Attribute Based Encryption for Turing Machines from Lattices

Shweta Agrawal*, Simran Kumari*, Shota Yamada†

*IIT Madras, India
shweta@cse.iitm.ac.in, sim78608@gmail.com
†AIST Tokyo, Japan
yamada-shota@aist.go.jp

Abstract

We provide the first attribute based encryption (ABE) scheme for Turing machines supporting unbounded collusions from lattice assumptions. In more detail, the encryptor encodes an attribute $\mathbf{x}$ together with a bound t on the machine running time and a message m into the ciphertext, the key generator embeds a Turing machine M into the secret key and decryption returns m if and only if $M(\mathbf{x}) = 1$. Crucially, the input $\mathbf{x}$ and machine M can be of unbounded size, the time bound t can be chosen dynamically for each input and decryption runs in input specific time.

Previously the best known ABE for uniform computation supported only non-deterministic log space Turing machines (NL) from pairings (Lin and Luo, Eurocrypt 2020). In the post-quantum regime, the state of the art supports non-deterministic finite automata from LWE in the *symmetric* key setting (Agrawal, Maitra and Yamada, Crypto 2019).

In more detail, our results are:

1. We construct the first ABE for NL from the LWE, evasive LWE (Wee, Eurocrypt 2022 and Tsabary, Crypto 2022) and Tensor LWE (Wee, Eurocrypt 2022) assumptions. This yields the first (conjectured) post-quantum ABE for NL.
2. Relying on LWE, evasive LWE and a new assumption called *circular tensor* LWE, we construct ABE for all Turing machines. At a high level, the circular tensor LWE assumption incorporates circularity into the tensor LWE (Wee, Eurocrypt 2022) assumption.

Towards our ABE for Turing machines, we obtain the first CP-ABE for circuits of unbounded depth and size from the same assumptions – this may be of independent interest.

Contents

1 Introduction

Attribute based encryption (ABE) [SW05, GPSW06] is a fundamental primitive in cryptography that enables fine-grained access control on encrypted data. In an ABE scheme, the ciphertext encodes a secret message m and a public *attribute* $\mathbf{x}$, the secret key encodes a (public) function f, and decryption outputs m if and only if $f(\mathbf{x}) = 1$. Security posits that an adversary cannot distinguish between an encryption of $(m_0, \mathbf{x})$ and $(m_1, \mathbf{x})$ given secret keys that do not decrypt the challenge. ABE has two variants – "key-policy" where the function f (typically representing an access control policy) is embedded in the key, or "ciphertext-policy" where it is embedded in the ciphertext, as the names suggest. These are denoted by KP-ABE and CP-ABE respectively.

Traditionally, the function f is represented by a circuit and while there has been fantastic progress in building ABE supporting general circuits [GPSW06, GVW13, BGG+14, AY20a, Wee22, HLL23], there are inherent limitations to the circuit model. Circuits force the size of the input to be fixed ahead of time and also incur *worst case* running time on every input – this is dissatisfying from both the theoretical and practical perspective. To overcome these limitations, a line of works [Wat12, GKP+13, AS16, AM18, KNTY19, GWW19, GW20, AMY19a, AMY19b, LL20a] has studied ABE for uniform models of computation but success has been much more limited. Without relying on heavy hammers such as indistinguishability obfuscation or compact functional encryption, the state of art ABE in this regime[1] supports non-deterministic log space Turing machines from pairings [LL20a]. In the post-quantum regime, the situation is even less satisfactory – the state of the art construction supports non-deterministic finite state automata from learning with errors (LWE) but only in the *symmetric* key setting [AMY19a]. In the public key setting, even ABE for DFA – supporting unbounded lengths and with provable security – is not known, to the best of our knowledge. Thus, an outstanding open question is:

Can we extend ABE for uniform computation beyond NL?

1.1 Our Results

In this work, we take a leap forward and provide the first ABE scheme for Turing machines supporting unbounded collusions from lattice assumptions. In more detail, the encryptor encodes an attribute $\mathbf{x}$ together with a bound t on the machine running time and a message m into the ciphertext, the key generator embeds a Turing machine M into the secret key and decryption returns m if and only if $M(\mathbf{x}) = 1$. Crucially, the input $\mathbf{x}$ and machine M can be of unbounded size, the time bound t can be chosen dynamically for each input and decryption runs in input specific time.

In more detail, our results are:

1. We construct the first ABE for NL from the LWE, evasive LWE [Wee22, Tsa22] and Tensor LWE [Wee22] assumptions. This yields the first (conjectured) post-quantum ABE for NL.
2. Relying on LWE, evasive LWE and a new assumption called *circular tensor* LWE, we construct ABE for all Turing machines. At a high level, the circular tensor LWE assumption incorporates circularity into the tensor LWE (Wee, Eurocrypt 2022) assumption.

Towards our ABE for Turing machines, we obtain the first CP-ABE for circuits of unbounded depth and size from the same assumptions – this may be of independent interest.

Our Assumptions. Below, we describe the evasive and circular tensor LWE assumptions. Below, we adopt the convention by Wee [Wee22] and let the underline denote that noise is added to the term, whose exact value is not important.

Evasive LWE. The evasive LWE assumption [Wee22] states that if

$$\text{if } \mathbf{B}, \mathbf{A}, \mathbf{P}, \underline{\mathbf{s}^\top \mathbf{B}}, \underline{\mathbf{s}^\top \mathbf{A}}, \underline{\mathbf{s}^\top \mathbf{P}}, \mathsf{aux} \approx \mathbf{B}, \mathbf{A}, \mathbf{P}, \$, \$, \$, \mathsf{aux}$$

$$\text{then } \mathbf{B}, \mathbf{A}, \mathbf{P}, \underline{\mathbf{s}^\top \mathbf{B}}, \underline{\mathbf{s}^\top \mathbf{A}}, \mathbf{B}^{-1}(\mathbf{P}), \mathsf{aux} \approx \mathbf{B}, \mathbf{A}, \mathbf{P}, \$, \$, \mathbf{B}^{-1}(\mathbf{P}), \mathsf{aux}$$

[1] In this work, we only consider ABE with unbounded collusion resistance.

Above $\mathbf{B}^{-1}(\mathbf{P})$ is a low norm Gaussian matrix such that $\mathbf{B}\ \mathbf{B}^{-1}(\mathbf{P}) = \mathbf{P}$, and $\mathbf{A}, \mathbf{P}$ are sampled in a correlated way. In the public coin version of this assumption, the auxiliary information aux above contains random coins using during the sampling. We rely on the private coin version of this assumption, similarly to prior work [ARYY23, VWW22].

