i,j}$ is an AND gate, hence $\rho_{(i-1,j_L) \to (i,j)} = 1 - a_{i,j}$ and $\rho_{(i-1,j_R) \to (i,j)} = a_{i,j}$. Nevertheless, both terms sum up to 1 and the same cancellation takes place. It remains to verify that $[\,\sigma_{(i-1,j_L),(i,j)}\,]^2_{[\ell+2,2\ell+i]}$ can be computed. We claim the following.

Claim 4. *For $i \in [1, \ell-1]$ and $j \in [1, m_i]$, $j' \in [1, m_{i+1}]$, $\mathcal{B}$ can efficiently compute*

$$[\,y_{i,j}\,]^2_{[\ell+2+i,2\ell+1]}, \qquad [\,\hat{\alpha}_{w_{i,j}}\,]^2_{[\ell+2,2\ell+1+i]}, \qquad [\,\sigma_{(i,j),(i+1,j')}\,]^2_{[\ell+2,2\ell+1+i]}.$$

Proof (of the claim). We prove by induction for i from $\ell-1$ to 1. Starting from gates of depth $\ell-1$, $\mathcal{B}$ can compute for any $j' \in [1, m_{\ell-1}]$,

$$\begin{aligned} [\,y_{\ell-1,j'}\,]^2_{\{2\ell+1\}} &= [\,\rho_{(\ell-1,j') \to (\ell,1)} d_{\ell,1}\,]^2_{\{2\ell+1\}}, \\ [\,\hat{\alpha}_{w_{\ell-1,j'}}\,]^2_{[\ell+2,3\ell]} &= [\,c_1 \cdots c_\ell \rho_{(\ell-1,j') \to (\ell,1)} d_{\ell,1}\,]^2_{[\ell+2,2\ell+1] \cup [2\ell+2,3\ell]}, \\ [\,\sigma_{(i-1,j'),(\ell,1)}\,]^2_{[\ell+2,3\ell]} &= [\,0\,]^2_{[\ell+2,2\ell+1] \cup [2\ell+2,3\ell]} \end{aligned}$$

where we note that the last equation is due to $m_\ell = 1$. $\mathcal{B}$ computes the elements for depth $i-1$ from those of depth i as follows.

$$\begin{aligned} [\,y_{i-1,j'}\,]^2_{[\ell+1+i,2\ell+1]} &= \sum_{\iota \in [1,m_i]} [\,\rho_{(i-1,j') \to (i,\iota)} d_{i,\iota}\,]^2_{\{\ell+1+i\}} \cdot [\,y_{i,\iota}\,]^2_{[\ell+2+i,2\ell+1]}, \\ [\,\hat{\alpha}_{w_{i-1,j'}}\,]^2_{[\ell+2,2\ell+i]} &= \sum_{\iota \in [1,m_i]} [\,c_1 \cdots c_i \rho_{(i-1,j') \to (i,\iota)} d_{i,\iota}\,]^2_{[\ell+2,\ell+1+i] \cup [2\ell+2,2\ell+i]} \cdot [\,y_{i,\iota}\,]^2_{[\ell+2+i,2\ell+1]}, \\ [\,\sigma_{(i-1,j'),(i,j)}\,]^2_{[\ell+2,2\ell+i]} &= \sum_{\substack{\iota \in [1,m_i] \\ \text{s.t. } \iota \neq j}} [\,c_1 \cdots c_{i+1} \rho_{(i-1,j') \to (i,\iota)} \frac{d_{i,\iota}}{d_{i,j}} y_{i,\iota}\,]^2_{[\ell+2,\ell+1+i] \cup [2\ell+2,2\ell+i]} \cdot [\,y_{i,\iota}\,]^2_{[\ell+2+i,2\ell+1]} \end{aligned}$$

All the encodings are available in D from the assumption. This concludes the proof of the claim. □

For further uses, for $j \in [1, n]$, $\mathcal{B}$ also computes

$$[\,\hat{h}_j\,]^2_{[\ell+2,2\ell+1]} = [\,c_1 d_{1,1}\,]^2_{\{\ell+2\}} \cdot [\,y_{1,j}\,]^2_{[\ell+3,2\ell+1]} + [\,h'_j\,]_\emptyset \cdot [\,1\,]^2_{[\ell+2,2\ell+1]} \tag{12}$$

(Re-randomizing Key). The simulated key is not perfectly distributed yet since their randomness are still correlated. Consider every variable $\hat{y}$ in $\{\hat{r}\} \cup \{\hat{\alpha}_{w_{i,j}}\}_{w_{i,j}\in\mathsf{Nodes}} \cup \{\hat{v}_{w_{1,j}}\}_{w_{1,j}\in\mathsf{Inputs}} \cup \{\hat{\ell}_{w_{i,j}}, \hat{r}_{w_{i,j}}\}_{w_{i,j}\in\mathsf{Gates}}$. We re-randomize them by implicitly setting new randomness as $\hat{y}'' = \hat{y} + y'$, where $\mathcal{B}$ samples $[\,y'\,]_\emptyset \leftarrow \mathsf{Samp}(\mathsf{param})$. To be able to compute corresponding keys with updated randomness, it amounts to verify that the encoding of the term that $\hat{y}$ is multiplied to in the key can be computed. For example, $\hat{r}$ appears in $\hat{D}_1 = [\,\beta_k + \hat{\phi}_2\hat{r}\,]^2_{[\ell+2,3\ell]}$. This can be re-randomized to $\hat{D}''_1 = [\,\beta_k + \hat{\phi}_2(\hat{r} + r')\,]^2_{[\ell+2,3\ell]}$ by computing $\hat{D}''_1 = \hat{D} + [\,r'\,]_\emptyset \cdot [\,\hat{\phi}_2\,]^2_{[\ell+2,3\ell]}$, which can be done since $[\,\hat{\phi}_2\,]^2_{[\ell+2,3\ell]}$ is computable (and we have $[\,vb\,]^2_{[\ell+2,3\ell]}$ from the assumption). The other terms can be verified as follows. $[\,\hat{\phi}_1\,]^2_{[\ell+2,3\ell]}$ is computable since we have $[\,v\,]^2_{[\ell+2,3\ell]}$. For $j \in [1, n]$, $[\,\hat{h}_j\,]^2_{[\ell+2,2\ell+1]}$ can be computed as in Eq.(12). Finally, for $w_{i,j} \in \mathsf{Gates}$, $[\,\hat{\alpha}_{\mathsf{L}(w_{i,j})}\,]^2_{[\ell+2,2\ell+i]}$ $[\,\hat{\alpha}_{\mathsf{R}(w_{i,j})}\,]^2_{[\ell+2,2\ell+i]}$ can be computed due to Claim 4, where we note that $\mathsf{L}(w_{i,j})$, $\mathsf{R}(w_{i,j})$ have depth $i-1$. $\mathcal{B}$ re-randomizes all terms and obtain a new semi-functional component $\widehat{\mathsf{SK}}''_k = \big(\{\hat{\mathcal{K}}''_w\}_{w\in\mathsf{Nodes}}, \hat{D}''_1, \hat{D}''_2, \hat{D}''_3\big)$.

(Returning Key). $\mathcal{B}$ adds the above semi-functional components to the normal components and returns $\mathsf{SK}_k + \widehat{\mathsf{SK}}''_k$.

Challenge. The adversary $\mathcal{A}$ outputs messages $M_0, M_1 \in \{0,1\}^\lambda$ along with a target string $x^\star$. $\mathcal{B}$ chooses $\mathfrak{b} \xleftarrow{\$} \{0,1\}$. Due to the game condition, it must be that $f(x^\star) = 0$. $\mathcal{B}$ first computes a normal ciphertext $(C_0, \mathcal{C}) \leftarrow \mathsf{Encrypt}(\mathsf{PK}, x^\star, M_\mathfrak{b})$, where we denote $\mathcal{C} = (C, \{C_j\}_{j\in A_x}, T_1, T_2)$. $\mathcal{B}$ also computes the message mask $K = [\,\alpha t\,]^1_{[1,3\ell]}$ in the process. $\mathcal{B}$ then produces semi-functional components as follows.

(Programming Randomness in Ciphertext). We first implicitly set $\hat{t} = -\frac{z}{b}$. Let

$$S = \left\{\, w \in \mathsf{Nodes} \;\middle|\; f_w(x^\star) = 0 \text{ and } f_{\mathsf{L}(w)}(x^\star) = 1 \,\right\}.$$

For $i \in [2, \ell]$, let $S_i = S \cap \{\, w \in \mathsf{Nodes} \mid \mathsf{Depth}(w) = i \,\}$. We then implicitly set

$$\hat{s} = z\left(1 + \sum_{w_{2,j}\in S_2} \frac{1}{a_{2,j}}\right) \cdots \left(1 + \sum_{w_{\ell,j}\in S_\ell} \frac{1}{a_{\ell,j}}\right). \tag{13}$$

(Simulating Ciphertext). From the above (implicit) definitions, the semi-functional component for ciphertext is well defined. $\mathcal{B}$ computes them as follows:

$$\hat{C} = [\,\frac{1}{\mu_1\cdots\mu_\ell} z\,]^2_{\{\ell+1\}} \cdot [\,\mu_1\,]^2_{\{1\}} \cdot \prod_{i\in[2,\ell]} \left([\,\mu_i\,]^2_{\{i\}} + \sum_{w_{i,j}\in S_i} [\,\mu_i \frac{1}{a_{i,j}}\,]^2_{\{i\}} \right) = [\,\hat{s}\,]^2_{[1,\ell+1]},$$
$$\hat{T}_1 = -[\,\frac{z}{b}\,]^2_{[1,\ell+1]} = [\,\hat{t}\,]^2_{[1,\ell+1]}.$$

Recall that $\mathcal{B}$ possesses $[\,\alpha\,]_\emptyset$ and hence can compute the semi-functional component of message mask as $\hat{K} = \hat{T}_1 \cdot [\,\alpha\,]^2_{[\ell+2,3\ell]} = [\,\alpha\hat{t}\,]^2_{[1,3\ell]}$. Next, $\mathcal{B}$ computes $\hat{T}_2 = [\,\hat{\phi}_2\hat{t} + \hat{\phi}_1\hat{s}\,]^2_{[1,\ell+1]}$ as follows. Substituting terms, we have $\hat{\phi}_2\hat{t} + \hat{\phi}_1\hat{s} = (vb + \phi'_2)(-\frac{z}{b}) + (v + \phi'_1)z\prod_{i\in[2,\ell]}\left(1 + \sum_{w_{i,k}\in S_i}\frac{1}{a_{i,k}}\right)$. A critical term zv is canceled out. Let $V = zv\prod_{i\in[2,\ell]}\left(1 + \sum_{w_{i,k}\in S_i}\frac{1}{a_{i,k}}\right) - zv$. As for intuition, $\mathcal{B}$

could compute the corresponding encoding of each monomial $\frac{zv}{a_{i_1,j_1}\cdots a_{i_\varepsilon,j_\varepsilon}}$ that appears in V and sum them up. However, the number of such terms in V could be of an exponential size in ℓ. We resolve this by instead writing $V = \sum_{i\in[2,\ell]}\left(\sum_{w_{i,k}\in S_i}\frac{zv}{a_{i,k}}\right)\prod_{j\in[i+1,\ell]}\left(1+\sum_{w_{j,k}\in S_j}\frac{1}{a_{j,k}}\right)$, where we purposely use zv in the sum that does not have the term 1, so that $\mathcal{B}$ can efficiently compute

$$[\,V\,]^2_{[1,\ell+1]} = \sum_{i\in[2,\ell]}\left(\prod_{j\in[2,i-1]}[\,\omega_j\,]^2_{\{j\}}\right)\cdot\left(\sum_{w_{i,k}\in S_i}[\,\frac{1}{\prod_{\iota\in[1,\ell]\setminus\{i\}}\omega_\iota}zv\frac{1}{a_{i,k}}\,]^2_{\{i,\ell+1\}}\right)$$
$$\cdot\left(\prod_{j\in[i+1,\ell]}\left([\,\omega_j\,]^2_{\{j\}}+\sum_{w_{j,k}\in S_j}[\,\omega_j\frac{1}{a_{j,k}}\,]^2_{\{j\}}\right)\right),$$

and $\hat{T}_2 = [\,\phi_2'\,]_\emptyset\cdot[\,\hat{t}\,]^2_{[1,\ell+1]} + [\,\phi_1'\,]_\emptyset\cdot[\,\hat{s}\,]^2_{[1,\ell+1]} + [\,V\,]^2_{[1,\ell+1]}$.

The trickiest part is to compute the remaining elements $\hat{C}_j = [\,\hat{s}\hat{h}_j\,]^2_{[1,\ell+1]}$ for all $j\in\mathcal{A}_{x^\star}$. From $\hat{h}_j = d_{1,1}y_{1,j}c_1 + h_j'$, we have

$$\hat{s}\hat{h}_j = \hat{s}h_j' + zc_1d_{1,1}\prod_{i\in[2,\ell]}\left(1+\sum_{w_{i,k}\in S_i}\frac{1}{a_{i,k}}\right)\cdot\sum_{(j_2,\ldots,j_{\ell-1})\in\mathsf{path}_{(1,j)}}\prod_{\iota\in[2,\ell]}\left(\rho_{(\iota-1,j_{\iota-1})\to(\iota,j_\iota)}d_{\iota,j_\iota}\right)$$
$$= \hat{s}h_j' + \sum_{(j_2,\ldots,j_{\ell-1})\in\mathsf{path}_{(1,j)}} zc_1d_{1,1}\prod_{i\in[2,\ell]}\underbrace{\left(1+\sum_{w_{i,k}\in S_i}\frac{1}{a_{i,k}}\right)\rho_{(i-1,j_{i-1})\to(i,j_i)}}_{\text{denote } P_{(i-1,j_{i-1})\to(i,j_i)}}d_{i,j_i}$$

where here we denote $j_1 = j$ and $j_\ell = 1$. For $i\in[2,\ell]$, we denote $P_{(i-1,j_{i-1})\to(i,j_i)}$ as above.

(Flawed Algorithm). For intuition, we first show a *flawed* algorithm for $\mathcal{B}$ to compute $[\,\hat{s}\hat{h}_j\,]^2_{[1,\ell+1]}$. Namely $\mathcal{B}$ would compute each product term in the sum. Each product term corresponds to one path from $w_{1,j}$ to $w_{\ell,1}$ and the sum consists of all such paths. Consider a path $(w_{1,j}, w_{2,j_2},\ldots, w_{\ell,1})$. Since $f_{w_{1,j}}(x^\star)=1$ and $f_{w_{\ell,1}}(x^\star)=f(x^\star)=0$, we have that there must be a node on the path, say w_{τ,j_τ}, with the largest depth such that $f_{w_{\tau,j_\tau}}(x^\star)=0$ but $f_{w_{\tau-1,j_{\tau-1}}}(x^\star)=1$. $\mathcal{B}$ would compute the corresponding encoding for this path as $[\,zc_1d_{1,1}\prod_{i\in[2,\ell]}P_{(i-1,j_{i-1})\to(i,j_i)}d_{i,j_i}\,]^2_{[1,\ell+1]} =$

$$[\,d_{1,1}\,]^2_{\{1\}}\cdot[\,zc_1P_{(\tau-1,j_{\tau-1})\to(\tau,j_\tau)}d_{\tau,j_\tau}\,]^2_{\{\tau,\ell+1\}}\cdot\prod_{\substack{i\in[2,\ell]\\ \text{s.t. } i\neq\tau}}[\,P_{(i-1,j_{i-1})\to(i,j_i)}d_{i,j_i}\,]^2_{\{i\}}.$$

This can be computed due to the two following claims.

Claim 5. *For $i\in[2,\ell]$, $j_{i-1}\in[1,m_{i-1}]$, $j_i\in[1,m_i]$, $\mathcal{B}$ can efficiently compute $[\,P_{(i-1,j_{i-1})\to(i,j_i)}d_{i,j_i}\,]^2_{\{i\}}$.*

Proof (of the claim). By definition, $P_{(i-1,j_{i-1})\to(i,j_i)} = \left(1+\sum_{w_{i,k}\in S_i}\frac{1}{a_{i,k}}\right)(b_1\cdot 1 + b_2\cdot a_{i,j_i})$, where $b_1,b_2\in\{0,1,-1\}$. By inspection, $P_{(i-1,j_{i-1})\to(i,j_i)}$ is a linear combination of $1, a_{i,j_i}, \frac{1}{a_{i,k}}, \frac{a_{i,j_i}}{a_{i,k}}$ for any $k\in[1,m_i]$, with coefficients in $\{0,1,-1\}$. We have the corresponding encodings in D. $\square$

Claim 6. *For any node w_{i,j_i} and $w_{i-1,j_{i-1}}\in\{\,\mathsf{L}(w_{i,j_i}),\mathsf{R}(w_{i,j_i})\}$ such that $f_{w_{i,j_i}}(x^\star)=0$ but $f_{w_{i-1,j_{i-1}}}(x^\star)=1$, $\mathcal{B}$ can efficiently compute $[\,zc_1P_{(i-1,j_{i-1})\to(i,j_i)}d_{i,j_i}\,]^2_{\{i,\ell+1\}}$.*

Proof (of the claim). Due to the definition of w_{i,j_i}, it must be that $\mathsf{GateType}(w_{i,j_i})=\mathsf{AND}$. There are two cases.