Tensor Circular LWE. Similar to how circularity was incorporated into the evasive LWE assumption by [HLL23], we will need to incorporate it in the tensor LWE assumption introduced by Wee [Wee22]. The tensor LWE assumption states that for all $\mathbf{x}_1, \cdots, \mathbf{x}_Q \in \{0,1\}^\ell$, we have

$$①: \ \mathbf{A}, \underline{\left\{(\mathbf{s}(\mathbf{I}\otimes\mathbf{r}_i))^\mathsf{T}(\mathbf{A} - \mathbf{x}_i\otimes\mathbf{G}), \mathbf{r}_i\right\}_{i\in[Q]}} \approx_c \ ②: \ \mathbf{A}, \{\$, \mathbf{r}_i\}_{i\in[Q]}$$

In the original formulation $\mathbf{s}$ is uniform and $\mathbf{r}$ is sampled from a discrete Gaussian. We will require $\mathbf{s}$ to be small, i.e. also sampled from a discrete Gaussian.

Our *tensor circular* LWE *assumption* basically incorporates the circular terms into the assumption. For notational brevity, we denote $\mathbf{s}(\mathbf{I}\otimes\mathbf{r}_i)$ by $\mathbf{s}_{\mathbf{r}_i}$. In more detail, for all $\mathbf{x}_1, \ldots, \mathbf{x}_Q \in \{0,1\}^L$,

$$\mathbf{A}_{\mathsf{circ}}, \left\{①, \ \mathbf{A}_{\mathbf{s}_{\mathbf{r}_i}}, \ \mathbf{S}_{\mathbf{s}_{\mathbf{r}_i}}, \ \underline{\mathbf{s}_{\mathbf{r}_i}^\mathsf{T}(\mathbf{A}_{\mathsf{circ}} - \mathbf{S}_{\mathbf{s}_{\mathbf{r}_i}})\otimes\mathbf{G})}, \ \mathbf{r}_i\right\}_{i\in[Q]} \approx_c \mathbf{A}_{\mathsf{circ}}, \left\{②, \ \$, \ \$, \ \$, \ \mathbf{r}_i\right\}_{i\in[Q]}$$

Above $\mathbf{A}_{\mathbf{s}_{\mathbf{r}_i}}$ is the FHE public key corresponding to secret $\mathbf{s}_{\mathbf{r}_i}$, and $\mathbf{S}_{\mathbf{s}_{\mathbf{r}_i}} = \mathsf{hct}_{\mathbf{s}_{\mathbf{r}_i}}(\mathbf{s}_{\mathbf{r}_i})$ is an FHE ciphertext.

The rationale of security for tensor LWE does not change with the addition of the circular terms, to the best of our knowledge. While we do not see any attack on this strengthening of tensor LWE, we note that the usage of tensor LWE in our construction is inherited from the construction by Wee [Wee22]. Any improvements to Wee's construction are likely to lead to improvements in our setting as well.

1.2 Technical Overview

Next, we outline the main technical ideas developed in our work.

Using KP-ABE and CP-ABE for circuits to build ABE for TM. The starting point of our work is the compiler by Agrawal et al. [AMVY21] that shows how to construct public key functional encryption (FE) for Turing machines using two circuit FE schemes in tandem, one ciphertext-policy and one key-policy. In more detail, their construction, building upon techniques developed in [AMY19a], relies on two restricted FE schemes: one that supports decryption in the case $|(\mathbf{x}, 1^t)| > |M|$ and one that supports the case where $|(\mathbf{x}, 1^t)| \leq |M|$. Here, $\mathbf{x}$ is the input chosen by the encryptor and is of unbounded length, M is the Turing machine chosen by the key generator and t is an upper bound on the runtime of the Turing machine on a given input $\mathbf{x}$. We note that t can be chosen by the encryptor dynamically for each x and is not a-priori bounded. These restricted schemes are then run in parallel so that the first scheme can be used to decrypt the ciphertext when $|(\mathbf{x}, 1^t)| > |M|$ and the second scheme can be used otherwise.

It was shown by [AMVY21] that the first sub-scheme can be instantiated using a CP-FE that supports unbounded size and unbounded depth circuits while the second can be instantiated using a KP-FE for unbounded size but bounded depth circuits. If the first scheme only supports bounded depth (but still supports unbounded size), this still yields an FE for a smaller class of computation NL. While the techniques of [AMVY21] were developed for the setting of bounded key FE, they can be adapted to the setting of unbounded key ABE as well. We skip the details here and refer the reader to Section 5 for the detailed compiler. Since KP-ABE for unbounded size but bounded depth circuits has been known since 2013 [GVW13, BGG+14], it remains to find a CP-ABE for unbounded size, unbounded depth circuits to instantiate the compiler.

CP-ABE for Unbounded Size Circuits. Traditionally ciphertext-policy schemes have been much harder to construct than their key-policy cousins – for instance, while KP-ABE for circuits has been known for over a decade from the standard LWE assumption, it was only very recently that the first CP-ABE for circuits was discovered [Wee22], and that too by relying on new lattice assumptions called the *evasive* and *tensor* LWE assumptions. Moreover, of these, the evasive LWE is actually a fairly strong, non-falsifiable assumption. Yet, these assumptions have generated significant excitement in the community since they enabled progress on problems that had remained "stuck" for a while.

Wee's construction supports circuits of unbounded size but only bounded depth. While this will turn out to be a significant barrier, this construction is all that we have to begin with. Let us recap this construction below and then try to adapt it to suit our needs.

Overview of Wee's CP-ABE. Wee's construction can be seen as broadly following a two step outline: (i) start with a CP-ABE which satisfies only single key security, (ii) add collusion resistance via randomization of keys. Indeed, this two-step recipe for CP-ABE has been used before [AY20b, BV22], where the first step is common to all works (barring minor syntactic differences) and can be based on LWE, while the second step has been implemented using pairings in [AY20b] and using heuristic lattice methods in [BV22]. Wee's construction achieves randomization of keys via *tensors*.

In more detail, from prior work on lattice based KP-ABE [BGG+14], it is known that given an input $\mathbf{x} \in \{0,1\}^\ell$, and a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times \ell m}$, one can homomorphically evaluate a circuit $f : \{0,1\}^\ell \to \{0,1\}$ on an "input encoding" matrix of form $\mathbf{A} - \mathbf{x} \otimes \mathbf{G}$ by multiplying on the right by a low norm matrix $\mathbf{H}_{\mathbf{A},f,\mathbf{x}}$ to obtain the term $\mathbf{A}_f - f(x)\mathbf{G}$. Here, $\mathbf{G}$ is a special gadget matrix defined as follows. Let $\mathbf{g} = [1, 2, 2^2, \ldots, 2^{\log q}]^\intercal$ and $\mathbf{G} = \mathbf{I} \otimes \mathbf{g}^\intercal$.

Towards CP-ABE, let the public key be the matrix $\mathbf{A}$ and the ciphertext for function f and message μ be of the form $\underline{\mathbf{s}^\intercal \mathbf{A}_f + \mu \lceil q/2 \rceil}$ for some randomly sampled LWE secret $\mathbf{s}$. To decrypt correctly, we would ideally want the secret key encoding attribute $\mathbf{x}$ to be $\underline{\mathbf{s}^\intercal(\mathbf{A} - \mathbf{x} \otimes \mathbf{G})}$ – this would let us recover the mask $\mathbf{s}^\intercal \mathbf{A}_f$ via homomorphic computation and subtract it from the ciphertext to recover the message (after rounding out a suitably bounded noise).

Of course, the key generator cannot know the encryption randomness $\mathbf{s}$, and indeed this randomness will change for every ciphertext (among an unbounded number) so the above does not work [2]. At this stage, to enable key generation, the evasive LWE assumption described above comes to our rescue. Loosely speaking, the evasive LWE assumption can be seen as a kind of "secret sharing" mechanism to generate our desired term $\underline{\mathbf{s}^\intercal(\mathbf{A} - \mathbf{x} \otimes \mathbf{G})}$ when the encryptor holds randomness $\mathbf{s}$ and the key generator holds attribute $\mathbf{x}$. The encryptor will now additionally provide an extra LWE sample $\underline{\mathbf{s}^\intercal \mathbf{B}}$ under a public matrix $\mathbf{B}$ and the key generator will provide a low norm Gaussian matrix $\mathbf{R}$ such that $\mathbf{B}\,\mathbf{R} = (\mathbf{A} - \mathbf{x} \otimes \mathbf{G})$. For readability, $\mathbf{R}$ is referred to as $\mathbf{B}^{-1}(\mathbf{A} - \mathbf{x} \otimes \mathbf{G})$ in the literature. Now, the decryptor can compute:

$$(\underline{\mathbf{s}^\intercal \mathbf{B}})\mathbf{B}^{-1}(\mathbf{A} - \mathbf{x} \otimes \mathbf{G}) = \underline{\mathbf{s}^\intercal(\mathbf{A} - \mathbf{x} \otimes \mathbf{G})}$$

and decryption can proceed as desired. However, while we achieve correctness, the above scheme is not secure. It is easy to see that an attacker, given keys for any $\mathbf{x}$ and $\bar{\mathbf{x}}$ can trivially break security. To achieve collusion resistance, it is necessary to randomize keys so that mix and match attacks are thwarted.

Randomizing keys. Wee [Wee22] observed that the correctness of the evaluation algorithm is preserved even under tensoring with random Gaussian vectors $\mathbf{r}$ – in more detail, one can evaluate f even on the randomized $(\mathbf{A} - \mathbf{x} \otimes \mathbf{G}) \otimes \mathbf{r}$ to recover $(\mathbf{A}_f - f(x)\mathbf{G}) \otimes \mathbf{r}$ by simply augmenting the low norm multiplication matrix to $\mathbf{H}_{\mathbf{A},f,\mathbf{x}} \otimes \mathbf{I}$. This is just a direct application of the mixed product property on tensors (see Section 2.6 for a refresher).

Armed with this technique, a modification of the above construction that is not immediately broken is:

$$\mathsf{ct}_f = \underline{\mathbf{s}^\intercal(\mathbf{A}_f \otimes \mathbf{I})} + \mu \cdot \mathbf{g},\ \ \underline{\mathbf{s}^\intercal \mathbf{B}},$$
$$\mathsf{sk}_\mathbf{x} = \mathbf{B}^{-1}((\mathbf{A} - \mathbf{x} \otimes \mathbf{G}) \otimes \mathbf{r}),\ \mathbf{r}$$

To decrypt, we compute

$$(\underline{\mathbf{s}^\intercal(\mathbf{A}_f \otimes \mathbf{I})} + \mu \cdot \mathbf{g})(\mathbf{I} \otimes \mathbf{r}) \approx \mathbf{s}^\intercal(\mathbf{A}_f \otimes \mathbf{r}) + \mu \mathbf{g} \cdot (\mathbf{I} \otimes \mathbf{r})$$
$$\underline{\mathbf{s}^\intercal \mathbf{B}}(\mathbf{B}^{-1}((\mathbf{A} - \mathbf{x} \otimes \mathbf{G}) \otimes \mathbf{r})) \cdot (\mathbf{H}_{\mathbf{A},f,\mathbf{x}} \otimes \mathbf{I}) \approx \mathbf{s}^\intercal(\mathbf{A}_f \otimes \mathbf{r})$$

and subtract the second from the first to recover μ. Now, for different keys, the random vectors $\mathbf{r}_i$ are different and mix and match attacks no longer apply.

The above scheme needs to be refined further to admit a proof, but we skip the details here for ease of exposition. We only remark that to support arbitrary (bounded) depth, Wee needs to make an additional assumption called *tensor* LWE as discussed above.

[2] A simple twist to this scheme yields a construction satisfying single-key security though by [SS10]

Overcoming bounded depth. Wee's construction only supports circuits of bounded depth, primarily because the evaluation algorithms used on the ciphertext and public key only support bounded depth. In this context, the recent exciting work by Hsieh et al. [HLL23] provides hope since they adapted the public key and ciphertext evaluation algorithms to support circuits of unbounded depth, and provided the first KP-ABE for circuits from a generalization of the evasive LWE assumption, called the *circular small secret evasive* LWE assumption.

One may hope that plugging in the algorithms of [HLL23] (henceforth HLL) into the CP-ABE construction of [Wee22] can give a CP-ABE of unbounded depth, but as we shall see this approach runs into multiple difficulties. While this approach is what we will finally make work, it is via several new ideas, new assumptions and an involved security proof. We proceed to outline these next.

Unbounded Evaluation by HLL. The work of HLL supports unbounded homomorphism via two steps: (i) noise removal and (ii) bootstrapping. Suppose we evaluate on the lattice encoding $\underline{\mathbf{s}^\mathsf{T}(\mathbf{A} - \mathbf{x} \otimes \mathbf{G})}$ as described previously and obtain $\mathbf{s}^\mathsf{T}(\mathbf{A}_f - f(\mathbf{x})\mathbf{G}) + \mathbf{e}_{\mathsf{large}}$ where $\mathbf{e}_{\mathsf{large}}$ is large noise, as the name suggests. We would like to reduce this noise, so that we can continue homomorphic evaluation. HLL provide a noise removal procedure inspired by the modulus reduction technique developed in the context of levelled FHE [BV11, BGV14], which allows to perform public operations and recover a *noiseless* encoding of the desired function $f(\mathbf{x})$. Unfortunately, the recovered encoding does not have the right structure and disallows further homomorphic computation. In more detail, their noise removal procedure allows to obtain a term

$$\mathsf{RndPad}_{\mathbf{A}_f}(\mathbf{s}) - f(\mathbf{x})\ \mathbf{s}^\mathsf{T}\mathbf{G}$$

where the second summand is what we want but the first is an ill-formed mask that is not amenable to homomorphism. The exact details of what this term looks like and how one obtains it are not too important here, and we will defer these to the technical sections. However, having reached this point, the obvious question is whether and how one can proceed to obtain a well formed encoding of $f(\mathbf{x})$ which has noise within the desired bound and permits further computation.