- Suppose $w_{i-1,j_{i-1}} = \mathsf{L}(w_{i,j_i})$. Hence, we have $w_{i,j_i} \in S_i$ and $\rho_{(i-1,j_{i-1})\to(i,j_i)} = 1 - a_{i,j_i}$. Therefore,

  $$P_{(i-1,j_{i-1})\to(i,j_i)} = \left(1 + \sum_{w_{i,k}\in S_i} \frac{1}{a_{i,k}}\right)(1 - a_{i,j_i}).$$

  We observe that the critical term, 1, is canceled out via $(1+\frac{1}{a_{i,j_i}})(1-a_{i,j_i})$. Indeed, $P_{(i-1,j_{i-1})\to(i,j_i)}$ is a linear combination of $a_{i,j_i}, \frac{1}{a_{i,j_i}}, \frac{1}{a_{i,k}}, \frac{a_{i,j_i}}{a_{i,k}}$, for $k \neq j_i$, with coefficients in $\{0, 1, -1\}$.

- Suppose $w_{i-1,j_{i-1}} = \mathsf{R}(w_{i,j_i})$. Hence, we have $w_{i,j_i} \notin S_i$ and $\rho_{(i-1,j_{i-1})\to(i,j_i)} = a_{i,j_i}$. Therefore,

  $$P_{(i-1,j_{i-1})\to(i,j_i)} = \left(1 + \sum_{w_{i,k}\in S_i} \frac{1}{a_{i,k}}\right)(a_{i,j_i}).$$

  We observe that the critical term, 1, does not appear since $w_{i,j_i} \notin S_i$. Indeed, $P_{(i-1,j_{i-1})\to(i,j_i)}$ is a linear combination of $a_{i,j_i}, \frac{a_{i,j_i}}{a_{i,k}}$, for $k \neq j_i$, with coefficients in $\{0, 1\}$.

All the encodings are available in D from the assumption. This concludes the proof of the claim. $\square$

The above algorithm is indeed flawed since although the term for each path can be efficiently computed as above, the number of all paths in the sum can be of an exponential size in ℓ.

(Corrected Algorithm). To resolve this issue, we present an efficient algorithm for $\mathcal{B}$ that outputs $[\,\hat{s}\hat{h}_j\,]^2_{[1,\ell+1]}$ and runs in polynomital time in the number of all nodes in the circuit. For $i \in [1, \ell-1], j_i \in [1, m_i]$, define $\mathsf{path}^{(0)}_{(i,j_i)} \subseteq \mathsf{path}_{(i,j_i)}$ as the set of all paths in $\mathsf{path}_{(i,j_i)}$ of which *all gates* on the path evaluates $x^\star$ to 0. That is, $(j_i, j_{i+1}, \dots, j_{\ell-1}, 1) \in \mathsf{path}^{(0)}_{(i,j_i)}$ iff for all $k \in [i, \ell]$ it holds that $f_{w_{k,j_k}}(x^\star) = 0$. Let $\mathsf{path}^{(1)}_{(i,j_i)} := \mathsf{path}_{(i,j_i)} \smallsetminus \mathsf{path}^{(0)}_{(i,j_i)}$. We define

$$\begin{aligned} Y_{i,j_i} &:= \sum_{(j_i,j_{i+1},\dots,j_{\ell-1},1)\in\mathsf{path}^{(0)}_{(i,j_i)}} \prod_{k\in[i+1,\ell]} P_{(k-1,j_{k-1})\to(k,j_k)} d_{k,j_k}, \\ W_{i,j_i} &:= zc_1 \cdot \sum_{(j_i,j_{i+1},\dots,j_{\ell-1},1)\in\mathsf{path}^{(1)}_{(i,j_i)}} \prod_{k\in[i+1,\ell]} P_{(k-1,j_{k-1})\to(k,j_k)} d_{k,j_k}. \end{aligned}$$

This implies that for $j \in A_{x^\star}$, we have $\hat{s}\hat{h}_j = \hat{s}h'_j + W_{1,j}d_{1,1}$. This is since for $j \in A_{x^\star}$ we have $f_{w_{1,j}}(x^\star) = 1$, hence $\mathsf{path}^{(1)}_{1,j} = \mathsf{path}_{1,j}$.

Claim 7. *For $i \in [1, \ell-1]$, $j_i \in [1, m_i]$, $\mathcal{B}$ can efficiently compute $[\,Y_{i,j_i}\,]^2_{[i+1,\ell]}$ and $[\,W_{i,j_i}\,]^2_{[i+1,\ell+1]}$.*

Proof. We prove by induction for i from $\ell-1$ to 1. We first prove for the base case. Consider node $w_{\ell-1,j_{\ell-1}}$ for any $j_{\ell-1} \in [1, m_{\ell-1}]$. Note that $\mathsf{path}_{w_{\ell-1,j_{\ell-1}}} = \{(j_{\ell-1}, 1)\}$. There are two cases:

- Case $f_{w_{\ell-1,j_{\ell-1}}}(x^\star) = 0$. We have $\mathsf{path}^{(0)}_{w_{\ell-1,j_{\ell-1}}} = \{(j_{\ell-1}, 1)\}$, $\mathsf{path}^{(1)}_{w_{\ell-1,j_{\ell-1}}} = \emptyset$. Therefore, $[\,W_{\ell-1,j_{\ell-1}}\,]^2_{[\ell,\ell+1]} = [\,0\,]^2_{[\ell,\ell+1]}$. $\mathcal{B}$ computes $[\,Y_{\ell-1,j_{\ell-1}}\,]^2_{\{\ell\}} = [\,P_{(\ell-1,j_{\ell-1})\to(\ell,1)} d_{\ell,1}\,]^2_{\{\ell\}}$ as in Claim 5.

- Case $f_{w_{\ell-1,j_{\ell-1}}}(x^\star) = 1$. We have $\mathsf{path}^{(0)}_{w_{\ell-1,j_{\ell-1}}} = \emptyset$, $\mathsf{path}^{(1)}_{w_{\ell-1,j_{\ell-1}}} = \{(j_{\ell-1}, 1)\}$. Thus, $[\,Y_{\ell-1,j_{\ell-1}}\,]^2_{\{\ell\}} = [\,0\,]^2_{\{\ell\}}$. $\mathcal{B}$ computes $[\,W_{\ell-1,j_{\ell-1}}\,]^2_{[\ell,\ell+1]} = [\,zc_1 P_{(\ell-1,j_{\ell-1})\to(\ell,1)} d_{\ell,1}\,]^2_{[\ell,\ell+1]}$ as in Claim 6.

Next, we show how to compute $[\,Y_{i-1,j_{i-1}}\,]^2_{[i,\ell]}$ and $[\,W_{i-1,j_{i-1}}\,]^2_{[i,\ell+1]}$ for $j_{i-1} \in [1, m_{i-1}]$ from $[\,Y_{i,j_i}\,]^2_{[i+1,\ell]}$ and $[\,W_{i,j_i}\,]^2_{[i+1,\ell+1]}$ for $j_i \in [1, m_i]$. There are two cases:

- Case $f_{w_{i-1,j_{i-1}}}(x^\star) = 0$. Thus, $\mathsf{path}^{(\iota)}_{w_{i-1,j_{i-1}}} = \bigcup_{j_i} \left\{ (j_{i-1}, j_i, \ldots, j_\ell) \,\middle|\, (j_i, \ldots, j_\ell) \in \mathsf{path}^{(\iota)}_{w_{i,j_i}} \right\}$, for both $\iota \in \{0,1\}$, where the union range is $j_i \in [1, m_i]$ such that $w_{i-1,j_{i-1}} \in \{\,\mathsf{L}(w_{i,j_i}),\, \mathsf{R}(w_{i,j_i})\}$. Therefore, we have that

$$[\,Y_{i-1,j_{i-1}}\,]^2_{[i,\ell]} = \sum_{j_i \in [1,m_i]} [\,P_{(i-1,j_{i-1}) \to (i,j_i)} d_{i,j_i}\,]^2_{\{i\}} \cdot [\,Y_{i,j_i}\,]^2_{[i+1,\ell]},$$
$$[\,W_{i-1,j_{i-1}}\,]^2_{[i,\ell]} = \sum_{j_i \in [1,m_i]} [\,P_{(i-1,2_{i-1}) \to (i,j_i)} d_{i,j_i}\,]^2_{\{i\}} \cdot [\,W_{i,j_i}\,]^2_{[i+1,\ell]}.$$

  Due to Claim 5, $\mathcal{B}$ can compute $[\,P_{(i-1,j_{i-1}) \to (i,j_i)} d_{i,j_i}\,]^2_{\{i\}}$. Note also that for j_i such that $w_{i-1,j_{i-1}} \notin \{\,\mathsf{L}(w_{i,j_i}),\, \mathsf{R}(w_{i,j_i})\}$, $P_{(i-1,j_{i-1}) \to (i,j_i)} = 0$ by definition Eq.(7).

- Case $f_{w_{i-1,j_{i-1}}}(x^\star) = 1$. Thus $\mathsf{path}^{(1)}_{w_{i-1,j_{i-1}}} = \bigcup_{j_i} \left\{ (j_{i-1}, j_i, \ldots, j_\ell) \,\middle|\, (j_i, \ldots, j_\ell) \in \mathsf{path}_{w_{i,j_i}} \right\}$, where the union range is $j_i \in [1, m_i]$ such that $w_{i-1,j_{i-1}} \in \{\,\mathsf{L}(w_{i,j_i}),\, \mathsf{R}(w_{i,j_i})\}$. On the other hand, we have $\mathsf{path}^{(0)}_{w_{i-1,j_{i-1}}} = \emptyset$. Therefore, $[\,Y_{i-1,j_{i-1}}\,]^2_{[i,\ell]} = [\,0\,]^2_{[i,\ell]}$ and

$$[\,W_{i-1,j_{i-1}}\,]^2_{[i,\ell]} = \sum_{j_i \in [1,m_i]} \Big([\,zc_1 P_{(i-1,j_{i-1}) \to (i,j_i)} d_{i,j_i}\,]^2_{\{i,\ell+1\}} \cdot [\,Y_{i,j_i}\,]^2_{[i+1,\ell]}$$
$$+ [\,P_{(i-1,j_{i-1}) \to (i,j_i)} d_{i,j_i}\,]^2_{\{i\}} \cdot [\,W_{i,j_i}\,]^2_{[i+1,\ell]} \Big).$$

  $\mathcal{B}$ can compute $[\,zc_1 P_{(i-1,j_{i-1}) \to (i,j_i)} d_{i,j_i}\,]^2_{\{i,\ell+1\}}$ due to Claim 6.

This concludes the proof of the claim. □

Finally, $\mathcal{B}$ computes for $j \in A_{x^\star}$, $[\,\hat{s}\hat{h}_j\,]^2_{[1,\ell+1]} = [\,\hat{s}\,]^2_{[1,\ell+1]} \cdot [\,h'_j\,]_\emptyset + [\,d_{1,1}\,]^2_{\{1\}} \cdot [\,W_{1,j}\,]^2_{[2,\ell+1]}$.

(Re-randomizing Ciphertext). The simulated ciphertext is not perfectly distributed yet since the randomness $\hat{s}, \hat{t}$ are still correlated. We will re-randomize $\hat{s}$, so that it is independent from $\hat{t}$. This can be done by sampling $[\,s'\,]_\emptyset \leftarrow \mathsf{Samp}(\mathsf{param})$ and computing a new semi-functional component $\hat{\mathcal{C}}'' = (\hat{C}'', \{\hat{C}''_j\}_{j \in A_x}, \hat{T}''_1, \hat{T}''_2)$ as $\hat{C}''_0 = \hat{C}_0$, $\hat{T}''_1 = \hat{T}_1$, and

$$\hat{C}'' = \hat{C} + [\,s'\,]_\emptyset \cdot [\,1\,]^2_{[1,\ell+1]}, \quad \forall_{j \in A_{x^\star}}\ \hat{C}''_j = \hat{C}_j + [\,s'\,]_\emptyset \cdot [\,\hat{h}_j\,]^2_{[1,\ell+1]}, \quad \hat{T}''_2 = \hat{T}_2 + [\,s'\,]_\emptyset \cdot [\,\hat{\phi}_1\,]^2_{[1,\ell+1]},$$

Note that the semi-functional message mask is unchanged $\hat{K}'' = \hat{K}$. Note that $[\,\hat{\phi}_1\,]^2_{[1,\ell+1]} = [\,\phi'_1\,]_\emptyset \cdot [\,1\,]^2_{[1,\ell+1]} + [\,v\,]^2_{[1,\ell+1]}$ is computable from D. For $j \in [1, n]$, $\mathcal{B}$ computes $[\,\hat{h}_j\,]^2_{[1,\ell+1]}$ in exactly the same recursive manner as computing $[\,\hat{h}_j\,]^2_{[\ell+2,2\ell+1]}$ in Eq.(12), except that this time $\mathcal{B}$ uses a different set of elements of D. To do so, we first claim:

Claim 8. *For $i \in [1, \ell-1]$ and $j \in [1, m_i]$, $\mathcal{B}$ can efficiently compute $[\,\nu_{i+1} \cdots \nu_\ell \cdot y_{i,j}\,]^2_{[i+1,\ell]}$, where ν_i is defined in D from the assumption.*

Proof (of the claim). We prove by induction for i from ℓ to 2. For the base case, $\mathcal{B}$ can compute for any $j' \in [1, m_{\ell-1}]$, $[\,\nu_\ell y_{\ell-1,j'}\,]^2_{\{\ell\}} = [\,\nu_\ell \rho_{(\ell-1,j') \to (\ell,1)} d_{\ell,1}\,]^2_{\{\ell\}}$. (This is available from D). Next the inductive step is $[\,\nu_i \cdots \nu_\ell \cdot y_{i-1,j'}\,]^2_{[i,\ell]} = \sum_{\iota \in [1,m_i]} [\,\nu_i \rho_{(i-1,j') \to (i,\iota)} d_{i,\iota}\,]^2_{\{i\}} \cdot [\,\nu_{i+1} \cdots \nu_\ell \cdot y_{i,\iota}\,]^2_{[i+1,\ell]}$. Note that $[\,\nu_i \rho_{(i-1,j') \to (i,\iota)} d_{i,\iota}\,]^2_{\{i\}}$ is available from D. This concludes the proof of the claim. □

$\mathcal{B}$ then compute $[\hat{h}_j]^2_{[1,\ell+1]} = [\frac{1}{\nu_1 \cdots \nu_\ell} c_1]^2_{\{\ell+1\}} \cdot [\nu_1 d_{1,1}]^2_{\{1\}} \cdot [\nu_2 \cdots \nu_\ell \cdot y_{1,j}]^2_{[2,\ell]} + [h'_j]_\emptyset \cdot [1]^2_{[1,\ell+1]}$.

(Returning Ciphertext). $\mathcal{B}$ computes C_0 from the message mask: $C_0 = \mathsf{Ext}(\mathsf{param}, K + \hat{K}'') \oplus M_{\mathfrak{b}}$. It returns a semi-functional ciphertext as $(C_0, \mathcal{C} + \hat{\mathcal{C}}'')$.

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

Guess. The algorithm $\mathcal{B}$ has properly simulated $\mathsf{G}_{k,1}$ if $\delta = 0$, and $\mathsf{G}_{k,2}$ if $\delta \in_R \mathcal{R}$. Hence, $\mathcal{B}$ can use the output of $\mathcal{A}$ to break the $\mathsf{EMDDH1}$ Assumption. This concludes the proof of Lemma 3. $\square$

8 A Fully Secure Ciphertext-Policy ABE Scheme for Circuits

In this section, we describe our CP-ABE for circuits. The scheme is dual to our KP-ABE: the key and ciphertext elements are swapped. We use the same variable notations from KP-ABE for unmodified terms. The difference is at the head elements, and both key and ciphertext will have one element more than KP-ABE. We note that the definition for CP-ABE is analogous to KP-ABE.