Bootstrapping or Structure Restoration. HLL provide a clever way out of the predicament by using circular security of LWE based FHE. At a high level, circular security posits that an FHE ciphertext remains pseudorandom even if it encrypts its own secret key. HLL first observe that if there was a way to compute $\underline{\mathbf{s}^\mathsf{T}\mathbf{A}'_f - \mathsf{RndPad}_{\mathbf{A}_f}(\mathbf{s})}$ then this could be combined with the term obtained above to recover $\underline{(\mathbf{s}^\mathsf{T}\mathbf{A}'_f - f(\mathbf{x})\ \mathbf{s}^\mathsf{T}\mathbf{G})}$ which has the right shape and allows to continue homomorphic evaluation. Thus, it suffices to compute the term $\underline{\mathbf{s}^\mathsf{T}\mathbf{A}'_f - \mathsf{RndPad}_{\mathbf{A}_f}(\mathbf{s})}$.

To do so, they leverage the algebraic structure of the GSW FHE scheme [GSW13] together with an elegant "automatic decryption" technique by Brakerski et al. [BTVW17]. In more detail, recall that in the GSW FHE scheme, the secret key is $\mathbf{s}^\mathsf{T}$, a ciphertext for message $\mathbf{y}^\mathsf{T}$ is a matrix $\mathbf{C}$ and decryption computes $\mathbf{s}^\mathsf{T}\mathbf{C}$ to recover $\underline{\mathbf{y}^\mathsf{T}}$. The work of [BTVW17] proposes to use the same secret $\mathbf{s}$ for encoding the attribute, and set the attribute to be an FHE encryption under $\mathbf{s}$. As we will see, this causes "automatic decryption" which will be useful to recover the term we are after. In more detail, HLL additionally provide the following elements in their ciphertext:

$$\mathbf{S} = \mathsf{hct}_{\mathbf{s}}(\mathbf{s}), \quad \underline{\mathbf{s}^\mathsf{T}(\mathbf{A}_{\mathsf{circ}} - \mathbf{S} \otimes \mathbf{G})}$$

Given this, one can evaluate the circuit $\mathsf{RndPad}_f(\cdot)$ homomorphically on the second term to obtain $\underline{\mathbf{s}^\mathsf{T}\mathbf{A}'_f} - \mathbf{s}^\mathsf{T}\mathsf{hct}(\mathsf{RndPad}_{\mathbf{A}_f}(\mathbf{s}))$ for some matrix $\mathbf{A}'_f$. Next, given the structure of GSW ciphertext, $\mathbf{s}^\mathsf{T}\mathsf{hct}(\underline{\mathsf{RndPad}_{\mathbf{A}_f}(\mathbf{s})})$ decrypts *automatically* to yield $\mathsf{RndPad}_f(\mathbf{s})$ giving the overall term $\underline{\mathbf{s}^\mathsf{T}\mathbf{A}'_f - \mathsf{RndPad}_{\mathbf{A}_f}(\mathbf{s})}$ as desired.

Randomizing Unbounded Homomorphic Evaluation. Let us now try to plug in the unbounded evaluation procedures of HLL into Wee's construction. As described above, decryption in Wee's CP-ABE recovers the terms

$$\underline{\mathbf{s}^\mathsf{T}(\mathbf{A}_f \otimes \mathbf{r})} + \mu\mathbf{g} \cdot (\mathbf{I} \otimes \mathbf{r}), \quad \underline{\mathbf{s}^\mathsf{T}((\mathbf{A} - \mathbf{x} \otimes \mathbf{G}) \otimes \mathbf{r})}$$

By the mixed product property of tensors, this can be written as

$$\underline{\mathbf{s}^\mathsf{T}(\mathbf{I} \otimes \mathbf{r})\mathbf{A}_f} + \mu\mathbf{g} \cdot (\mathbf{I} \otimes \mathbf{r}), \quad \underline{\mathbf{s}^\mathsf{T}(\mathbf{I} \otimes \mathbf{r})(\mathbf{A} - \mathbf{x} \otimes \mathbf{G})}$$

Denoting $\mathbf{s}^\mathsf{T}(\mathbf{I} \otimes \mathbf{r})$ as $\mathbf{s}_\mathbf{r}^\mathsf{T}$, we obtain well formed ABE encodings

$$\underline{\mathbf{s}_\mathbf{r}^\mathsf{T}\mathbf{A}_f + \mu\mathbf{g} \cdot (\mathbf{I} \otimes \mathbf{r})}, \quad \underline{\mathbf{s}_\mathbf{r}^\mathsf{T}(\mathbf{A} - \mathbf{x} \otimes \mathbf{G})}$$

In particular, we can perform the bounded depth evaluation of [BGG+14] as done by Wee [Wee22] to obtain $\underline{\mathbf{s}_\mathbf{r}^\mathsf{T}(\mathbf{A}_f - f(\mathbf{x})\mathbf{G}))}$ and plug this into the noise removal procedure of HLL. Taking appropriate care, this works out to yield

$$\underline{\mathbf{s}_\mathbf{r}{}^\mathsf{T}(\mathsf{RndPad}_{\mathbf{A}_f}(\mathbf{s}_\mathbf{r}) - f(\mathbf{x})\mathbf{G})}$$

which appears suitable for further processing. To continue further, HLL requires the terms

$$\mathbf{S}_\mathbf{r} = \mathsf{hct}_{\mathbf{s}_\mathbf{r}}(\mathbf{s}_\mathbf{r}), \quad \mathbf{E}_\mathbf{r} = \underline{\mathbf{s}_\mathbf{r}^\mathsf{T}(\mathbf{A}_{\mathsf{circ}} - \mathbf{S}_\mathbf{r} \otimes \mathbf{G})}$$

Above, the subscript in hct denotes the secret key under which the ciphertext can be decrypted. Thus, by HLL, it suffices to compute the above terms so as to bootstrap and continue.

Randomized Bootstrapping. Unfortunately, careful examination shows how demanding the above requirement is. Note that $\mathbf{r}$ is chosen by the key generator while $\mathbf{s}$ (and hence $\mathbf{S} = \mathsf{hct}_\mathbf{s}(\mathbf{s})$) by the encryptor. To randomize $\mathbf{S}$ to $\mathsf{hct}_{\mathbf{s}_\mathbf{r}}(\mathbf{s}_\mathbf{r})$ might seem intuitively possible at first – after all, this is an FHE ciphertext and can be evaluated upon using knowledge of $\mathbf{r}$ to obtain $\mathsf{hct}_\mathbf{s}(\mathbf{s}_\mathbf{r})$. However, while modifying the underlying message is indeed easy, modifying the underlying *secret key* is anything but! In its full generality, modifying the secret key of an FHE scheme should not even be possible since it violates semantic security. Yet, there are amazing ways already known to modify the underlying secret key in FHE schemes – key shrinking in single key FHE [BV11, BGV14] as well as key *expansion* in multi-key FHE [CM15, MW16]. Sadly, neither approach (nor anything related) works for us since they require providing some "advice" terms that cannot be computed in our setting. Note that our situation is additionally complicated by the fact that we are also required to provide a circular encoding of $\mathbf{S}_\mathbf{r}$ under the secret $\mathbf{s}_\mathbf{r}$.