- **Setup**$(1^\lambda, n, \ell) \to (\mathsf{PK}, \mathsf{MSK})$. Set $\kappa = 3\ell$ and $\nu = 3$. Run $\mathsf{InstGen}(1^\lambda, \kappa, \nu) \to (\mathsf{param}, \mathsf{esk})$. Sample $\alpha, h_1, \ldots, h_n, \phi_1, \psi_2, \psi_3 \xleftarrow{\$} \mathcal{R}$. Output

$$\mathsf{PK} = \Big(\mathsf{param}, \{[1]^1_S, [1]^3_S\}_{S \in \mathcal{S} = \{[1,\ell+1],[\ell+2,2\ell+1],\{2\ell+2\},\ldots,\{3\ell\}\}},$$
$$[\alpha]^1_{[1,3\ell]}, [h_1]^1_{[\ell+2,2\ell+1]} \ldots, [h_n]^1_{[\ell+2,2\ell+1]}, [\phi_1]^1_{[\ell+2,3\ell]}, [\psi_2]^1_{[\ell+2,3\ell]}, [\psi_3]^1_{[\ell+2,3\ell]}\Big).$$
$$\mathsf{MSK} = \Big(\mathsf{param}, [1]^1_{[1,\ell+1]}, [\alpha]^{1,2}_{[1,\ell+1]}, [h_1]^1_{[1,\ell+1]} \ldots, [h_n]^1_{[1,\ell+1]}, [\phi_1]^1_{[1,\ell+1]}, [\psi_2]^1_{[1,\ell+1]}, [\psi_3]^1_{[1,\ell+1]}\Big).$$

- **Encrypt**$(\mathsf{PK}, f \in \mathbb{F}_{n,\ell}) \to \mathsf{CT}$. Sample $[r]_\emptyset, [u]_\emptyset$ and $[\alpha_w]_\emptyset$ for all $w \in \mathsf{Nodes}$ from $\mathsf{Samp}(\mathsf{param})$, and set

$$W_0 = [u]^1_{[\ell+2,3\ell]}, \quad W_1 = [\psi_3 u + \psi_2 r]^1_{[\ell+2,3\ell]}, \quad D_2 = [r]^1_{[\ell+2,3\ell]}, \quad D_3 = [\phi_1 r - \alpha_{w_{\mathsf{top}}}]^1_{[\ell+2,3\ell]}.$$

Compute the element $\mathcal{K}_w$ for each $w \in \mathsf{Nodes}$ in exactly the same way as in the KP-ABE scheme except that we do not add masks from the $\mathbb{Z}_{N_3}$ subring. The message is masked in $C_0 = \mathsf{Ext}(\mathsf{param}, [\alpha u]^1_{[1,3\ell]}) \oplus M$. Output the ciphertext as $\mathsf{CT} = (C_0, \{\mathcal{K}_w\}_{w \in \mathsf{Nodes}}, W_0, W_1, D_2, D_3)$.

- **KeyGen**$(\mathsf{MSK}, x \in \{0,1\}^n, M \in \{0,1\}^\lambda) \to \mathsf{SK}$. Let $A_x = \{ j \in [1,n] \mid x_j = 1 \}$. Sample $[\tau]_\emptyset, [s]_\emptyset \leftarrow \mathsf{Samp}(\mathsf{param})$. Compute $C' = [s]^1_{[1,\ell+1]}, C'_j = [h_j s]^1_{[1,\ell+1]}$, and

$$Q'_1 = [\tau]^1_{[1,\ell+1]}, \qquad Q'_2 = [\psi_2 \tau + \phi_1 s]^1_{[1,\ell+1]}, \qquad Q'_3 = [\alpha]^{1,2}_{[1,\ell+1]} + [\psi_3 t]^1_{[1,\ell+1]}.$$

The algorithm then adds a random mask from the $\mathbb{Z}_{N_3}$ subring to each element as usual. It outputs the key $\mathsf{SK} = (C, \{C_j\}_{j \in A_x}, Q_1, Q_2, Q_3)$.

- **Decrypt**$(\mathsf{SK}, \mathsf{CT}) \to M$. Assume that $f(x) = 1$. We compute exactly as in KP-ABE until we obtain $E_{w_{\mathsf{top}}} = [\alpha_{w_{\mathsf{top}}} s]^1_{[1,3\ell]}$ at the top gate. From this, it computes

$$\begin{aligned}
&- E_{w_{\mathsf{top}}} - C D_3 - Q_1 W_1 + Q_2 D_2 + Q_3 W_0 \\
&= -[\alpha_{w_{\mathsf{top}}} s]^1_{[1,3\ell]} - [s]^1_{[1,\ell+1]} \cdot [\phi_1 r - \alpha_{w_{\mathsf{top}}}]^1_{[\ell+2,3\ell]} - [\tau]^1_{[1,\ell+1]} \cdot [\psi_3 u + \psi_2 r]^1_{[\ell+2,3\ell]} \\
&\quad + [\psi_2 \tau + \phi_1 s]^1_{[1,\ell+1]} \cdot [r]^1_{[\ell+2,3\ell]} + ([\alpha]^{1,2}_{[1,\ell+1]} + [\psi_3 \tau]^1_{[1,\ell+1]}) \cdot [u]^1_{[\ell+2,3\ell]} \\
&= [\alpha u]^1_{[1,3\ell]},
\end{aligned}$$

and computes the message mask $\mathsf{Ext}(\mathsf{param}, [\alpha u]^1_{[1,3\ell]})$ and obtains M from C_0.

Properties of Our CP-ABE for Circuits. Our CP-ABE system is succinct, in an analogous manner to our KP-ABE. The size of a ciphertext for circuit f is proportional to the size of circuit f (the number of nodes), while the size of a key for string x is proportional to the number of 1's in x. Also similarly to our KP-ABE, our CP-ABE system does not require any bound on circuit size and fan-out, *i.e.,* we can setup a fixed system, and ciphertexts for circuits of any size and fan-out can be constructed (as long as they are polynomial sizes, since the ciphertext size is linear to the circuit size). We only require bounds on input length n and depth ℓ. We remark as in our KP-ABE case that, however, the assumption for CP-ABE will be parameterized by the maximum number of gates per layer (and hence the circuit size and the maximum fan-out) of the circuit for which the adversary issues the challenge query. We emphasize that this number is not bounded at the system setup.

Comparison to the CP-ABE of GGHSW. We compare our CP-ABE to the (selectively secure) *CP*-ABE system of GGHSW [9] for the bounded/unbounded property and succinctness. The GGHSW system requires a bound for circuit sizes (for ciphertexts) when the system is setup. This is essential, due to their use of universal circuits. Next, we consider succinctness of their scheme. In their scheme, a string x (for key) will be converted to a (variant of) universal circuit, U_x, while a circuit f (for ciphertext) will be converted to a string y_f (to be input to U_x). The efficiency of conversion to U_x and y_f will depend on the implementation scheme of the universal circuit. Hence, even though the resulting scheme has ciphertext size proportional to the number of 1 in y_f and key size proportional to the size of U_x (as in their KP-ABE), whether their scheme is succinct will depend on the conversion efficiency. Therefore, we cannot conclude that their scheme is succinct or not. On the other hand, our CP-ABE scheme will always be succinct.

8.1 Assumptions and Security Theorem for CP-ABE

The security proof for CP-ABE works in a dual manner to that of KP-ABE. Essentially, the proof of co-selective security based transition in KP-ABE will be used as that of selective security based transition in CP-ABE (and vice-versa in the dual way). Hence, $\mathsf{EMDDH1}, \mathsf{EMDDH2}$ will be used in phase 2 and 1 respectively. We indeed use slight variants of the assumptions as follows.

Definition 7 ((ℓ, m)-$\mathsf{EMDDH1\text{-}dual}$)**.** Let $\mathsf{InstGen}(1^\lambda, 3\ell, 3) \rightarrow (\mathsf{param}, \mathsf{esk})$. 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 $\mathcal{R}$. The Dual (ℓ, m)-Expanded Multi-linear Decisional Diffie-Hellman Assumption 1 states that the following distributions are computationally indistinguishable:

$$\left(\bar{D}, Z = [\, bz \,]^2_{[1,\ell+1]}\right) \qquad \text{and} \qquad \left(\bar{D}, Z = [\, \zeta \,]^2_{[1,\ell+1]}\right),$$

where $\bar{D} := D \cup \{[\, \frac{c_1 \cdots c_{\ell+1}}{b} \,]^2_{[\ell+2,3\ell]}, [\, b \,]^2_{[1,\ell+1]}\} \smallsetminus \{[\, \frac{z}{b} \,]^2_{[1,\ell+1]}, [\, vb \,]^2_{[\ell+2,3\ell]}\}$, where D is defined in the definition of the $\mathsf{EMDDH1}$ assumption (Definition 5).

We prove the generic hardness of the $\mathsf{EMDDH1\text{-}dual}$ Assumption in Lemma 11 in §A.

Definition 8 (ℓ-$\mathsf{EMDDH2\text{-}dual}$)**.** Let $\mathsf{InstGen}(1^\lambda, 3\ell, 3) \rightarrow (\mathsf{param}, \mathsf{esk})$. Sample $z, c_1, \cdots, c_{\ell+1}$, and ζ from $\mathcal{R}$. The Dual ℓ-Expanded Multi-linear Decisional Diffie-Hellman Assumption 2 states that the following distributions are computationally indistinguishable:

$$\left(D, Z = [\, c_1 \cdots c_{\ell+1} z \,]^2_{[1,\ell+1]}\right) \qquad \text{and} \qquad \left(D, Z = [\, \zeta \,]^2_{[1,\ell+1]}\right),$$

where D consists of: param, $\left\{[\, 1 \,]^1_{\{i\}}, [\, 1 \,]^2_{\{i\}}, [\, 1 \,]^3_{\{i\}}\right\}_{i\in[1,3\ell]}$ and $[\, z \,]^2_{[1,\ell+1]}, [\, c_1 \cdots c_{\ell+1} \,]^2_{[1,\ell+1]}$,

$$[\, c_1 \,]^2_{[1,\ell+1]}, [\, c_1 \,]^2_{[\ell+2,2\ell+1]}, [\, c_1 \,]^2_{\{2\ell+2\}}, \ldots, [\, c_1 \,]^2_{\{3\ell\}}, \quad [\, c_2 \,]^2_{[\ell+2,2\ell+1]}, [\, c_3 \,]^2_{\{2\ell+2\}}, \ldots, [\, c_{\ell+1} \,]^2_{\{3\ell\}}.$$

The $\mathsf{EMDDH2\text{-}dual}$ assumption extends the regular Multi-linear DDH (in asymmetric settings) by also giving out one more element $[\, c_1 \cdots c_{\ell+1} \,]^2_{[1,\ell+1]}$. This is also the only difference from $\mathsf{EMDDH2}$. We can see that this would not help attacking since it cannot be multiplied with available $[\, z \,]^2_{[1,\ell+1]}$.

We can now state the security theorem for our CP-ABE for circuits.

Theorem 9. *Suppose that the* $\mathsf{SD1}, \mathsf{SD2}, \mathsf{EMDDH1\text{-}dual}, \mathsf{EMDDH2\text{-}dual}$ *Assumptions hold. Then our CP-ABE for circuits is fully secure. More precisely, for any PPT adversary* $\mathcal{A}$ *that attacks our CP-ABE for circuits in* $\mathbb{F}_{n,\ell}$*, there exist PPT algorithms* $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3, \mathcal{B}_4$*, whose running times are that of* $\mathcal{A}$ *plus some polynomial times, such that for any* λ*,*

$$\mathsf{Adv}^{(n,\ell)\text{-}\mathsf{CPABE}}_{\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{SD1}}_{\mathcal{B}_1}(\lambda) + (2q_1+2)\mathsf{Adv}^{\mathsf{SD2}}_{\mathcal{B}_2}(\lambda) + q_1 \mathsf{Adv}^{\ell\text{-}\mathsf{EMDDH2\text{-}dual}}_{\mathcal{B}_3}(\lambda) + \mathsf{Adv}^{(\ell,m)\text{-}\mathsf{EMDDH1\text{-}dual}}_{\mathcal{B}_4}(\lambda),$$

where q_1 *is the number of key queries by* $\mathcal{A}$ *in phase 1, and* m *is the maximum number of (internal) gates per layer of the circuit* $f^\star$ *for which* $\mathcal{A}$ *issues the challenge query.*

The security proof for CP-ABE has exactly the same structure as that of KP-ABE. The semi-functional elements can be defined analogously. The proofs for transitions that are based on Subgroup Decision Assumptions can be done in almost the same manner as those of KP-ABE. Therefore, we present only the proofs for the transitions that modify semi-functional keys from type-1 to type-2 for both phases. Nevertheless, they are also quite similar to the proofs for KP-ABE albeit in a dual manner. We present the two Lemmata and their proofs in §C.

Acknowledgement. I would like to thank Takahiro Matsuda, Shota Yamada, and Goichiro Hanaoka for their helpful comments on this paper. I would like to also thank Mehdi Tibouchi for answering my questions about the CLT scheme.

References

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

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

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

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

[5] 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.

[6] D. Boneh, A. Silverberg. Applications of multilinear forms to cryptography. Contemporary Mathematics Vol. 324, pp. 71–90, 2003.

[7] J. Coron, T. Lepoint, M. Tibouchi. Practical Multilinear Maps over the Integers. In *Crypto 2013*, pp. 476-493, 2013.

[8] S. Garg, C. Gentry, S. Halevi. Candidate multilinear maps from ideal lattices In *Eurocrypt 2013*, pp. 1–17, 2013.

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

[10] 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.

[11] S. Garg, C. Gentry, S. Halevi, M. Zhandry. Fully Secure Attribute Based Encryption from Multilinear Maps. Cryptology ePrint Archive: Report 2014/622. (Retrieved version posted 13-Aug-2014).

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

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

[14] C. Gentry, A. Lewko, A. Sahai, B. Waters. Indistinguishability Obfuscation from the Multilinear Subgroup Elimination Assumption. Cryptology ePrint Archive: Report 2014/309.

[15] S. Goldwasser, Y. Kalai, R.A. Popa, V. Vaikuntanathan, N. Zeldovich. Reusable garbled circuits and succinct functional encryption. In *STOC 2013*, pp. 555–564, 2013.

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

[17] S. Gorbunov, V. Vaikuntanathan, H. Wee. Attribute-based encryption for circuits. In *STOC 2013*, pp. 545-554, 2013.

[18] 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.

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

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

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

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

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

[24] Y. Rouselakis, B. Waters New Constructions and Proof Methods for Large Universe Attribute-Based Encryption Cryptology ePrint Archive: Report 2012/583.

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

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

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

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

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

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

[31] B. Waters. A Punctured Programming Approach to Adaptively Secure Functional Encryption. Cryptology ePrint Archive: Report 2014/588. (Retrieved version posted on 30-Jul-2014).

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

A Generic Hardness of Our Assumptions

Lemma 10. *The* EMDDH1 *Assumption holds in the generic graded encoding model.*

Proof. Similar to [30], it suffices to prove that there is no generic attack, and that the number of elements in D and the maximum degree of elements (which are monomials) in D are of polynomial size. The elements with the most flexible range vary with $i \in [1,\ell], j \in [1,m], j' \in [1,m], (e,e') \in \mathsf{E}$. Hence, the number of elements is $O(\ell m^2)$. More specifically, it is about $11(\ell m^2 + \ell m)$. To inspect the maximum degree monomial, we must first normalize all monomials so that they have no denominator. We normalize all $\mu_i, \nu_i, \omega_i, a_{i,j}, d_{i,j}$ which takes $2\ell m + 3\ell$ degrees. The maximum degree in numerator is $\ell + 3$, hence the maximum degree of normalized monomials is $2\ell m + 4\ell + 3$.

Next, we prove that there is no $[\,X_1\,]^2_{S_1}, \ldots, [\,X_k\,]^2_{S_k}$, $[\,Y_1\,]^2_{T_1}, \ldots, [\,Y_t\,]^2_{T_t}$ in D such that

$$[\,X_1\,]^2_{S_1} \cdots [\,X_k\,]^2_{S_k} \cdot [\,c_1 \cdots c_{\ell+1} b\,]^2_{[\ell+2,3\ell]} = [\,Y_1\,]^2_{T_1} \cdots [\,Y_t\,]^2_{T_t}$$

where $\bigcup_{j\in[1,k]} S_j = [1,\ell+1]$ and $\bigcup_{j\in[1,t]} T_j = [1,3\ell]$ (such that for $j \neq j'$, $S_j \cap S_{j'} = \emptyset$, $T_j \cap T_{j'} = \emptyset$). Intuitively, the equation models the zero-test procedure. We observe that since b appears only in $[\,\frac{z}{b}\,]^2_{[1,\ell+1]}$ and $[\,vb\,]^2_{[\ell+2,3\ell]}$ in D, to make the equation hold, it must be that either $[\,X_1\,]^2_{S_1} = [\,\frac{z}{b}\,]^2_{[1,\ell+1]}$ or $[\,Y_1\,]^2_{T_1} = [\,vb\,]^2_{[\ell+2,3\ell]}$ (wlog for the index of X, Y). We first assume that $[\,Y_1\,]^2_{T_1} = [\,vb\,]^2_{[\ell+2,3\ell]}$. This implies that wlog X_1 must contain v or Y_2 must contain v^{-1} so as to make it hold. In the latter case, no such Y_2 encoded in level $\subseteq [1,\ell+1]$ exists in D. In the former case, there are only two such elements for X_1, namely, $[\,v\,]^2_{[1,\ell+1]}$ and $[\,\frac{1}{\omega_1\cdots\omega_{i-1}\omega_{i+1}\cdots\omega_\ell} zv \frac{1}{a_{i,j}}\,]^2_{\{i,\ell+1\}}$. To make the equation hold, there must be a combination of Y's that contains $c_1 \cdots c_{\ell+1}$ encoded in $[1,\ell+1]$, but we do not have such elements. Hence, by contradiction, the assumption is wrong and it must be the other case: $[\,X_1\,]^2_{S_1} = [\,\frac{z}{b}\,]^2_{[1,\ell+1]}$. In this case, the left-hand side of the equation becomes $[\,\frac{z}{b}\,]^2_{[1,\ell+1]} \cdot [\,c_1 \cdots c_{\ell+1} b\,]^2_{[\ell+2,3\ell]} = [\,zc_1 \cdots c_{\ell+1}\,]^2_{[1,3\ell]}$.