Another difficulty is that if we wish to randomize the circular encoding $\underline{\mathbf{s}^\mathsf{T}(\mathbf{A}_{\mathsf{circ}} - \mathbf{S} \otimes \mathbf{G})}$ using tensors as discussed earlier, our key generator would be required to give a term that looks like $\mathbf{B}^{-1}((\mathbf{A}_{\mathsf{circ}} - \mathbf{S} \otimes \mathbf{G}) \otimes \mathbf{r})$, which it cannot because unlike before, the inner quantity is not fixed and public but depends on $\mathbf{S}$ which is only known to the encryptor.

Our Approach. We overcome both hurdles together by relying on the following obvious fact, which previously seemed like a barrier – FHE ciphertexts are good for computing on the underlying messages, not on the underlying keys. We observe that nothing forces us to provide an FHE ciphertext of $\mathbf{s}$ under the secret key $\mathbf{s}$ itself! We can have our encryptor sample a new FHE scheme with unrelated secret, $\mathbf{t}$ (say) and provide an FHE encryption of $\mathbf{s}$ under the public key corresponding to $\mathbf{t}$. Now, given knowledge of $\mathbf{r}$, an evaluator can easily compute any complicated circuit, including those necessary to compute $\mathbf{S}_\mathbf{r}$ and $\mathbf{E}_\mathbf{r}$ described in Eq. 1. We can also provide an ABE encoding using the same trick of reusing $\mathbf{t}$ as the randomness to cause automatic decryption as was done in HLL.

In more detail, the encryptor can provide

$$\mathbf{T} = \mathsf{hct}_\mathbf{t}(\mathbf{s}, \mathsf{sd}), \quad \mathbf{D} = \underline{\mathbf{t}^\mathsf{T}(\mathbf{A}_1 - (1, \mathsf{bits}(\mathbf{T})) \otimes \mathbf{G})}$$

where sd is a PRF seed, and $\mathbf{A}_1$ is a public matrix of appropriate dimensions. Now, one can homomorphically evaluate on the encoding $\mathbf{D}$ in *bounded* depth, using knowledge of $\mathbf{T}$, to obtain

$$\underline{\mathbf{t}^\mathsf{T}\mathbf{A}'_\mathbf{r} + \mathbf{t}^\mathsf{T}\mathsf{hct}_\mathbf{t}(\mathbf{S}_\mathbf{r}, \mathbf{E}_\mathbf{r})} = \underline{\mathbf{t}^\mathsf{T}\mathbf{A}'_\mathbf{r} + (\mathbf{S}_\mathbf{r}, \mathbf{E}_\mathbf{r})}$$

where $\mathbf{A}'_\mathbf{r}$ is some $\mathbf{r}$ dependent matrix and the equality follows by automatic decryption. Recall that:

$$\mathbf{S}_\mathbf{r} = \mathsf{hct}(\mathbf{A}_{\mathbf{s}_\mathbf{r}}, \mathbf{s}_\mathbf{r}), \quad \mathbf{E}_\mathbf{r} = \underline{\mathbf{s}_\mathbf{r}^\mathsf{T}(\mathbf{A}_{\mathsf{circ}} - (1, \mathsf{bits}(\mathbf{S}_\mathbf{r})) \otimes \mathbf{G})}$$

which we require to plug into the HLL bootstrapping procedure.

Now, our desired output is masked by $\mathbf{t}^\mathsf{T}\mathbf{A}'_\mathbf{r}$ so it is unclear we have achieved anything. However, note that we now have the happy situation that $\mathbf{t}$ is known to the encryptor and $\mathbf{A}'_\mathbf{r}$ can be computed by the key generator! So we can plug back evasive LWE so that the encryptor gives $\underline{\mathbf{t}^\mathsf{T}\mathbf{C}}$ for some fixed matrix $\mathbf{C}$ and the key generator gives $\mathbf{C}^{-1}(\mathbf{A}'_\mathbf{r})$ so that we can recover the term $\underline{\mathbf{t}^\mathsf{T}\mathbf{A}'_\mathbf{r}}$ and cancel it out. Please see Section 3 for the detailed description.

Onward to Unbounded CP-ABE. The core idea for randomizing the bootstrapping discussed above applied carefully to Wee's construction allows us to obtain a correct scheme supporting unbounded depth circuits. However, proving security must still contend with several hurdles. While the high level structure of our proof resembles the proof of Wee's CP-ABE, our distributions are significantly more complex and need more work to analyze. To argue indistnguishability of hybrids, we must rely on the new assumptions described previously. We refer the reader to Section 4 for a detailed description of the assumptions, the scheme and the proof.

2 Preliminaries

In this section, we define some notation and preliminaries that we require.

Notation. We use bold letters to denote vectors and use the convention of vectors being columns. The notation $[a, b]$ denotes the set of integers $\{k \in \mathbb{N} \mid a \leq k \leq b\}$. We use $[n]$ to denote the set $[1, n]$. When we consider a string of form $(x, 1^t)$, we assume that x and 1^t can be derived from it and we differentiate $(x, 1^t)$ from $(x1, 1^{t-1})$ for instance. To do so, we consider the alphabet "," in addition to 0 and 1 and represent each alphabet by 2-bit for example. We say a function $f(n)$ is *negligible* if it is $O(n^{-c})$ for all $c > 0$, and we use $\mathsf{negl}(n)$ to denote a negligible function of n. We say $f(n)$ is *polynomial* if it is $O(n^c)$ for some constant $c > 0$, and we use $\mathrm{poly}(n)$ to denote a polynomial function of n. We use the abbreviation PPT for probabilistic polynomial-time. We say an event occurs with *overwhelming probability* if its probability is $1 - \mathsf{negl}(\lambda)$. For two distributions X_λ and Y_λ, $X_\lambda \approx_c Y_\lambda$ denotes that they are computationally indistinguishable for any PPT algorithm. For a vector $\mathbf{x}$, we let x_i denote its i-th entry. For a set S, we let $|S|$ denote the number of elements in S. For a binary string x, we let $|x|$ denote the length of x.