Now we must pick a combination of Y_j's that equals to $[\,zc_1 \cdots c_{\ell+1}\,]^2_{[1,3\ell]}$. In particular, we observe that the combination must cover the whole level $[1,3\ell]$, and that variables that appear in Y_j but do not appear in $zc_1 \cdots c_{\ell+1}$ needed to be canceled out in the combination. Since z appears in the left-hand-side, we must pick an element containing z; there are only three cases:

- $[\,Y_1\,]^2_{T_1} = [\,\frac{1}{\omega_1\cdots\omega_{i-1}\omega_{i+1}\cdots\omega_\ell} zv \frac{1}{a_{i,j_i}}\,]^2_{\{i,\ell+1\}}$. In this case, we must cancel all ω_i by combining to

$$[\,\frac{1}{\omega_1\cdots\omega_{i-1}\omega_{i+1}\cdots\omega_\ell} zv \frac{1}{a_{i,j_i}}\,]^2_{\{i,\ell+1\}} \cdot \prod_{k=[1,\ell]\setminus\{i\}} [\,\omega_k a^{e_k}_{k,j_k}\,]^2_{\{k\}} = [\,zv\frac{1}{a_{i,j_i}} \prod_{k=[1,\ell]\setminus\{i\}} a^{e_k}_{k,j_k}\,]^2_{[1,\ell+1]}.$$

  Then, we must use $[\,\frac{c_1\cdots c_{\ell+1}}{v}\,]^2_{[\ell+2,3\ell]}$ to cancel out v. But this would result in a term that always contain $\frac{1}{a_{i,j_i}}$ for some i, j_i that cannot be canceled.

- $[\,Y_1\,]^2_{T_1} = [\,\frac{1}{\mu_1\cdots\mu_\ell} z\,]^2_{\{\ell+1\}}$. In this case, we must cancel all μ_i by combining to

$$[\,\mu_1 a^{e_1}_{1,j_1}\,]^2_{\{1\}} \cdots [\,\mu_\ell a^{e_\ell}_{\ell,j_\ell}\,]^2_{\{\ell\}} \cdot [\,\frac{1}{\mu_1\cdots\mu_\ell} z\,]^2_{\{\ell+1\}} = [\,z a^{e_1}_{1,j_1} \cdots a^{e_\ell}_{\ell,j_\ell}\,]^2_{[1,\ell+1]}.$$

  Then, to obtain $[\,zc_1 \cdots c_{\ell+1}\,]^2_{[1,3\ell]}$, we must pick the rest of combination to contain $c_1 \cdots c_{\ell+1}$, encoded in level $[\ell+2, 3\ell]$. The only possible combinations having this are

$$[\,c_1 \cdots c_{i+1} a^{e_i}_{i,j_i} \frac{d_{i,j_i}}{d_{i,j'_i}}\,]^2_{\substack{[\ell+2,\ell+1+i]\cup\\ [2\ell+2,2\ell+i]}} \cdot \left(\prod_{\iota\in[i+1,\ell]} [\,a^{e_\iota}_{\iota,j_\iota} d_{\iota,j_\iota}\,]^2_{\{\ell+1+\iota\}} \right) \cdot \left(\prod_{\iota\in[i+1,\ell]} [\,\frac{c_{\iota+1}}{d_{\iota,j_\iota}}\,]^2_{\{2\ell+\iota\}} \right)$$

  for any $i \in [1,\ell]$. But this would result in a term that always contains $d_{i,j_i}/d_{i,j'_i}$ for some $j_i \neq j'_i$, which cannot be canceled out.

- $[\,Y_1\,]^2_{T_1} = [\,zc_1\frac{a^{e_i}_{i,j_i}}{a^{e'_i}_{i,j'_i}}d_{i,j_i}\,]^2_{\{i,\ell+1\}}$, where $(e_i, e'_i) \in \mathsf{E}^\star$. In this case, to cover level $[1, \ell+1]$, we have only combinations:

$$[\,zc_1\frac{a^{e_i}_{i,j_i}}{a^{e'_i}_{i,j'_i}}d_{i,j_i}\,]^2_{\{i,\ell+1\}} \cdot \prod_{k\in[1,\ell]\smallsetminus\{i\}} [\,\frac{a^{e_k}_{k,j_k}}{a^{e'_k}_{k,j'_k}}d_{k,j_k}\,]^2_{\{k\}}$$

  Note that we cannot multiply $[\,Y_1\,]^2_{\{i,\ell+1\}}$ with other terms of level $\subset [1, \ell+1]$ since they would contain some $\mu_\iota, \nu_\iota, \omega_\iota$, and to cancel, we must use another element that contains their inverse, but it is encoded in level $\{i, \ell+1\}$, and we cannot multiply to. Also note that we do not possess $[\,1\,]^2_S$ for any $S \subset [1, \ell+1]$. Next, to obtain $[\,zc_1 \cdots c_{\ell+1}\,]^2_{[1,3\ell]}$, the rest of combination must contain $c_2 \cdots c_{\ell+1}$, encoded in level $[\ell+2, 3\ell]$. The only possible combinations that containing this are

$$[\,\frac{c_2}{d_{1,j_1}}\,]^2_{[\ell+2,2\ell+1]} \cdot [\,\frac{c_3}{d_{2,j_2}}\,]^2_{\{2\ell+2\}} \cdots [\,\frac{c_{\ell+1}}{d_{\ell,j_\ell}}\,]^2_{\{3\ell\}}.$$

  But this would result in a term that always contains $a^{e_i}_{i,j_i}/a^{e'_i}_{i,j'_i}$ for some $i \in [1, \ell]$, $j_i \neq j'_i$, $(e_i, e'_i) \in \mathsf{E}^\star$, which cannot be canceled out since $(0, 0) \notin \mathsf{E}^\star$.

This concludes all the cases and hence the proof. □

Lemma 11. *The* EMDDH1-dual *Assumption holds in the generic graded encoding model.*

Proof. It suffices to prove that there is no $[\,X_1\,]^2_{S_1}, \ldots, [\,X_k\,]^2_{S_k}$, $[\,Y_1\,]^2_{T_1}, \ldots, [\,Y_t\,]^2_{T_t}$ in $\bar{D}$ such that

$$[\,bz\,]^2_{[1,\ell+1]} \cdot [\,X_1\,]^2_{S_1} \cdots [\,X_k\,]^2_{S_k} = [\,Y_1\,]^2_{T_1} \cdots [\,Y_t\,]^2_{T_t}$$

where $\bigcup_{j\in[1,k]} S_j = [\ell+2, 3\ell]$ and $\bigcup_{j\in[1,t]} T_j = [1, 3\ell]$ (such that for $j \neq j'$, $S_j \cap S_{j'} = \emptyset$, $T_j \cap T_{j'} = \emptyset$). We observe that since b appears only in $[\,\frac{c_1\cdots c_{\ell+1}}{b}\,]^2_{[\ell+2,3\ell]}$ and $[\,b\,]^2_{[1,\ell+1]}$ in $\bar{D}$, to make the equation hold, it must be that either $[\,X_1\,]^2_{S_1} = [\,\frac{c_1\cdots c_{\ell+1}}{b}\,]^2_{[\ell+2,3\ell]}$ or $[\,Y_1\,]^2_{T_1} = [\,b\,]^2_{[1,\ell+1]}$ (wlog for the index of X, Y). We first assume the latter case. This implies that X_1 must contain z^{-1} or Y_2 must contain z for some j so as to make it hold. There is no such element in the first case. For the second case, there is also no z encoded in level $\subseteq [\ell+2, 3\ell]$. Therefore, by contradiction, the assumption is wrong, and it must be the case that $[\,X_1\,]^2_{S_1} = [\,\frac{c_1\cdots c_{\ell+1}}{b}\,]^2_{[\ell+2,3\ell]}$. In this case, the product in the equation becomes $[\,bz\,]^2_{[1,\ell+1]} \cdot [\,\frac{c_1\cdots c_{\ell+1}}{b}\,]^2_{[\ell+2,3\ell]} = [\,zc_1 \cdots c_{\ell+1}\,]^2_{[1,3\ell]}$. From this point on, the analysis is exactly the same as Lemma 10. □

B Security Proof for KP-ABE

B.1 Normal to Semi-functional Ciphertext

Lemma 12 (G_{real} to G_0)**.** *For any adversary $\mathcal{A}$, there exists an algorithm $\mathcal{B}$ that breaks the Subgroup Decision Assumption 1 with* $|\mathsf{G}_{\text{real}}\mathsf{Adv}^{(n,\ell)\text{-KPABE}}_{\mathcal{A}}(\lambda) - \mathsf{G}_0\mathsf{Adv}^{(n,\ell)\text{-KPABE}}_{\mathcal{A}}(\lambda)| \leq \mathsf{Adv}^{\mathsf{SD1}}_{\mathcal{B}}(\lambda)$.

Proof. The algorithm $\mathcal{B}$ obtains an input (D, Z) from the SD1 Assumption, where D consists of param, $\{[\,1\,]^1_{\{j\}}, [\,1\,]^3_{\{j\}}\}_{j\in[1,3\ell]}$, $[\,b\,]^{1,2}_{[\ell+2,3\ell]}$. $\mathcal{B}$ needs to guess whether $Z = [\,z\,]^1_{[1,\ell+1]}$ or $Z = [\,z\,]^{1,2}_{[1,\ell+1]}$. We may write $Z = [\,z_1\,]^1_{[1,\ell+1]} + [\,z_2\,]^2_{[1,\ell+1]}$, and B's task is equivalent to guess if $z_2 = 0$ or $z_2 \in_R \mathcal{R}$.

Setup. The algorithm $\mathcal{B}$ simulates $\mathsf{SFSetup}(1^\lambda, 3\ell, 3)$ as follows. First, $\mathcal{B}$ samples $[\,\tilde{\alpha}\,]_\emptyset \leftarrow \mathsf{Samp}(\mathsf{param})$ and sets $[\,\alpha\,]^{1,2}_{[\ell+2,3\ell]} = [\,\tilde{\alpha}\,]_\emptyset \cdot [\,b\,]^{1,2}_{[\ell+2,3\ell]}$ for MSK, and $[\,\alpha\,]^1_{[1,3\ell]} = [\,\alpha\,]^{1,2}_{[\ell+2,3\ell]} \cdot [\,1\,]^1_{[1,\ell+1]}$ for

PK. Next, we observe that from $[\,1\,]^1_{\{i\}}, [\,1\,]^3_{\{i\}}$ for all $i \in [1, 3\ell]$, we can compute $[\,1\,]^1_S$ and $[\,1\,]^3_S$ for any S. This is done by computing $\prod_{i\in S}[\,1\,]^j_{\{i\}} = [\,1\,]^j_S$. For each variable y in $H := \{h_1, \dots, h_n, \phi_1, \phi_2\}$, $\mathcal{B}$ does as follows. It samples $[\,\tilde{y}\,]_\emptyset \leftarrow \mathsf{Samp}(\mathsf{param})$. For each $[\,y\,]^1_S$ term that appears in $\mathsf{PK}, \mathsf{MSK}$, $\mathcal{B}$ sets $[\,y\,]^1_S = [\,\tilde{y}\,]_\emptyset \cdot [\,1\,]^1_S$. Hence we have $y = \tilde{y} \bmod N_1$. Note that these can be computed since $\mathcal{B}$ possesses $[\,1\,]^1_S$. All these terms completely define $\mathsf{PK}, \mathsf{MSK}$. PK is given to $\mathcal{A}$.

Similarly, for each variable $\hat{y} \in \hat{H} := \{\hat{h}_1, \dots, \hat{h}_n, \hat{\phi}_1, \hat{\phi}_2\}$, we also define each $[\,\hat{y}\,]^2_S$ term that appears in $\widehat{\mathsf{PK}}$ by *implicitly* setting $[\,\hat{y}\,]^2_S = [\,\tilde{y}\,]_\emptyset \cdot [\,1\,]^2_S$. Note that these are not computable, since $\mathcal{B}$ does not possess any of $[\,1\,]^2_S$. This implicit definition will be used when defining the challenge ciphertext. From this we have $\hat{y} = \tilde{y} \bmod N_2$. Therefore, due to CRT, each $y \in H$ and $\hat{y} \in \hat{H}$ distribute independently, as required by definition of $\mathsf{SFSetup}$. We note that $\widehat{\mathsf{MSK}}_{\mathsf{base}}, \widehat{\mathsf{MSK}}_{\mathsf{aux}}$ are also completely defined, but they are not used here in $\mathsf{G}_{\mathrm{real}}$ or G_0.

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

Challenge. The adversary $\mathcal{A}$ outputs messages $M_0, M_1 \in \{0,1\}^\lambda$ along with a target string $x^\star$. $\mathcal{B}$ chooses $\mathfrak{b} \xleftarrow{\$} \{0,1\}$. $\mathcal{B}$ samples $[\,\tilde{t}\,]_\emptyset, [\,\tilde{s}\,]_\emptyset \leftarrow \mathsf{Samp}(\mathsf{param})$. $\mathcal{B}$ then sets

$$\begin{aligned}
C_0 &= \mathsf{Ext}(\mathsf{param}, [\,\alpha\,]^{1,2}_{[\ell+2,3\ell]} \cdot [\,\tilde{t}\,]_\emptyset \cdot Z) \oplus M_{\mathfrak{b}} &&= \mathsf{Ext}(\mathsf{param}, [\,\alpha\tilde{t}z_1\,]^1_{[1,3\ell]} + [\,\alpha\tilde{t}z_2\,]^2_{[1,3\ell]}) \oplus M_{\mathfrak{b}},\\
C &= [\,\tilde{s}\,]_\emptyset \cdot Z &&= [\,\tilde{s}z_1\,]^1_{[1,\ell+1]} + [\,\tilde{s}z_2\,]^2_{[1,\ell+1]},\\
C_j &= [\,\tilde{h}_j\,]_\emptyset \cdot [\,\tilde{s}\,]_\emptyset \cdot Z &&= [\,h_j\tilde{s}z_1\,]^1_{[1,\ell+1]} + [\,\hat{h}_j\tilde{s}z_2\,]^2_{[1,\ell+1]},\\
T_1 &= [\,\tilde{t}\,]_\emptyset \cdot Z &&= [\,\tilde{t}z_1\,]^1_{[1,\ell+1]} + [\,\tilde{t}z_2\,]^2_{[1,\ell+1]},\\
T_2 &= \Big([\,\tilde{\phi}_2\,]_\emptyset \cdot [\,\tilde{t}\,]_\emptyset + [\,\tilde{\phi}_1\,]_\emptyset \cdot [\,\tilde{s}\,]_\emptyset\Big) \cdot Z &&= [\,\phi_2\tilde{t}z_1 + \phi_1\tilde{s}z_1\,]^1_{[1,\ell+1]} + [\,\hat{\phi}_2\tilde{t}z_2 + \hat{\phi}_1\tilde{s}z_2\,]^2_{[1,\ell+1]}.
\end{aligned} \tag{14}$$

We can see that CT distributes as a normal or semi-functional ciphertext with $s = \tilde{s}z_1 \bmod N_1$ and $t = \tilde{t}z_1 \bmod N_1$. Moreover, if $z_2 = 0$, then CT distributes as a normal ciphertext. Otherwise $z_2 \in_R \mathcal{R}$, then CT distributes as a semi-functional ciphertext with $\hat{s} = \tilde{s}z_2 \bmod N_2$ and $\hat{t} = \tilde{t}z_2 \bmod N_2$. From CRT, we have that $s, \hat{s}, t, \hat{t}$ are independently distributed as required.

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

Guess. The algorithm $\mathcal{B}$ has properly simulated $\mathsf{G}_{\mathrm{real}}$ if $z_2 = 0$, and G_0 if $z_2 \in_R \mathcal{R}$. Hence, $\mathcal{B}$ can use the output of $\mathcal{A}$ to break the SD1 Assumption. □

B.2 Normal to Type-1 Semi-functional Key in Phase 1

Lemma 13 ($\mathsf{G}_{k-1,3}$ to $\mathsf{G}_{k,1}$). *For any adversary $\mathcal{A}$, there exists an algorithm $\mathcal{B}$ that breaks the Subgroup Decision Assumption 2 with* $|\mathsf{G}_{k-1,3}\mathsf{Adv}^{(n,\ell)\text{-}\mathsf{KPABE}}_{\mathcal{A}}(\lambda) - \mathsf{G}_{k,1}\mathsf{Adv}^{(n,\ell)\text{-}\mathsf{KPABE}}_{\mathcal{A}}(\lambda)| \le \mathsf{Adv}^{\mathsf{SD2}}_{\mathcal{B}}(\lambda)$.

Proof. The algorithm $\mathcal{B}$ obtains an input $(D, Z = \{Z_i\}_{i\in[\ell+2,3\ell]})$ from the $\mathsf{SD2}$ Assumption, where D consists of $\mathsf{param}, \{[\,1\,]^1_{\{j\}}, [\,1\,]^3_{\{j\}}\}_{j\in[1,3\ell]}, [\,a\,]^{1,2}_{[1,\ell+1]}, [\,b\,]^{1,2}_{[\ell+2,3\ell]}, [\,c\,]^{2,3}_{[\ell+2,3\ell]}$. $\mathcal{B}$ needs to guess whether $Z = \{[\,z_i\,]^{1,2}_{\{i\}}\}_{i\in[\ell+2,3\ell]}$ or $Z = \{[\,z_i\,]^{1,2,3}_{\{i\}}\}_{i\in[\ell+2,3\ell]}$. When we write $Z_i = [\,z_{i,1}\,]^{1,3}_{\{i\}} + [\,z_{i,2}\,]^2_{\{i\}}$ for $i \in [\ell+2, 3\ell]$, B's task is equivalent to guess whether $z_{i,2} = 0$ for all $i \in [\ell+2, 3\ell]$ or $z_{i,2} \in_R \mathcal{R}$ for all $i \in [\ell+2, 3\ell]$.

Setup. The algorithm $\mathcal{B}$ simulates $\mathsf{SFSetup}(1^\lambda, 3\ell, 3)$ in exactly the same manner as in the previous proof (of Lemma 12). These completely defines $\mathsf{PK}, \mathsf{MSK}, \widehat{\mathsf{PK}}, \widehat{\mathsf{MSK}}_{\mathsf{base}}, \widehat{\mathsf{MSK}}_{\mathsf{aux}}$, albeit only $\mathsf{PK}, \mathsf{MSK}$ are computable.

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

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

[Case $j < k$]. $\mathcal{B}$ generates a type-3 semi-functional key as follows. First it generates a normal key $\big(\{\mathcal{K}_w\}_{w\in\mathsf{Nodes}}, D_1, D_2, D_3\big) \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, f^{(j)})$. A type-3 semi-functional key is different from a normal key at only D_1. $\mathcal{B}$ samples $[\,\tilde{\beta}_j\,]_\emptyset \leftarrow \mathsf{Samp}(\mathsf{param})$ and sets

$$\bar{D}_1 = D_1 + ([\,\tilde{\beta}_j\,]_\emptyset \cdot [\,c\,]^{2,3}_{[\ell+2,3\ell]}) = D_1 + [\,\tilde{\beta}_j c\,]^{2,3}_{[\ell+2,3\ell]}. \tag{15}$$

$\mathcal{B}$ returns $\mathsf{SK}_j = \big(\{\mathcal{K}_w\}_{w\in\mathsf{Nodes}}, \bar{D}_1, D_2, D_3\big)$. This is a type-3 semi-functional key with $\beta_j = \tilde{\beta}_j c \bmod N_2$.

[Case $j = k$]. $\mathcal{B}$ generates a normal key or a type-1 semi-functional key as follows. From Z_i for all $i \in [\ell+2, 3\ell]$, compute $Z_S := \prod_{i\in S} Z_i$ for $S = [\ell+2, 2\ell+1]$ to $[\ell+2, 3\ell]$. We have $Z_S = [\,\prod_{i\in S} z_{i,1}\,]^{1,3}_S + [\,\prod_{i\in S} z_{i,2}\,]^2_S = [\,z_{S,1}\,]^{1,3}_S + [\,z_{S,2}\,]^2_S$ by orthogonality, where we denote $z_{S,j} := \prod_{i\in S} z_{i,j}$, for $j = 1, 2$. $\mathcal{B}$ uses $\mathsf{Samp}(\mathsf{param})$ to sample $[\,\tilde{r}\,]_\emptyset, \{[\,\tilde{v}_w\,]_\emptyset\}_{w\in\mathsf{Inputs}}, \{[\,\tilde{\alpha}_w\,]_\emptyset\}_{w\in\mathsf{Nodes}}$, and $\{[\,\tilde{\ell}_w\,]_\emptyset, [\,\tilde{r}_w\,]_\emptyset\}_{w\in\mathsf{Gates}}$. $\mathcal{B}$ then sets

$$\begin{array}{lll}
 & D_1' & = [\,\alpha\,]^{1,2}_{[\ell+2,3\ell]} + [\,\tilde{\phi}_2\,]_\emptyset \cdot [\,\tilde{r}\,]_\emptyset \cdot Z_{[\ell+2,3\ell]} \\
 & D_2' & = [\,\tilde{r}\,]_\emptyset \cdot Z_{[\ell+2,3\ell]} \\
 & D_3' & = ([\,\tilde{\phi}_1\,]_\emptyset \cdot [\,\tilde{r}\,]_\emptyset - [\,\tilde{\alpha}_{w_{\mathsf{top}}}\,]_\emptyset) \cdot Z_{[\ell+2,3\ell]} \\
\forall_{w\in\mathsf{Inputs}} & U_w' & = [\,\tilde{v}_w\,]_\emptyset \cdot Z_{[\ell+2,2\ell+1]} \\
\forall_{w\in\mathsf{Inputs}} & K_w' & = ([\,\tilde{\alpha}_w\,]_\emptyset + [\,\tilde{h}_w\,]_\emptyset \cdot [\,\tilde{v}_w\,]_\emptyset) \cdot Z_{[\ell+2,2\ell+1]} \\
\forall_{w\in\mathsf{Gates}} & L_w' & = [\,\tilde{\ell}_w\,]_\emptyset \cdot Z_{2\ell+i_w} \\
\forall_{w\in\mathsf{Gates}} & R_w' & = [\,\tilde{r}_w\,]_\emptyset \cdot Z_{2\ell+i_w} \\
\forall_{w\in\mathsf{GatesOR}} & K_{w,1}' & = ([\,\tilde{\alpha}_w\,]_\emptyset + [\,\tilde{\alpha}_{\mathsf{L}(w)}\,]_\emptyset \cdot [\,\tilde{\ell}_w\,]_\emptyset) \cdot Z_{[\ell+2,2\ell+i_w]} \\
\forall_{w\in\mathsf{GatesOR}} & K_{w,2}' & = ([\,\tilde{\alpha}_w\,]_\emptyset + [\,\tilde{\alpha}_{\mathsf{R}(w)}\,]_\emptyset \cdot [\,\tilde{r}_w\,]_\emptyset) \cdot Z_{[\ell+2,2\ell+i_w]} \\
\forall_{w\in\mathsf{GatesAND}} & K_w' & = ([\,\tilde{\alpha}_w\,]_\emptyset + [\,\tilde{\alpha}_{\mathsf{L}(w)}\,]_\emptyset \cdot [\,\tilde{\ell}_w\,]_\emptyset + [\,\tilde{\alpha}_{\mathsf{R}(w)}\,]_\emptyset \cdot [\,\tilde{r}_w\,]_\emptyset) \cdot Z_{[\ell+2,2\ell+i_w]},
\end{array} \tag{16}$$

where we denote $i_w = \mathsf{Depth}(w)$. $\mathcal{B}$ then adds random masks from subring $\mathbb{Z}_{N_3}$ as in KeyGen, and returns SK_i. Analogously to Equation (14) in the previous proof, we can deduce the following. The simulated SK_i distributes as a normal or type-1 semi-functional with $r = \tilde{r} z_{[\ell+2,3\ell],1} \bmod N_1$, $v_w = \tilde{v}_w z_{[\ell+2,2\ell+1],1} \bmod N_1$, $\alpha_w = \tilde{\alpha}_w z_{[\ell+2,2\ell+i_w],1} \bmod N_1$, $\ell_w = \tilde{\ell}_w z_{2\ell+i_w,1} \bmod N_1$, and $r_w = \tilde{r}_w z_{2\ell+i_w,1} \bmod N_1$. Moreover, if $z_{i,2} = 0$ for all $i \in [\ell+2, 3\ell]$, then SK_i distributes as a normal key. Otherwise $z_{i,2} \in_R \mathcal{R}$ for all $i \in [\ell+2, 3\ell]$, then SK_i distributes as a type-1 semi-functional key with $\hat{r} = \tilde{r} z_{[\ell+2,3\ell],2} \bmod N_2$, $\hat{v}_w = \tilde{v}_w z_{[\ell+2,2\ell+1],2} \bmod N_2$, $\hat{\alpha}_w = \tilde{\alpha}_w z_{[\ell+2,2\ell+i_w],2} \bmod N_2$, $\hat{\ell}_w = \tilde{\ell}_w z_{2\ell+i_w,2} \bmod N_2$, and $\hat{r}_w = \tilde{r}_w z_{2\ell+i_w,2} \bmod N_2$. Again due to CRT, any hatted variable (*e.g.*, $\hat{\alpha}_w$) is independent from its non-hatted counterpart (*e.g.*, α_w), as required.

Challenge. The adversary $\mathcal{A}$ outputs messages $M_0, M_1 \in \{0,1\}^\lambda$ along with a target string $x^\star$. $\mathcal{B}$ chooses $\mathfrak{b} \xleftarrow{\$} \{0,1\}$. $\mathcal{B}$ samples $[\,\tilde{t}\,]_\emptyset, [\,\tilde{s}\,]_\emptyset \leftarrow \mathsf{Samp}(\mathsf{param})$. $\mathcal{B}$ then sets

$$\begin{array}{ll}
C_0 = \mathsf{Ext}(\mathsf{param}, [\,\alpha\,]^{1,2}_{[\ell+2,3\ell]} \cdot [\,\tilde{t}\,]_\emptyset \cdot [\,a\,]^{1,2}_{[1,\ell+1]}) \oplus M_{\mathfrak{b}} & = \mathsf{Ext}(\mathsf{param}, [\,\alpha \tilde{t} a_1\,]^1_{[1,3\ell]} + [\,\alpha \tilde{t} a_2\,]^2_{[1,3\ell]}) \oplus M_{\mathfrak{b}}, \\
C \;\; = [\,\tilde{s}\,]_\emptyset \cdot [\,a\,]^{1,2}_{[1,\ell+1]} & = [\,\tilde{s} a_1\,]^1_{[1,\ell+1]} + [\,\tilde{s} a_2\,]^2_{[1,\ell+1]}, \\
C_j = [\,\tilde{h}_j\,]_\emptyset \cdot [\,\tilde{s}\,]_\emptyset \cdot [\,a\,]^{1,2}_{[1,\ell+1]} & = [\,h_j \tilde{s} a_1\,]^1_{[1,\ell+1]} + [\,\hat{h}_j \tilde{s} a_2\,]^2_{[1,\ell+1]}, \\
T_1 = [\,\tilde{t}\,]_\emptyset \cdot [\,a\,]^{1,2}_{[1,\ell+1]} & = [\,\tilde{t} a_1\,]^1_{[1,\ell+1]} + [\,\tilde{t} a_2\,]^2_{[1,\ell+1]}, \\
T_2 = \Big([\,\tilde{\phi}_2\,]_\emptyset \cdot [\,\tilde{t}\,]_\emptyset + [\,\tilde{\phi}_1\,]_\emptyset \cdot [\,\tilde{s}\,]_\emptyset\Big) \cdot [\,a\,]^{1,2}_{[1,\ell+1]} & = [\,\phi_2 \tilde{t} a_1 + \phi_1 \tilde{s} a_1\,]^1_{[1,\ell+1]} + [\,\hat{\phi}_2 \tilde{t} a_2 + \hat{\phi}_1 \tilde{s} a_2\,]^2_{[1,\ell+1]},
\end{array} \tag{17}$$

where we write $[\,a\,]^{1,2}_{[1,\ell+1]} = [\,a_1\,]^{1}_{[1,\ell+1]} + [\,a_2\,]^{2}_{[1,\ell+1]}$. We can see that CT distributes as a semi-functional ciphertext with $s = \tilde{s}a_1 \bmod N_1$, $t = \tilde{t}a_1 \bmod N_1$, $\hat{s} = \tilde{s}a_2 \bmod N_2$, and $\hat{t} = \tilde{t}a_2 \bmod N_2$. From CRT, we have that $s, \hat{s}, t, \hat{t}$ are independently distributed as required.

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

Guess. The algorithm $\mathcal{B}$ has properly simulated $\mathsf{G}_{k-1,3}$ if $z_{i,2} = 0$ for all $i \in [\ell+2, 3\ell]$, and $\mathsf{G}_{k,1}$ if $z_{i,2} \in_R \mathcal{R}$ for all $i \in [\ell+2, 3\ell]$. Hence, $\mathcal{B}$ can use the output of $\mathcal{A}$ to break the $\mathsf{SD2}$ Assumption. $\square$

B.3 Type-2 to Type-3 Semi-functional Key in Phase 1

Lemma 14 ($\mathsf{G}_{k,2}$ to $\mathsf{G}_{k,3}$). *For any adversary $\mathcal{A}$, there exists an algorithm $\mathcal{B}$ that breaks the Subgroup Decision Assumption 2 with* $|\mathsf{G}_{k,2}\mathsf{Adv}^{(n,\ell)\text{-}\mathsf{KPABE}}_{\mathcal{A}}(\lambda) - \mathsf{G}_{k,3}\mathsf{Adv}^{(n,\ell)\text{-}\mathsf{KPABE}}_{\mathcal{A}}(\lambda)| \le \mathsf{Adv}^{\mathsf{SD2}}_{\mathcal{B}}(\lambda)$.

Proof. The proof is exactly the same as that of Lemma 13, where we moved from normal to type-1 semi-functional key, except only one simulated element D'_1 for the k-th key query. We modify D'_1 from Equation (16) to the following. $\mathcal{B}$ samples $[\,\tilde{\beta}_i\,]_\emptyset \leftarrow \mathsf{Samp}(\mathsf{param})$ and sets

$$D'_1 = [\,\alpha\,]^{1,2}_{[\ell+2,3\ell]} + [\,\tilde{\phi}_2\,]_\emptyset \cdot [\,\tilde{r}\,]_\emptyset \cdot Z_{[\ell+2,3\ell]} + [\,\tilde{\beta}_i\,]_\emptyset \cdot [\,c\,]^{2,3}_{[\ell+2,3\ell]}, \tag{18}$$

where we note that the difference is the last term in the sum, which is $[\,\tilde{\beta}_i c\,]^{2,3}_{[\ell+2,3\ell]}$. The simulated key SK_i is a type-2 or type-3 with the same randomness as stated in the proof of Lemma 13 but now also with $\beta_j = \tilde{\beta}_j c \bmod N_2$. The rest of the proof follows exactly the same as that of Lemma 13. $\square$

B.4 Normal to Type-1 Semi-functional Keys in Phase 2

Lemma 15 ($\mathsf{G}_{q_1,3}$ to G_{q_1+1}). *For any adversary $\mathcal{A}$, there exists an algorithm $\mathcal{B}$ that breaks the Subgroup Decision Assumption 2 with* $|\mathsf{G}_{q_1,3}\mathsf{Adv}^{(n,\ell)\text{-}\mathsf{KPABE}}_{\mathcal{A}}(\lambda) - \mathsf{G}_{q_1+1}\mathsf{Adv}^{(n,\ell)\text{-}\mathsf{KPABE}}_{\mathcal{A}}(\lambda)| \le \mathsf{Adv}^{\mathsf{SD2}}_{\mathcal{B}}(\lambda)$.