2.1 Garbled Circuits

Our definition of garbled circuits is based upon the one considered in [GKW16]. For $\lambda \in \mathbb{N}$, let $\mathcal{C}_{\mathsf{inp}}$ denote a family of circuits with inp bit inputs and $\mathcal{C} = \{\mathcal{C}_{\mathsf{inp}(\lambda)}\}_{\lambda \in \mathbb{N}}$. A garbling scheme for $\mathcal{C}$ consists of two algorithms $\mathsf{GC} = (\mathsf{Garble}, \mathsf{Eval})$ with the following syntax.

$\mathsf{Garble}(1^\lambda, C) \to \{\mathsf{lab}_{i,b}\}_{i \in [\mathsf{inp}], b \in \{0,1\}}$. The garbling algorithm takes as input the security parameter λ and a circuit $C \in \mathcal{C}_{\mathsf{inp}(\lambda)}$. It outputs a set of labels $\{\mathsf{lab}_{i,b}\}_{i \in [\mathsf{inp}], b \in \{0,1\}}$.

$\mathsf{Eval}(\{\mathsf{lab}_i\}_{i \in [\mathsf{inp}]}) \to y$. The evaluation algorithm takes as input an inp labels $\{\mathsf{lab}_i\}_{i \in [\mathsf{inp}]}$ and outputs y.

Definition 2.1 (Correctness). A garbling scheme GC for circuit family $\{\mathcal{C}_{\mathsf{inp}(\lambda)}\}_{\lambda \in \mathbb{N}}$ is correct if for all $C \in \mathcal{C}_{\mathsf{inp}(\lambda)}$ and all $x \in \{0,1\}^{\mathsf{inp}(\lambda)}$, we have

$$\Pr\left[\mathsf{Eval}\left(\{\mathsf{lab}_{i,x_i}\}_{i \in [\mathsf{inp}]}\right) \neq C(x) : \left(\{\mathsf{lab}_{i,b}\}_{i \in [\mathsf{inp}], b \in \{0,1\}}\right) \leftarrow \mathsf{Garble}(1^\lambda, C)\right] = \mathsf{negl}(\lambda),$$

where the probability is taken over the random coins of Garble.

Definition 2.2 (Security). A garbling scheme $\mathsf{GC} = (\mathsf{Garble}, \mathsf{Eval})$ for $\mathcal{C} = \{\mathcal{C}_{\mathsf{inp}(\lambda)}\}_{\lambda \in \mathbb{N}}$ is said to be a secure garbling scheme if there exists a PPT simulator Sim such that for all $\lambda \in \mathbb{N}, \mathsf{inp}, C \in \mathcal{C}_{\mathsf{inp}(\lambda)}$ and $x \in \{0,1\}^{\mathsf{inp}(\lambda)}$, we have

$$\begin{aligned} & \left\{ \{\mathsf{lab}_i\}_{i \in [\mathsf{inp}]} \ : \ \{\mathsf{lab}_i\}_{i \in [\mathsf{inp}]} \leftarrow \mathsf{Sim}\left(1^\lambda, 1^{\mathsf{inp}}, 1^{|C|}, C(x)\right) \right\} \\ \approx_c \quad & \left\{ \{\mathsf{lab}_{i,x_i}\}_{i \in [\mathsf{inp}]} \ : \ \{\mathsf{lab}_{i,b}\}_{i \in [\mathsf{inp}], b \in \{0,1\}} \leftarrow \mathsf{Garble}\left(1^\lambda, C\right) \right\} \end{aligned}$$

Garbled circuits are known to exist from one-way functions [Yao82, BHR12b]. Though the above definition is not as general as one in [BHR12b], it suffices for our constructions in this paper.

Remark 2.3. Note that in the standard syntax of garbling scheme [LP09, BHR12a], the output of the garbling algorithm consists of labels along with a garbled circuit. On the other hand, the output only consists of labels in our syntax. However, the former can easily be converted into the latter by including the garbled circuit into a label for example. This change in syntax is made for simplifying our constructions and notations. We also note that the same primitive is called decomposable randomized encoding in [GVW12].

Remark 2.4 (Multi-instance security definition). The above definition allows to argue security of a single instance of GC. In the multi-instance variant, the adversary can adaptively make polynomial number of garbling queries to the challenger. All queries are answered by honestly generated labels in the real world whereas they are all simulated in the ideal world. If both worlds are computationally indistinguishable, we say that GC satisfies multi-instance security. Note that such *multi-instance* security follows from the standard single instance security (from Definition 2.2 above) by a set of simple hybrid arguments.

2.2 Identity-Based Encryption

We define identity-based encryption (IBE) [KT18, BF01, Coc01] in this section. An IBE scheme IBE with identity space $\mathcal{I} = \{\mathcal{I}_\lambda\}_{\lambda\in\mathbb{N}}$ consists of the following algorithms.

$\mathsf{Setup}(1^\lambda) \to (\mathsf{mpk}, \mathsf{msk})$. The setup algorithm takes as input the unary representation of security parameter and outputs the master public and secret keys mpk and msk.

$\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}) \to \mathsf{sk}_{\mathsf{id}}$. The key generation algorithm takes as input the master secret key msk and an identity $\mathsf{id} \in \mathcal{I}_\lambda$. It outputs a corresponding secret key $\mathsf{sk}_{\mathsf{id}}$.

$\mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, m) \to \mathsf{ct}$. The encryption algorithm takes as input the master public parameter mpk, an identity $\mathsf{id} \in \mathcal{I}_\lambda$ and a message $m \in \{0,1\}^*$. It outputs a ciphertext ct.

$\mathsf{Dec}(\mathsf{sk}_{\mathsf{id}}, \mathsf{ct}) \to m'/\perp$. The decryption algorithm takes as input the secret key $\mathsf{sk}_{\mathsf{id}}$ and a ciphertext ct, and outputs either a message $m' \in \{0,1\}^*$ or $\perp$.

Definition 2.5 (Correctness). For correctness, we require that for all $\lambda \in \mathbb{N}, m \in \{0,1\}^*, \mathsf{id} \in \mathcal{I}_\lambda$,

$$\Pr\left[\begin{array}{c} \mathsf{Dec}(\mathsf{sk}_{\mathsf{id}}, \mathsf{ct}_{\mathsf{id}}) = m \mid (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^L), \\ \mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}), \mathsf{ct}_{\mathsf{id}} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, m) \end{array}\right] = 1,$$

where the probability is taken over the random coins of Setup, KeyGen and Enc.

Remark 2.6 (About the identity space.). In this paper, we set the identity space $\mathcal{I}$ to be $\{0,1\}^*$ by default. This is without loss of generality assuming collision resistant hash function and an IBE scheme with sufficiently large identity space (e.g., $\mathcal{I}_\lambda = \{0,1\}^{2\lambda}$), since we can hash any string into a string of fixed length using collision resistant hash function.

Remark 2.7 (About the message length.). In the above definition, we assume that the message space is $\{0,1\}^*$. This is without loss of generality if we consider $\mathsf{IND\text{-}CPA}$ security for IBE, since it is straightforward to extend the message space by encrypting each bit (or chunk) of the message in parallel.

Definition 2.8 ($\mathsf{IND\text{-}CPA}$ Security). An IBE scheme IBE for an identity space $\mathcal{I}_\lambda$ and message space $\{0,1\}^*$ is said to satisfy indistinguishability based security if for any stateful PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(.)$ such that

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

where for each $b \in \{0,1\}$ and $\lambda \in \mathbb{N}$, the experiment $\mathsf{Exp}^{(b)}_{\mathsf{IBE},\mathcal{A}}$, modelled as a game between adversary and challenger, is defined as follows:

1. **Setup Phase:** On input 1^λ from the adversary $\mathcal{A}$, the challenger samples $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and replies to $\mathcal{A}$ with mpk.

2. **Query phase:** During the game, $\mathcal{A}$ adaptively makes the following queries in any arbitrary order and unbounded many times.

   (a) **Key Queries:** $\mathcal{A}$ chooses an identity $\mathsf{id} \in \mathcal{I}_\lambda$ and sends it to the challenger. For each such query, the challenger replies with $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$.

   (b) **Encryption Queries:** $\mathcal{A}$ submits to the challenger, an identity $\mathsf{id}^* \in \mathcal{I}_\lambda$ and a pair of equal length messages (m_0, m_1). The challenger replies with $\mathsf{ct}_{\mathsf{id}^*} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}^*, m_b)$.

3. **Output phase:** $\mathcal{A}$ outputs it's guess b' as output of the experiment.

We say an adversary $\mathcal{A}$ is legitimate if during the challenge phase, it is restricted to query for ciphertexts corresponding to identities $\mathsf{id}^* \in \mathcal{I}_\lambda$ whose secret keys are not queried during any key query.

Remark 2.9. Since we are in the public key setting, we can simplify the above experiment so that the encryption query is allowed for only once. This simplified definition is shown to equivalent to the definition above by a simple hybrid argument. We adopt the above multi-challenge security definition since it is convenient for our purpose in this paper.

2.3 Turing Machines

Here, we recall the definition of a Turing machine (TM) following [LL20a]. The definition considers Turing machines with two tapes, namely, input tape and working tape.

Definition 2.10 (Turing Machine). A (deterministic) TM M is represented by the tuple $M = (Q, \delta, F)$ where Q is the number of states (we use $[Q]$ as the set of states and 1 as the initial state), $F \subset [Q]$ is the set of accepting state and

$$\begin{aligned} \delta : [Q] \times \{0,1\} \times \{0,1\} &\to [Q] \times \{0,1\} \times \{0,\pm 1\} \times \{0,\pm 1\} \\ (q, b_1, b_2) &\mapsto (q', b_2', \Delta i, \Delta j) \end{aligned}$$

is the state transition function, which, given the current state q, the symbol b_1 on the input tape under scan, and the symbol b_2 on the work tape under scan, specifies the new state q', the symbol b_2', overwriting b_2, the direction Δi to which the input tape pointers moves, and the direction Δj to which the work tape pointer moves. The machine is required to hang (instead of halting) once it reaches an accepting state, i.e., for all $q \in [Q]$ such that $q \in F$ and $b_1, b_2 \in \{0,1\}$, it holds that $\delta(q, b_1, b_2) = (q, b_2, 0, 0)$.

For input length $n \geq 1$ and space complexity bound $s \geq 1$, the set of internal configurations of M is

$$\mathcal{Q}_{M,n,s} = [n] \times [s] \times \{0,1\}^s \times [Q]$$

where $(i, j, W, q) \in \mathcal{Q}_{M,n,s}$ specifies the input tape pointer $i \in [n]$, the work tape pointer $j \in [s]$, the content of the work tape $W \in \{0,1\}^s$ and the machine state $q \in [Q]$.

For any bit-string $x \in \{0,1\}^n$ for $n \geq 1$ and time/space complexity bounds $t, s \geq 1$, the machine M accepts x within time t and space s if there exists a sequence of internal configurations (computation path of t steps) $c_0, \ldots, c_t \in \mathcal{Q}_{M,n,s}$ with $c_k = (i_k, j_k, W_k, q_k)$ such that $(i_0, j_0, W_0, q_0) = (1, 1, 0^s, 1)$ (initial configuration),

$$\text{for all } 0 \leq k < t\text{:} \quad \begin{cases} \delta(q_k, x[i_k], W_k[j_k]) = (q_{k+1}, W_{k+1}[j_k], i_{k+1} - i_k, j_{k+1} - j_k) \\ W_{k+1}[j] = W_k[j] \quad \text{for all } j \neq j_k \qquad \text{(valid transitions);} \end{cases}$$

where $x[i]$ is the i the bit of the string x and $W_k[j]$ is the j the bit of the string W_k, and $q_t \in F$ (accepting). We also say M accepts x within time t (without the space bound) if M accepts x within time t and space $s = t$.

Next, we define time/space bounded computation with *non-deterministic* Turing machines. The definition is the same as Definition 2.10, except with the following changes:

- The transition criterion δ can be any relation between (i.e., any subset of the Cartesian product of) $[Q] \times \{0,1\}^2$ and $[Q] \times \{0,1\} \times \{0, \pm q\}^2$, where $((q, b_1, b_2), (b_2', b_2', \Delta i, \Delta j)) \in \delta$ means that if the current state is q, the input tape symbol under scan is b_1 and the work tape symbol under scan is b_2, then it is valid to transit into state q', overwrite b_2 with b_2', and move the input and work tape pointer by offsets Δi and Δj respectively.

- The definition of hanging in accepting states is that for all $q \in [Q]$ such that $q \in F$ and all $b_1, b_2 \in \{0,1\}$,

$$\delta \cap \left(\{(q, b_1, b_2)\} \times ([Q] \times \{0,1\} \times \{0, \pm 1\}^2) \right) = \{(q, b_1, b_2), (q, b_2, 0, 0)\}.$$

- In the definition of acceptance

$$\delta(q_k, x[i_k], W_k[j_k]) = (q_{k+1}, W_{k+1}[j_k], i_{k+1} - i_k, j_{k+1} - j_k)$$

$$\text{is changed to } ((q_k, x[i_k], W_k[j_k]), (q_{k+1}, W_{k+1}[j_k], i_{k+1} - i_k, j_{k+1} - j_k)) \in \delta.$$

The following lemma can be obtained by a simple argument on emulating a Turing machine on Boolean circuits. While we have many other clever methods of simulating Turing machines on circuits (e.g., [PF79]), we use the following simple one because it evaluates the depth of the circuit as a function on the size of Turing machine, which is usually regarded as a constant and ignored.