Proof. The proof is exactly the same as when we modify normal to type-1 semi-functional key in phase 1 (the proof of Lemma 13), except that this time we modify the post-challenge keys *all at once*, instead of one key at a time. In particular, the simulation of $\mathsf{PK}, \mathsf{MSK}$ and the challenge ciphertext is exactly the same. The simulation for type-3 semi-functional key queries in phase 1 is done using $[\,c\,]^{2,3}_{[\ell+2,3\ell]}$ as in Eq.(15). The simulation of every key in phase 2 can be done using the problem instance Z in exactly the same way as in Eq.(16), with a remark that the randomness $([\,\tilde{r}\,]_\emptyset, \{[\,\tilde{v}_w\,]_\emptyset\}_{w\in\mathsf{Inputs}}, \{[\,\tilde{\alpha}_w\,]_\emptyset\}_{w\in\mathsf{Nodes}}, \{[\,\tilde{\ell}_w\,]_\emptyset, [\,\tilde{r}_w\,]_\emptyset\}_{w\in\mathsf{Gates}})$ is sampled fresh for each key. $\square$

B.5 Type-1 to Type-2 Semi-functional Keys in Phase 2

Lemma 16 (G_{q_1+1} to G_{q_1+2}). *For any adversary $\mathcal{A}$, there exists an algorithm $\mathcal{B}$ that breaks the ℓ-EMDDH2 with* $|\mathsf{G}_{q_1+1}\mathsf{Adv}^{(n,\ell)\text{-}\mathsf{KPABE}}_{\mathcal{A}}(\lambda) - \mathsf{G}_{q_1+2}\mathsf{Adv}^{(n,\ell)\text{-}\mathsf{KPABE}}_{\mathcal{A}}(\lambda)| \le \mathsf{Adv}^{\ell\text{-}\mathsf{EMDDH2}}_{\mathcal{B}}(\lambda)$, *where ℓ is the bounded depth.*

Proof. The algorithm $\mathcal{B}$ obtains an input (D, Z) from the $\mathsf{EMDDH2}$ Assumption. Denote $Z = [\,\delta + c_1 \cdots c_{\ell+1} b\,]^{2}_{[\ell+2,3\ell]}$. Its task is to guess whether $\delta = 0$ or $\delta \in_R \mathcal{R}$. $\mathcal{B}$ will implicitly define $\beta = \delta$ in the simulation for all the post-challenge keys.

Setup. The algorithm $\mathcal{B}$ simulates $\mathsf{PK}, \mathsf{MSK}, \widehat{\mathsf{MSK}}_{\mathsf{base}}$ in exactly the same manner as when we modified semi-functional key from type-1 to type-2 in phase 1 (the full proof of Lemma 3 in §7.3). In particular,$\mathcal{B}$ knows all $[\,y\,]_\emptyset$ for all variables y in $H := \{h_1, \ldots, h_n, \phi_1, \phi_2\}$. $\mathcal{B}$ also knows $[\,\alpha\,]_\emptyset$, which will be used for simulating challenge ciphertext. Note that $\mathcal{B}$ will not define all the hatted variables (and hence, $\widehat{\mathsf{PK}}, \widehat{\mathsf{MSK}}_{\mathsf{aux}}$) until the first query that requires using them, which is the challenge query below.

Phase 1. When $\mathcal{A}$ makes the j-th key query for $f^{(j)}$, $\mathcal{B}$ samples $[\,\beta_j\,]_\emptyset \leftarrow \mathsf{Samp}(\mathsf{param})$ and generates a type-3 semi-functional key by setting $\mathsf{SK}_j \leftarrow \mathsf{SFKeyGen}(\mathsf{MSK}, f^{(j)}, \widehat{\mathsf{MSK}}_{\mathsf{base}}, -, 3, [\,\beta_j\,]_\emptyset)$.

Challenge. The adversary $\mathcal{A}$ outputs messages $M_0, M_1 \in \{0,1\}^\lambda$ along with a target string $x^\star$. $\mathcal{B}$ chooses $\mathfrak{b} \xleftarrow{\$} \{0,1\}$. $\mathcal{B}$ first computes a normal ciphertext $(C_0, \mathcal{C}) \leftarrow \mathsf{Encrypt}(\mathsf{PK}, x^\star, M_\mathfrak{b})$, where we denote $\mathcal{C} = (C, \{C_j\}_{j\in A_x}, T_1, T_2)$. $\mathcal{B}$ also computes the message mask $K = [\,\alpha t\,]^1_{[1,3\ell]}$ in the process. $\mathcal{B}$ then produces semi-functional components as follows.

(Programming Parameters and Randomness in Ciphertext). $\mathcal{B}$ first samples $[\,h'_1\,]_\emptyset, \ldots, [\,h'_n\,]_\emptyset$, $[\,\phi'_1\,]_\emptyset, [\,\phi'_2\,]_\emptyset \leftarrow \mathsf{Samp}(\mathsf{param})$. $\mathcal{B}$ then implicitly sets

$$\hat{h}_j = \begin{cases} h'_j & \text{if } j \in A_{x^\star}, \\ h'_j + c_1 & \text{if } j \notin A_{x^\star}. \end{cases}$$

$\mathcal{B}$ also samples $[\,\phi'_1\,]_\emptyset, [\,\phi'_2\,]_\emptyset \leftarrow \mathsf{Samp}(\mathsf{param})$ and implicitly sets $\hat{\phi}_1 = \phi'_1 + c_1$ and $\hat{\phi}_2 = \phi'_2 + c_1 z$. For the randomness in ciphertext, $\mathcal{B}$ implicitly sets $\hat{t} = 1$, $\hat{s} = -z$ (this will be re-randomized later).

(Simulating Ciphertext). From the above (implicit) definitions, the semi-functional component for ciphertext is well defined. $\mathcal{B}$ computes them as follows.

$$\hat{C} = -[\,z\,]^2_{[1,\ell+1]} = [\,\hat{s}\,]^2_{[1,\ell+1]}, \quad \forall_{j\in A_{x^\star}}\ \hat{C}_j = -[\,h'_j\,]_\emptyset \cdot [\,z\,]^2_{[1,\ell+1]} = [\,\hat{h}_j\hat{s}\,]^2_{[1,\ell+1]},$$
$$\hat{T}_1 = [\,1\,]^2_{[1,\ell+1]} = [\,\hat{t}\,]^2_{[1,\ell+1]}, \qquad \hat{T}_2 = [\,\phi'_2\,]_\emptyset \cdot [\,1\,]^2_{[1,\ell+1]} - [\,\phi'_1\,]_\emptyset \cdot [\,z\,]^2_{[1,\ell+1]}. = [\,\hat{\phi}_2\hat{t} + \hat{\phi}_1\hat{s}\,]^2_{[1,\ell+1]},$$

where we notice that in $\hat{T}_2$, a critical term $c_1 z$ from $\hat{\phi}_2\hat{t}$ and $\hat{\phi}_1\hat{s}$ is canceled out. Next, from $[\,\alpha\,]_\emptyset$, $\mathcal{B}$ computes the semi-functional component of message mask $\hat{K} = \hat{T}_1 \cdot [\,\alpha\,]^2_{[\ell+2,3\ell]} = [\,\alpha\hat{t}\,]^2_{[1,3\ell]}$.

(Re-randomizing Ciphertext). The simulated ciphertext is not perfectly distributed yet. $\mathcal{B}$ re-randomize it as follows. $\mathcal{B}$ first samples $[\,s'\,]_\emptyset, [\,t'\,]_\emptyset \leftarrow \mathsf{Samp}(\mathsf{param})$ and computes a new semi-functional component $\hat{\mathcal{C}}'' = (\hat{C}'', \{\hat{C}''_j\}_{j\in A_x}, \hat{T}''_1, \hat{T}''_2)$ that has new randomness $\hat{s}'' = t'\hat{s} + s'$ and $\hat{t}'' = t'$ as follows. $\mathcal{B}$ sets $\hat{T}''_1 = [\,t'\,]_\emptyset \cdot \hat{T}_1$ and $\hat{T}''_2 = [\,t'\,]_\emptyset \cdot \hat{T}_2 + [\,s'\,]_\emptyset \cdot [\,\hat{\phi}_1\,]^2_{[1,\ell+1]}$, and

$$\hat{C}'' = [\,t'\,]_\emptyset \cdot \hat{C} + [\,s'\,]_\emptyset \cdot [\,1\,]^2_{[1,\ell+1]}, \qquad \forall_{j\in A_{x^\star}}\ \hat{C}''_j = [\,t'\,]_\emptyset \cdot \hat{C}_j + [\,s'\,]_\emptyset \cdot [\,\hat{h}_j\,]^2_{[1,\ell+1]}.$$

Note that $[\,\hat{\phi}_1\,]^2_{[1,\ell+1]} = [\,\phi'_1\,]_\emptyset \cdot [\,1\,]^2_{[1,\ell+1]} + [\,c_1\,]^2_{[1,\ell+1]}$ and $[\,\hat{h}_j\,]^2_{[1,\ell+1]} = [\,h'_j\,]_\emptyset \cdot [\,1\,]^2_{[1,\ell+1]}$ for $j \in A_{x^\star}$ are computable. ($[\,c_1\,]^2_{[1,\ell+1]}$ is available from D). $\mathcal{B}$ also computes $\hat{K}'' = [\,t'\,]_\emptyset \cdot \hat{K} = [\,\alpha\hat{t}''\,]^2_{[1,3\ell]}$.

(Returning Ciphertext). $\mathcal{B}$ computes C_0 from the message mask: $C_0 = \mathsf{Ext}(\mathsf{param}, K + \hat{K}'') \oplus M_\mathfrak{b}$. It returns a semi-functional ciphertext as $(C_0, \mathcal{C} + \hat{\mathcal{C}}'')$.

Phase 2. $\mathcal{B}$ implicitly sets $\beta = \delta$. $\mathcal{B}$ will use $Z = [\,\delta + c_1 \cdots c_{\ell+1} z\,]^2_{[\ell+2,3\ell]}$ from the challenge to embed it to each key. When $\mathcal{A}$ makes the j-th key query for $f^{(j)}$, $\mathcal{B}$ generates a type-1 or type-2 semi-functional key as follows. First it generates a normal key $\mathsf{SK}_j \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, f^{(j)})$. A type-1 or type-2 semi-functional key is different from a normal key by having additional N_2 components. $\mathcal{B}$ defines these semi-functional components as follows.

(Programming Randomness in Key). $\mathcal{B}$ implicitly sets $\hat{r} = c_2 \cdots c_{\ell+1}$ and

$$\forall_{w_{1,j}\in\mathsf{Inputs}} : \hat{v}_{w_{1,j}} = \begin{cases} -c_2 & \text{if } j \notin A_{x^\star}, \\ 0 & \text{if } j \in A_{x^\star} \end{cases}, \quad \forall_{w_{i,j}\in\mathsf{Nodes}} : \hat{\alpha}_{w_{i,j}} = \begin{cases} c_1 \cdots c_{i+1} & \text{if } f_{w_{i,j}}(x^\star) = 0, \\ 0 & \text{if } f_{w_{i,j}}(x^\star) = 1. \end{cases}$$

It then sets $\hat{\ell}_{w_{i,j}}, \hat{r}_{w_{i,j}}$ for all $w_{i,j} \in \mathsf{Gates}$ as follows. There are three cases:

- If $f_{w_{i,j}}(x^\star) = 1$, then $\mathcal{B}$ sets $\hat{\ell}_{w_{i,j}} = \hat{r}_{w_{i,j}} = 0$.
- If $f_{w_{i,j}}(x^\star) = 0$ and $\mathsf{GateType}(w_{i,j}) = \mathsf{OR}$, then $\mathcal{B}$ sets $\hat{\ell}_{w_{i,j}} = \hat{r}_{w_{i,j}} = -c_{i+1}$.
- If $f_{w_{i,j}}(x^\star) = 0$ and $\mathsf{GateType}(w_{i,j}) = \mathsf{AND}$, then
  - if $f_{\mathsf{L}(w_{i,j})}(x^\star) = 0$, then $\mathcal{B}$ sets $\hat{\ell}_{w_{i,j}} = -c_{i+1}$ and $\hat{r}_{w_{i,j}} = 0$,
  - otherwise $f_{\mathsf{R}(w_{i,j})}(x^\star) = 0$, then $\mathcal{B}$ sets $\hat{\ell}_{w_{i,j}} = 0$ and $\hat{r}_{w_{i,j}} = -c_{i+1}$.

(Simulating Key). From the above (implicit) definitions, the semi-functional component for key is well defined. $\mathcal{B}$ computes them as follows.

$$\begin{aligned}
\hat{D}_1 &= Z + [\,\phi_2'\,]_\emptyset \cdot [\,c_2 \cdots c_{\ell+1}\,]^2_{[\ell+2,3\ell]} = [\,\beta + \hat{\phi}_2 \hat{r}\,]^2_{[\ell+2,3\ell]},\\
\hat{D}_2 &= [\,c_2 \cdots c_{\ell+1}\,]^2_{[\ell+2,3\ell]} = [\,\hat{r}\,]^2_{[\ell+2,3\ell]},\\
\hat{D}_3 &= [\,\phi_1'\,]_\emptyset \cdot [\,c_2 \cdots c_{\ell+1}\,]^2_{[\ell+2,3\ell]} = [\,\hat{\phi}_1 \hat{r} - \hat{\alpha}_{w_{\ell,1}}\,]^2_{[\ell+2,3\ell]},
\end{aligned}$$

where we note that $[\,c_2 \cdots c_{\ell+1}\,]^2_{[\ell+2,3\ell]} = [\,c_2\,]^2_{[\ell+2,2\ell+1]} [\,c_3\,]^2_{\{2\ell+2\}} \cdots [\,c_{\ell+1}\,]^2_{\{3\ell\}}$ is computable, and that a critical term $\hat{\alpha}_{w_{\ell,1}} = c_1 \cdots c_{\ell+1}$ is canceled out in $\hat{D}_3$. Next, for all $w_{1,j} \in \mathsf{Inputs}$, $\mathcal{B}$ computes $\hat{U}_{w_{1,j}}, \hat{K}_{w_{1,j}}$ as follows. If $j \in A_{x^\star}$, then we have $\hat{v}_{w_{1,j}} = 0$ and $f_{w_{1,j}}(x^\star) = 1$. Hence, $\hat{\alpha}_{w_{1,j}} = 0$. $\mathcal{B}$ thus trivially computes

$$\hat{U}_{w_{1,j}} = [\,0\,]^2_{[\ell+2,2\ell+1]} = [\,\hat{v}_{w_{1,j}}\,]^2_{[\ell+2,2\ell+1]}, \qquad \hat{K}_{w_{1,j}} = [\,0\,]^2_{[\ell+2,2\ell+1]} = [\,\hat{\alpha}_{w_{1,j}} + \hat{h}_j \hat{v}_{w_{1,j}}\,]^2_{[\ell+2,2\ell+1]}.$$

On the other hand, if $j \notin A_{x^\star}$, then we have $\hat{v}_{w_{1,j}} = -c_2$ and $f_{w_{1,j}}(x^\star) = 0$. Hence, $\hat{\alpha}_{w_{1,j}} = c_1 c_2$. In this case, $\mathcal{B}$ computes

$$\hat{U}_{w_{1,j}} = -[\,c_2\,]^2_{[\ell+2,2\ell+1]} = [\,\hat{v}_{w_{1,j}}\,]^2_{[\ell+2,2\ell+1]}, \quad \hat{K}_{w_{1,j}} = -[\,h_j'\,]_\emptyset \cdot [\,c_2\,]^2_{[\ell+2,2\ell+1]} = [\,\hat{\alpha}_{w_{1,j}} + \hat{h}_j \hat{v}_{w_{1,j}}\,]^2_{[\ell+2,2\ell+1]},$$

where we note that since $\hat{h}_j = h_j' + c_1$, a critical term $c_1 c_2$ in $\hat{K}_{w_{1,j}}$ is canceled out.

For each gate $w_{i,j} \in \mathsf{Gates}$, $\mathcal{B}$ computes their corresponding elements as follows. For the case $f_{w_{i,j}}(x^\star) = 1$, we have $\hat{\ell}_{w_{i,j}} = \hat{r}_{w_{i,j}} = 0$ and $\hat{\alpha}_{w_{i,j}} = 0$. Hence, in this case, all the elements are the encodings of zero, which can be trivially constructed. Now, we suppose that $f_{w_{i,j}}(x^\star) = 0$. If $\mathsf{GateType}(w_{i,j}) = \mathsf{OR}$, then we have $\hat{\ell}_{w_{i,j}} = \hat{r}_{w_{i,j}} = -c_{i+1}$ and $f_{\mathsf{L}(w_{i,j})}(x^\star) = f_{\mathsf{R}(w_{i,j})}(x^\star) = 0$. Hence, $\hat{\alpha}_{w_{i,j}} = c_1 \cdots c_{i+1}$ and $\hat{\alpha}_{\mathsf{L}(w_{i,j})} = \hat{\alpha}_{\mathsf{R}(w_{i,j})} = c_1 \cdots c_i$. $\mathcal{B}$ thus computes

$$\begin{aligned}
&\hat{L}_{w_{i,j}} = -[\,c_{i+1}\,]^2_{\{2\ell+i\}} = [\,\hat{\ell}_{w_{i,j}}\,]^2_{\{2\ell+i\}}, \qquad \hat{R}_{w_{i,j}} = -[\,c_{i+1}\,]^2_{\{2\ell+i\}} = [\,\hat{r}_{w_{i,j}}\,]^2_{\{2\ell+i\}},\\
&\hat{K}_{w_{i,j},1} = [\,0\,]^2_{[\ell+2,2\ell+i]} = [\,c_1 \cdots c_{i+1} + (c_1 \cdots c_i)(-c_{i+1})\,]^2_{[\ell+2,2\ell+i]} = [\,\hat{\alpha}_{w_{i,j}} + \hat{\alpha}_{\mathsf{L}(w_{i,j})} \hat{\ell}_{w_{i,j}}\,]^2_{[\ell+2,2\ell+i]},\\
&\hat{K}_{w_{i,j},2} = [\,0\,]^2_{[\ell+2,2\ell+i]} = [\,c_1 \cdots c_{i+1} + (c_1 \cdots c_i)(-c_{i+1})\,]^2_{[\ell+2,2\ell+i]} = [\,\hat{\alpha}_{w_{i,j}} + \hat{\alpha}_{\mathsf{R}(w_{i,j})} \hat{r}_{w_{i,j}}\,]^2_{[\ell+2,2\ell+i]}.
\end{aligned}$$

For the case $\mathsf{GateType}(w_{i,j}) = \mathsf{AND}$, wlog we can assume $f_{\mathsf{L}(w_{i,j})}(x^\star) = 0$. (The case where $f_{\mathsf{L}(w_{i,j})}(x^\star) = 1$ but $f_{\mathsf{R}(w_{i,j})}(x^\star) = 0$ can be done analogously). Therefore, $\hat{\ell}_{w_{i,j}} = -c_{i+1}$, $\hat{r}_{w_{i,j}} = 0$ and $\hat{\alpha}_{\mathsf{L}(w_{i,j})} = c_1 \cdots c_i$. $\mathcal{B}$ thus computes

$$\hat{L}_{w_{i,j}} = -[\,c_{i+1}\,]^2_{\{2\ell+i\}} = [\,\hat{\ell}_{w_{i,j}}\,]^2_{\{2\ell+i\}}, \qquad \hat{R}_{w_{i,j}} = -[\,0\,]^2_{\{2\ell+i\}} = [\,\hat{r}_{w_{i,j}}\,]^2_{\{2\ell+i\}},$$

and

$$\begin{aligned}
\hat{K}_{w_{i,j}} &= [\,0\,]^2_{[\ell+2,2\ell+i]} = [\,c_1 \cdots c_{i+1} + (c_1 \cdots c_i)(-c_{i+1}) + \hat{\alpha}_{\mathsf{R}(w_{i,j})} \cdot 0\,]^2_{[\ell+2,2\ell+i]}\\
&= [\,\hat{\alpha}_{w_{i,j}} + \hat{\alpha}_{\mathsf{L}(w_{i,j})} \hat{\ell}_{w_{i,j}} + \hat{\alpha}_{\mathsf{R}(w_{i,j})} \hat{r}_{w_{i,j}}\,]^2_{[\ell+2,2\ell+i]}.
\end{aligned}$$

(Re-randomizing Key). The simulated key is not perfectly distributed yet since their randomness are still correlated. Consider every variable $\hat{y}$ in $\{\hat{r}\} \cup \{\hat{\alpha}_{w_{i,j}}\}_{w_{i,j} \in \mathsf{Nodes}} \cup \{\hat{v}_{w_{1,j}}\}_{w_{1,j} \in \mathsf{Inputs}} \cup \{\hat{\ell}_{w_{i,j}}, \hat{r}_{w_{i,j}}\}_{w_{i,j} \in \mathsf{Gates}}$. We re-randomize them by implicitly setting new randomness as $\hat{y}'' = \hat{y} + y'$, where $\mathcal{B}$ samples $[\,y'\,]_{\emptyset} \leftarrow \mathsf{Samp}(\mathsf{param})$. To be able to compute corresponding keys with updated randomness, it amounts to verify that the encoding of the term that $\hat{y}$ is multiplied to in the key can be computed. For example, $\hat{r}$ appears in $\hat{D}_1 = [\,\beta_k + \hat{\phi}_2 \hat{r}\,]^2_{[\ell+2,3\ell]}$. This can be re-randomized to $\hat{D}_1'' = [\,\beta_k + \hat{\phi}_2(\hat{r} + r')\,]^2_{[\ell+2,3\ell]}$ by computing $\hat{D}_1'' = \hat{D} + [\,r'\,]_{\emptyset} \cdot [\,\hat{\phi}_2\,]^2_{[\ell+2,3\ell]}$, which can be done since $[\,\hat{\phi}_2\,]^2_{[\ell+2,3\ell]}$ is computable (since we have $[\,c_1 z\,]^2_{[\ell+2,3\ell]}$ from the assumption). The other terms can be verified as follows. $[\,\hat{\phi}_1\,]^2_{[\ell+2,3\ell]} = [\,\phi_1'\,]^2_{[\ell+2,3\ell]} + [\,c_1\,]^2_{[\ell+2,3\ell]}$ is computable. For $j \notin A_{x^\star}$, $[\,\hat{h}_j\,]^2_{[\ell+2,2\ell+1]}$ can be computed from $[\,c_1\,]^2_{[\ell+2,2\ell+1]}$, while for $j \in A_{x^\star}$, it can be computed trivially. For $w_{i,j} \in \mathsf{Gates}$, note that $\mathsf{L}(w_{i,j})$, $\mathsf{R}(w_{i,j})$ have depth $i-1$, hence $[\,\hat{\alpha}_{\mathsf{L}(w_{i,j})}\,]^2_{[\ell+2,2\ell+i]} = [\,\hat{\alpha}_{\mathsf{R}(w_{i,j})}\,]^2_{[\ell+2,2\ell+i]} = [\,c_1 \cdots c_i\,]^2_{[\ell+2,2\ell+i]} = [\,c_1\,]^2_{\{2\ell+i\}}[\,c_2\,]^2_{[\ell+2,2\ell+1]}[\,c_3\,]^2_{\{2\ell+2\}} \cdots [\,c_i\,]^2_{\{2\ell+i-1\}}$ can be computed from D.

(Returning Key). $\mathcal{B}$ adds the above semi-functional components to the normal components and returns $\mathsf{SK}_k + \widehat{\mathsf{SK}}_k''$.

Guess. The algorithm $\mathcal{B}$ has properly simulated G_{q_1+1} if $\delta = 0$, and G_{q_1+2} if $\delta \in_R \mathcal{R}$. Hence, $\mathcal{B}$ can use the output of $\mathcal{A}$ to break the $\mathsf{EMDDH2}$ Assumption. □

B.6 Type-2 to Type-3 Semi-functional Keys in Phase 2

Lemma 17 (G_{q_1+2} to G_{q_1+3}). *For any adversary $\mathcal{A}$, there exists an algorithm $\mathcal{B}$ that breaks the Subgroup Decision Assumption 2 with* $|\mathsf{G}_{q_1+2}\mathsf{Adv}_{\mathcal{A}}^{(n,\ell)\text{-}\mathsf{KPABE}}(\lambda) - \mathsf{G}_{q_1+3}\mathsf{Adv}_{\mathcal{A}}^{(n,\ell)\text{-}\mathsf{KPABE}}(\lambda)| \leq \mathsf{Adv}_{\mathcal{B}}^{\mathsf{SD2}}(\lambda)$.

Proof. The proof is exactly the same as when we modify type-2 to type-3 semi-functional key in phase 1 (the proof of Lemma 14), except that this time we modify the post-challenge keys *all at once*, instead of one key at a time. In particular, the simulation of $\mathsf{PK}, \mathsf{MSK}$ and the challenge ciphertext is exactly the same. The simulation for type-3 semi-functional key queries in phase 1 is done using $[\,c\,]^{2,3}_{[\ell+2,3\ell]}$ as in Eq.(15). The simulation of every key in phase 2 can be done using the problem instance Z in exactly the same way as in Eq.(16) for all elements except D_1'. $\mathcal{B}$ simulates D_1' in a similar way as in Eq.(18) except that this time the same $\tilde{\beta}$ is used for every key. Namely at the beginning of phase 2, $\mathcal{B}$ samples $[\,\beta\,]_{\emptyset} \leftarrow \mathsf{Samp}(\mathsf{param})$. For the j-th key query, $\mathcal{B}$ computes

$$D_1'^{(j)} = [\,\alpha\,]^{1,2}_{[\ell+2,3\ell]} + [\,\tilde{\phi}_2\,]_{\emptyset} \cdot [\,\tilde{r}^{(j)}\,]_{\emptyset} \cdot Z_{[\ell+2,3\ell]} + [\,\tilde{\beta}\,]_{\emptyset} \cdot [\,c\,]^{2,3}_{[\ell+2,3\ell]}. \tag{19}$$

Again we remark that the randomness in a key, *e.g.,* ($[\,\tilde{r}^{(j)}\,]_{\emptyset}$, is sampled fresh for each key. We can see that the simulated keys are all type-2 or all type-3 with $\beta = \tilde{\beta}c \bmod N_2$. □

B.7 Final Game

Lemma 18 (G_{q_1+3} to $\mathsf{G}_{\mathrm{final}}$). *We have* $\mathsf{G}_{q_1+3}\mathsf{Adv}_{\mathcal{A}}^{(n,\ell)\text{-}\mathsf{KPABE}}(\lambda) = \mathsf{G}_{\mathrm{final}}\mathsf{Adv}_{\mathcal{A}}^{(n,\ell)\text{-}\mathsf{KPABE}}(\lambda)$.

Proof. We first claim that in G_{q_1+3}, $\alpha \bmod N_2$ is uniformly random in $\mathbb{Z}_{N_2}$ in the adversary $\mathcal{A}$'view. This is since all the appearance of $\alpha \bmod N_2$ in all the semi-functional keys (of type-3) is added by uniformly random values (β_j for the pre-challenge keys and β for the post-challenge keys). Hence, $\alpha \bmod N_2$ in C_0 in the challenge ciphertext is uniformly random to $\mathcal{A}$. From this claim and from the property of Ext, we have that the message mask $\mathsf{Ext}(\mathsf{param}, [\,\alpha t\,]^1_{[1,3\ell]} + [\,\alpha \hat{t}\,]^2_{[1,3\ell]})$ is uniformly random in $\{0,1\}^{\lambda}$ and hence $M_{\mathfrak{b}}$ is completely hidden. Therefore, we can modify to encrypt any random message. □

C Security Proof for CP-ABE

Lemma 19 ($\mathsf{G}_{k,1}$ to $\mathsf{G}_{k,2}$). *For any adversary $\mathcal{A}$, there exists an algorithm $\mathcal{B}$ that breaks the* ℓ-$\mathsf{EMDDH2\text{-}dual}$ *with advantage* $|\mathsf{G}_{k,1}\mathsf{Adv}^{(n,\ell)\text{-}\mathsf{CPABE}}_{\mathcal{A}}(\lambda) - \mathsf{G}_{k,2}\mathsf{Adv}^{(n,\ell)\text{-}\mathsf{CPABE}}_{\mathcal{A}}(\lambda)| \leq \mathsf{Adv}^{\ell\text{-}\mathsf{EMDDH2\text{-}dual}}_{\mathcal{B}}(\lambda)$, *where ℓ is the bounded depth.*

Proof. The proof follows almost the same way as for that of KP-ABE in phase 2 (Lemma 16), where we simulate ciphertext for string $x^\star$ before simulating key for circuits f. This time, we simulate key for string x before simulating ciphertext for circuit $f^\star$. We will only describe the difference here. The setup is exactly the same except the following two points. First, it computes $[\,\alpha\,]^{1,2}_{[1,\ell+1]}$ for MSK. (This is due to the difference in the scheme). Second, $\mathcal{B}$ additionally samples $[\,\psi_2\,]_\emptyset, [\,\psi_3\,]_\emptyset \leftarrow \mathsf{Samp}(\mathsf{param})$ and defines $\mathsf{PK}, \mathsf{MSK}$ accordingly.

Phase 1. When $\mathcal{A}$ makes the j-th key query for $x^{(j)}$, $\mathcal{B}$ simulates a type-3 semi-functional key if $j < k$ and a normal key if $j > k$ straightforwardly as usual. When $j = k$, $\mathcal{B}$ simulates SK_k for string $x^{(k)}$ in almost the same manner as the simulation of the ciphertext for string $x^\star$ in the KP-ABE proof (Lemma 16). It programs parameters $\hat{\phi}_1, \{\hat{h}_j\}_{j\in[1,n]}$ and key randomness $\hat{s}$ in exactly the same manner. (Note that we do not use ϕ_2 in CP-ABE). $\mathcal{B}$ further simulates the additional parameters for CP-ABE as follows. It samples $[\,\psi_2'\,]_\emptyset, [\,\psi_3'\,]_\emptyset \leftarrow \mathsf{Samp}(\mathsf{param})$ and implicitly defines

$$\hat{\psi}_2 = c_1 + \psi_2', \qquad \hat{\psi}_3 = c_1 \cdots c_{\ell+1} + \psi_3', \qquad \hat{\tau} = z, \qquad \beta_k = \delta.$$

where $\mathcal{B}$ will use $Z = [\,\delta + c_1 \cdots c_{\ell+1} z\,]^2_{[1,\ell+1]}$ from the problem instance to embed δ below. From these implicit definitions, $\mathcal{B}$ then computes $\hat{C}, \hat{C}_j$ exactly as in the KP-ABE proof. The additional elements are computed as: $\hat{Q}_1 = [\,z\,]^2_{[1,\ell+1]} = [\,\hat{\tau}\,]^2_{[1,\ell+1]}$, and

$$\hat{Q}_2 = [\,\psi_2'\,]_\emptyset \cdot [\,z\,]^2_{[1,\ell+1]} - [\,\phi_1'\,]_\emptyset \cdot [\,z\,]^2_{[1,\ell+1]}. = [\,\hat{\psi}_2\hat{\tau} + \hat{\phi}_1\hat{s}\,]^2_{[1,\ell+1]},$$
$$\hat{Q}_3 = Z + [\,\psi_3'\,]_\emptyset \cdot [\,z\,]^2_{[1,\ell+1]} = [\,\beta_k + \hat{\psi}_3\hat{\tau}\,]^2_{[1,\ell+1]}$$

where we notice that in $\hat{Q}_2$, a critical term $c_1 z$ from $\hat{\psi}_2\hat{\tau}$ and $\hat{\phi}_1\hat{s}$ is canceled out, and in $\hat{Q}_3$, $c_1 \cdots c_{\ell+1} z$ in Z combines with ψ_3' to yield $\hat{\psi}_3$. $\mathcal{B}$ then re-randomizes all randomness in key, say to $\hat{\tau} + \tau', \hat{s} + s'$. This can be done since for the additional elements to CP-ABE, $[\,\hat{\psi}_2\,]^2_{[1,\ell+1]}, [\,\hat{\psi}_3\,]^2_{[1,\ell+1]}$ are computable (since we have $[\,c_1\,]^2_{[1,\ell+1]}, [\,c_1 \cdots c_{\ell+1}\,]^2_{[1,\ell+1]}$ respectively).

Challenge. $\mathcal{B}$ simulates a challenge ciphertext for circuit $f^\star$ in almost the same way as the simulation of the key for circuit f_k in the KP-ABE proof (Lemma 16). It programs ciphertext randomness $\hat{r}, \{\hat{\alpha}_{w_{i,j}}\}_{w_{i,j}\in\mathsf{Nodes}}, \{\hat{v}_{w_{1,j}}\}_{w_{1,j}\in\mathsf{Inputs}}, \{\hat{\ell}_{w_{i,j}}, \hat{r}_{w_{i,j}}\}_{w_{i,j}\in\mathsf{Gates}}$ in exactly the same manner. Additionally for CP-ABE, $\mathcal{B}$ defines $\hat{u} = -1$ (this will be re-randomized later). From these implicit definitions, $\mathcal{B}$ computes $\{\hat{\mathcal{K}}_w\}_{w\in\mathsf{Nodes}}$ and $\hat{D}_2, \hat{D}_3$ as in Lemma 16. The remaining additional elements are computed as: $\hat{W}_0 = -[\,1\,]^2_{[\ell+2,3\ell]} = [\,\hat{u}\,]^1_{[\ell+2,3\ell]}$, and

$$\hat{W}_1 = -[\,\psi_3'\,]_\emptyset \cdot [\,1\,]^2_{[\ell+2,3\ell]} + [\,\psi_2'\,]_\emptyset \cdot [\,c_2 \cdots c_{\ell+1}\,]^2_{[\ell+2,3\ell]} = [\,\hat{\psi}_3\hat{u} + \hat{\psi}_2\hat{r}\,]^2_{[\ell+2,3\ell]},$$

where we recall that $\hat{r} = c_2 \cdots c_{\ell+1}$ and see that a critical term $c_1 \cdots c_{\ell+1}$ is canceled out in $\hat{W}_1$. $\mathcal{B}$ also computes the semi-functional component of message mask as $\hat{K} = [\,\alpha\,]^{1,2}_{[1,\ell+1]} \cdot \hat{W}_0 = [\,\alpha\hat{u}\,]^2_{[1,3\ell]}$. $\mathcal{B}$ re-randomizes the ciphertext by first sampling $[\,u'\,]_\emptyset \leftarrow \mathsf{Samp}(\mathsf{param})$. It also samples $[\,y'\,]_\emptyset \leftarrow \mathsf{Samp}(\mathsf{param})$ for every variable $\hat{y}$ in $\{\hat{r}\} \cup \{\hat{\alpha}_{w_{i,j}}\}_{w_{i,j}\in\mathsf{Nodes}} \cup \{\hat{v}_{w_{1,j}}\}_{w_{1,j}\in\mathsf{Inputs}} \cup \{\hat{\ell}_{w_{i,j}}, \hat{r}_{w_{i,j}}\}_{w_{i,j}\in\mathsf{Gates}}$. It then sets the new randomness as $\hat{u}'' = u' \cdot \hat{u} = -u'$, and $\hat{y}'' = u'\hat{y} + y'$ for every variable $\hat{y}$ listed above. The original elements can be re-randomized due to the same reason for re-randomizing key in Lemma 16. One difference is that this time we also have factor u' (multiplied to $\hat{y}$), but this is straightforward to deal with. (This is indeed analogous to re-randomizing *ciphertext* for KP-ABE in Lemma 16). For the additional elements to CP-ABE, we re-randomize as follows. $\mathcal{B}$ sets $\hat{W}_0'' = [\, u' \,]_\emptyset \cdot \hat{W}_0$, and $\hat{W}_1'' = [\, u' \,]_\emptyset \cdot \hat{W}_1 + [\, r' \,]_\emptyset \cdot [\, \hat{\psi}_2 \,]^2_{[\ell+2,3\ell]}$, where we note that $[\, \hat{\psi}_2 \,]^2_{[\ell+2,3\ell]}$ is computable (since we have $[\, c_1 \,]^2_{[\ell+2,3\ell]}$). $\mathcal{B}$ also computes $\hat{K}'' = \hat{K} \cdot [\, u' \,]_\emptyset = [\, \alpha \hat{u}'' \,]^2_{[1,3\ell]}$. $\mathcal{B}$ adds the normal part and the simulated semi-functional component of ciphertext and returns the challenge.

Guess. The algorithm $\mathcal{B}$ has properly simulated $\mathsf{G}_{k,1}$ if $\delta = 0$, and $\mathsf{G}_{k,2}$ if $\delta \in_R \mathcal{R}$. Hence, $\mathcal{B}$ can use the output of $\mathcal{A}$ to break the $\mathsf{EMDDH2\text{-}dual}$ Assumption. $\square$

Lemma 20 (G_{q_1+1} to G_{q_1+2}). *For any adversary $\mathcal{A}$ which issues the challenge query for a circuit $f^\star$ of which the maximum number of (internal) gates per layer is m, there exists an algorithm $\mathcal{B}$ that breaks the (ℓ, m)-$\mathsf{EMDDH1\text{-}dual}$ with $|\mathsf{G}_{q_1+1}\mathsf{Adv}_{\mathcal{A}}^{(n,\ell)\text{-}\mathsf{CPABE}}(\lambda) - \mathsf{G}_{q_1+2}\mathsf{Adv}_{\mathcal{A}}^{(n,\ell)\text{-}\mathsf{CPABE}}(\lambda)| \leq \mathsf{Adv}_{\mathcal{B}}^{(\ell,m)\text{-}\mathsf{EMDDH1\text{-}dual}}(\lambda)$.*

Proof. The proof follows almost the same way as for that of KP-ABE in phase 1 (Lemma 3), where we simulate key for circuit f before simulating ciphertext for string $x^\star$. This time, we simulate ciphertext for circuit $f^\star$ before simulating key for string x. We will only describe the difference here. The setup is exactly the same except the following two points. First, it computes $[\, \alpha \,]^{1,2}_{[1,\ell+1]}$ for MSK. (This is due to the difference in the scheme). Second, $\mathcal{B}$ additionally samples $[\, \psi_2 \,]_\emptyset, [\, \psi_3 \,]_\emptyset \leftarrow \mathsf{Samp}(\mathsf{param})$ and defines $\mathsf{PK}, \mathsf{MSK}$ accordingly. In phase 1, answering type-3 semi-functional keys can be done straightforwardly in an analogous manner to KP-ABE.

Challenge. $\mathcal{B}$ simulates a challenge ciphertext for circuit $f^\star$ in almost the same way as the simulation of the key for circuit f_k in KP-ABE (in Lemma 3). It programs parameters $\hat{\phi}_1, \{\hat{h}_j\}_{j\in[1,n]}$ and ciphertext randomness $\hat{r}, \{\hat{\alpha}_{w_{i,j}}\}_{w_{i,j}\in\mathsf{Nodes}}, \{\hat{v}_{w_{1,j}}\}_{w_{1,j}\in\mathsf{Inputs}}, \{\hat{\ell}_{w_{i,j}}, \hat{r}_{w_{i,j}}\}_{w_{i,j}\in\mathsf{Gates}}$ in exactly the same manner. (Note that we do not use ϕ_2 in CP-ABE). $\mathcal{B}$ further simulates the additional parameters for CP-ABE as follows. It samples $[\, \psi_2' \,]_\emptyset, [\, \psi_3' \,]_\emptyset \leftarrow \mathsf{Samp}(\mathsf{param})$ and implicitly defines

$$\hat{\psi}_2 = v + \psi_2', \qquad \hat{\psi}_3 = -b + \psi_3', \qquad \hat{u} = \frac{c_1 \cdots c_{\ell+1}}{b}.$$

From these implicit definitions, $\mathcal{B}$ computes $\{\hat{\mathcal{K}}_w\}_{w\in\mathsf{Nodes}}$ and $\hat{D}_2, \hat{D}_3$ as in Lemma 3. The remaining additional elements are computed as:

$$\hat{W}_0 = [\, \frac{c_1 \cdots c_{\ell+1}}{b} \,]^2_{[\ell+2,3\ell]}, \qquad \hat{W}_1 = [\, \psi_3' \,]_\emptyset \cdot [\, \frac{c_1 \cdots c_{\ell+1}}{b} \,]^2_{[\ell+2,3\ell]} + [\, \psi_2' \,]_\emptyset \cdot [\, \frac{c_1 \cdots c_{\ell+1}}{v} \,]^2_{[\ell+2,3\ell]}$$

where $c_1 \cdots c_{\ell+1}$ is canceled out in $\hat{W}_1$. $\mathcal{B}$ also computes the semi-functional component of message mask as $\hat{K} = [\, \alpha \,]^{1,2}_{[1,\ell+1]} \cdot \hat{W}_0 = [\, \alpha \hat{u} \,]^1_{[1,3\ell]}$. $\mathcal{B}$ then re-randomizes the ciphertext as usual, to have all randomness to be independent from $\hat{u}$. The additional component $\hat{W}_1$ can be re-randomized since $[\, \hat{\psi}_2 \,]^2_{[\ell+2,3\ell]}$ can be computed (since we have $[\, v \,]^2_{[\ell+2,3\ell]}$). $\mathcal{B}$ adds the normal part and the simulated semi-functional component of ciphertext and returns the challenge.

Phase 2. $\mathcal{B}$ implicitly sets $\beta = \delta$. $\mathcal{B}$ will use $Z = [\, \delta + bz \,]^2_{[1,\ell+1]}$ from the problem instance to embed it to each key. When $\mathcal{A}$ makes the j-th key query for $x^{(j)}$, $\mathcal{B}$ generates a type-1 or type-2 semi-functional key as follows. First it generates a normal key $\mathsf{SK}_j \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, x^{(j)})$. $\mathcal{B}$ then defines semi-functional components as follows. It implicitly defines $\hat{s}$ as in the ciphertext in the KP-ABE proof. Additionally for CP-ABE, $\mathcal{B}$ defines $\hat{\tau} = -z$. $\mathcal{B}$ then computes $\hat{C}, \hat{C}_j$ exactly as in the KP-ABE proof. The additional elements are computed as follows. First, $\hat{Q}_1 = -[\, z \,]^2_{[1,\ell+1]}$. To compute $\hat{Q}_2$, we substitute terms and see $\hat{\psi}_2 \hat{\tau} + \hat{\phi}_1 \hat{s} = (v + \psi_2')(-z) + (v + \phi_1') z \prod_{i\in[2,\ell]} \left(1 + \sum_{w_{i,k}\in S_i} \frac{1}{a_{i,k}}\right)$.

A critical term zv is canceled out and $\hat{Q}_2$ can be computed almost exactly as $\hat{D}_2$ in the KP-ABE proof. $\mathcal{B}$ then computes

$$\hat{Q}_3 = Z - [\,\psi_3'\,]_{\emptyset} \cdot [\,z\,]^2_{[1,\ell+1]} = [\,\beta + \hat{\psi}_3 \hat{\tau}\,]^2_{[1,\ell+1]}.$$

where we recall $Z = [\,\beta + bz\,]^2_{[1,\ell+1]}$ and $\hat{\psi}_3 = -b + \psi_3'$. $\mathcal{B}$ then re-randomizes all randomness to say, $\hat{s} + s', \hat{\tau} + \tau'$. The additional elements can be re-randomized since $[\,\hat{\psi}_2\,]^2_{[1,\ell+1]}, [\,\hat{\psi}_3\,]^2_{[1,\ell+1]}$ are computable (since we have $[\,v\,]^2_{[1,\ell+1]}, [\,b\,]^2_{[1,\ell+1]}$ respectively).

Guess. The algorithm $\mathcal{B}$ has properly simulated G_{q_1+1} if $\delta = 0$, and G_{q_1+2} if $\delta \in_R \mathcal{R}$. Hence, $\mathcal{B}$ can use the output of $\mathcal{A}$ to break the EMDDH1-dual Assumption. $\square$

D Omitted Discussions

Why Traditional Dual System Would Fail for GGHSW. (Continued from §1.1). We examine our canonical scheme, the KP-ABE of GGHSW. We use a toy circuit f^3 in Figure 1. This toy circuit seems to be a suitable representative for general circuits since it contains fan-out 2 (albeit from the input). We consider the following variant of their scheme. We use asymmetric graded 3-linear maps $\mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{G}_3 \to \mathbb{G}_T$. Let g_1, g_2, g_3 be a generator in each group respectively. Denote $g_{12} = e(g_1, g_2), g_{23} = e(g_2, g_3)$, $g_{123} = e(g_1, g_2, g_3)$. Our toy scheme has a public key $\mathsf{PK} = (g_1^{h_1}, g_1^{h_2}, g_1^{h_3}, g_{123}^{\alpha})$, and a master key $\mathsf{MSK} = \alpha$. A key for f^3 is $\mathsf{SK}_{f^3} = (K_{\mathsf{a}}, K_{\mathsf{b}}, K_{\mathsf{c}})$ where $K_{\mathsf{a}} = (g_2^{\alpha_{\mathsf{a}} + h_1\ell_{\mathsf{a}} + h_2 r_{\mathsf{a}}}, g_2^{\ell_{\mathsf{a}}}, g_2^{r_{\mathsf{a}}})$, $K_{\mathsf{b}} = (g_2^{\alpha_{\mathsf{b}} + h_2\ell_{\mathsf{b}}}, g_2^{\alpha_{\mathsf{b}} + h_3 r_{\mathsf{b}}}, g_2^{\ell_{\mathsf{b}}}, g_2^{r_{\mathsf{b}}})$, and $K_{\mathsf{c}} = (g_{23}^{\alpha + \alpha_{\mathsf{a}}\ell_{\mathsf{c}} + \alpha_{\mathsf{b}} r_{\mathsf{c}}}, g_3^{\ell_{\mathsf{c}}}, g_3^{r_{\mathsf{c}}})$. Note that α_w, ℓ_w, r_w for gate w are randomness dedicated to only this key. A ciphertext encrypting a message M for $x = 101$ are $\mathsf{CT}_{101} = (g_{123}^{s\alpha} M, g_1^s, g_1^{h_1 s}, g_1^{h_3 s})$.

The underlying pair encoding of the above scheme is indeed the set of all polynomials in the exponents that define a key and a ciphertext (the one with M is the only exception that we do not include). We refer to the details in [1]. We note that, although only the case of bilinear maps was provided in [1], we expect the framework to function also in multi-linear map settings. The encoding function for x in this scheme is a polynomial set $(s, h_1 s, h_3 s)$. The case for the encoding of f is similar. We claim that this pair encoding is *not* information-theoretically secure, meaning that α is exposed given encodings of x, f such that $f(x) = 0$. Since $f^3(101) = 0$, we can give out the encoding functions of SK_{f^3} and CT_{101}. We can see that α is exposed as follows. First, we have h_1, h_3 from CT_{101}. From h_3 and K_{b}, we obtain α_{b} and then h_2. From h_1, h_2 and K_{a}, we obtain α_{a}. From $\alpha_{\mathsf{a}}, \alpha_{\mathsf{b}}$ and K_{c}, we obtain α. This concludes our claim. We note that the very nature of multi-fan-out allows this "backtracking" attack: h_2, which is not expected to be exposed since $x_2 = 0$, is exposed from h_3 via their parent OR gate. Since the multi-fan-out feature is essential for general circuits, we could not hope much to use the information-theoretic approach.

Decryption for Toy Examples. (Continued from §2). For the first toy example, to decrypt CT_{10} by $\mathsf{SK}_{\mathsf{OR}}$, one computes $e(g_1^s, g_2^{\alpha + h_1\ell}) e(g_1^{h_1 s}, g_2^{\ell})^{-1} = e(g_1, g_2)^{s\alpha}$. To decrypt CT_{11} by $\mathsf{SK}_{\mathsf{AND}}$, one computes $e(g_1^s, g_2^{\alpha + h_1\ell + h_2 r}) e(g_1^{h_1 s}, g_2^{\ell})^{-1} e(g_1^{h_2 s}, g_2^{r})^{-1} = e(g_1, g_2)^{s\alpha}$. For the second toy example, as an example, to decrypt CT_{1110} by SK_f, we compute from input gates a, b to the output gate c as follows. First, we compute $e(g_1^s, g_2^{\alpha_{\mathsf{a}} + h_1\ell_{\mathsf{a}} + h_2 r_{\mathsf{a}}}) e(g_1^{h_1 s}, g_2^{\ell_{\mathsf{a}}})^{-1} e(g_1^{h_2 s}, g_2^{r_{\mathsf{a}}})^{-1} = g_{12}^{\alpha_{\mathsf{a}} s}$, and $e(g_1^s, g_2^{\alpha_{\mathsf{b}} + h_3\ell_{\mathsf{b}}}) e(g_1^{h_3 s}, g_2^{\ell_{\mathsf{b}}})^{-1} = g_{12}^{\alpha_{\mathsf{b}} s}$. Then, we compute $e(g_1^s, g_{23}^{\alpha + \alpha_{\mathsf{a}}\ell_{\mathsf{c}} + \alpha_{\mathsf{b}} r_{\mathsf{c}}}) e(g_{12}^{\alpha_{\mathsf{a}} s}, g_3^{\ell_{\mathsf{c}}})^{-1} e(g_{12}^{\alpha_{\mathsf{b}} s}, g_3^{r_{\mathsf{c}}})^{-1}$ $= g_{123}^{s\alpha}$.

One More Technical Intuition. (Continued from §2). When generalizing to circuits, we also have one more technique for simulation. This is continued from the end of §2. As an attempt for the simulation of s in the ciphertext when generalizing to larger circuits, we can set $s =$

$z(1+\frac{1}{a_{i_1}})\cdots(1+\frac{1}{a_{i_k}})$, where each $(1+\frac{1}{a_i})$ would "select " some corresponding gate in chains defined for h_j to enable the canceling in $g_1^{sh_j}$. To be able to compute such s, again we must decompose the multiplication. However, this time, the number of multiplication can be as large as the number of all gates in the circuit, which would then define multi-linearity required to be as many. This would not be desirable. We resolve this by observing that in each path, there is only one corresponding gate of each depth. Hence, gathering all selectors in corresponding the same depth as a sum would not cause any problem. That is, we have $s = z(1+\frac{1}{a_{i_{11}}}+\cdots+\frac{1}{a_{i_{1t}}})\cdots(1+\frac{1}{a_{i_{\ell 1}}}+\cdots+\frac{1}{a_{i_{\ell t}}})$, where $a_{i_{ju}}$ is corresponding to gates with the same depth j. This allows us to use only ℓ multi-linearity. In the simulation, this translates to Eq. (13). (If there were two selectors in the sum that trigger canceling in the same chain, an unexpected canceling might occur that leads to a critical term $g_1^{zc_1}$, but we do not elaborate here).

Contents