Lemma 2.11 (Emulating a Turing Machine on Circuit). Consider a circuit that takes as input a description of a (deterministic) Turing machine $M = (Q, \delta, F)$, input x to M, a configuration $(i, j, W, q) \in \mathcal{Q}_{M,|x|,|W|}$ and outputs the next configuration (i', j', W', q'). We can implement such a circuit with depth $\mathrm{poly}(\log |x|, \log |W|, \log |M|)$ and size $\mathrm{poly}(|x|, |W|, |M|)$.

Proof. The circuit is implemented as follows. We focus on the depth of the circuits, since the bound on the size will be clear from the description. Given the input, it first retrieves the i-th bit $x[i] \in \{0,1\}$ of the input tape. This can be done by a circuit with depth $O(\log |x|)$, which checks whether $i = \nu$ or not for each position ν of the string x in parallel and returns $x[\nu]$ for ν such that $\nu = i$. Similarly, it can retrieve $W[j]$ with depth $O(\log |W|)$. Given $x[i]$ and $W[j]$, it then retrieves $\delta(q, x[i], W[j])$ from δ, which can be done with depth $O(\log |Q|)$ similarly to the above. Given $\delta(q, x[i], W[j]) = (q', b', \Delta i, \Delta j)$, the update of i and j can be done in depth $O(\log |x|)$ and $O(\log |W|)$, respectively. Writing back the new value b' can also be done in depth $O(\log |W|)$ by finding the right place to write in the tape and change the value there. From the above discussion, the total depth of the circuit is $\mathrm{poly}(\log |x|, \log |W|, \log |M|)$ as desired. □

We also need the following lemma, which can be obtained by a simple observation.

Lemma 2.12 (Checking Transition for Non-deterministic Turing Machine). Consider a circuit that takes as input a description of a non-deterministic Turing machine $M = (Q, \delta, F)$, input x to M, two configurations $(i, j, W, q) \in \mathcal{Q}_{M,|x|,|W|}$ and $(i', j', W', q') \in \mathcal{Q}_{M,|x|,|W|}$ and outputs whether $((i, j, W, q), (i', j', W', q')) \in \delta$ or not. We can implement such a circuit with depth $\mathrm{poly}(\log |x|, \log |W|, \log |M|)$ and size $\mathrm{poly}(|x|, |W|, |M|)$.

Proof. The circuit is implemented as follows. We focus on the depth of the circuits, since the bound on the size will be clear from the description. Given the input, it first checks whether $i' - i \in \{0, \pm 1\}$, $j' - j \in \{0, \pm 1\}$. Clearly, this can be done in depth $\mathrm{poly}(\log |x|, \log |W|)$. It then checks whether $W'[k] = W[k]$ for all $k \in [|W|] \backslash \{k\}$, which can be done in depth $\mathrm{poly}(\log |W|)$. It then checks whether $((q, x[i], W[j]), (q', W'[j], i' - i, j' - j)) \in \delta$ or not. This can be checked in depth $\mathrm{poly}(\log |Q|)$. Therefore, the total depth of the circuit is $\mathrm{poly}(\log |x|, \log |W|, \log |M|)$ as desired. □

2.4 Attribute Based Encryption

Let $R : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ be a relation where $\mathcal{X}$ and $\mathcal{Y}$ denote "ciphertext attribute" and "key attribute" spaces, respectively. Ideally, we would like to have an ABE scheme that handles the relation R directly, where we can encrypt w.r.t any ciphertext attribute $x \in \mathcal{X}$ and can generate a secret key for any key attribute $y \in \mathcal{Y}$. However, in many cases, we are only able to construct a scheme that poses restrictions on the ciphertext attribute space and key attribute space. To capture such restrictions, we introduce a parameter prm and consider subsets of the domains $\mathcal{X}_{\mathsf{prm}} \subseteq \mathcal{X}$ and $\mathcal{Y}_{\mathsf{prm}} \subseteq \mathcal{Y}$ specified by it and the function R_{prm} defined by restricting the function R on $\mathcal{X}_{\mathsf{prm}} \times \mathcal{Y}_{\mathsf{prm}}$.

An attribute-based encryption (ABE) scheme for $R = \{R_{\mathsf{prm}} : \mathcal{X}_{\mathsf{prm}} \times \mathcal{Y}_{\mathsf{prm}} \to \{0,1\}\}_{\mathsf{prm}}$ and a message space $\mathcal{M}$ is defined by the following algorithms.

$\mathsf{Setup}(1^\lambda, \mathsf{prm}) \to (\mathsf{mpk}, \mathsf{msk})$. The setup algorithm takes as input the unary representation of the security parameter λ and a parameter prm and outputs a master public key mpk and a master secret key msk.

$\mathsf{Enc}(\mathsf{mpk}, X, \mu) \to \mathsf{ct}_X$. The encryption algorithm takes as input a master public key mpk, a ciphertext attribute $X \in \mathcal{X}_{\mathsf{prm}}$, and a message $\mu \in \mathcal{M}$. It outputs a ciphertext ct_X.

$\mathsf{KeyGen}(\mathsf{msk}, Y) \to \mathsf{sk}_Y$. The key generation algorithm takes as input the master secret key msk and a key attribute $Y \in \mathcal{Y}_{\mathsf{prm}}$. It outputs a private key sk_Y.

$\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_Y, Y, \mathsf{ct}_X, X) \to \mu$ or $\perp$. The decryption algorithm takes as input the master public key mpk, a private key sk_Y, private key attribute $Y \in \mathcal{Y}_{\mathsf{prm}}$, a ciphertext ct_X and ciphertext attribute $X \in \mathcal{X}_{\mathsf{prm}}$. It outputs the message μ or $\perp$ which represents that the ciphertext is not in a valid form.

Definition 2.13 (Correctness). An ABE scheme for relation family R is correct if for all prm, $X \in \mathcal{X}_{\mathsf{prm}}, Y \in \mathcal{Y}_{\mathsf{prm}}$ such that $R(X, Y) = 0$, and for all messages $\mu \in \mathcal{M}$,

$$\Pr\left[\begin{array}{l}(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{prm}),\\ \mathsf{sk}_Y \leftarrow \mathsf{KeyGen}(\mathsf{msk}, Y),\\ \mathsf{ct}_X \leftarrow \mathsf{Enc}(\mathsf{mpk}, X, \mu) :\\ \mathsf{Dec}\Big(\mathsf{mpk}, \mathsf{sk}_Y, Y, \mathsf{ct}_X, X\Big) \neq \mu\end{array}\right] = \mathsf{negl}(\lambda)$$

where the probability is taken over the coins of Setup, KeyGen, and Enc.

Definition 2.14 (Sel-IND security for ABE). For an ABE scheme $\mathsf{ABE} = \{\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec}\}$ for a relation family $R = \{R_{\mathsf{prm}} : \mathcal{X}_{\mathsf{prm}} \times \mathcal{Y}_{\mathsf{prm}} \to \{0,1\}\}_{\mathsf{prm}}$ and a message space $\mathcal{M}$ and an adversary $\mathcal{A}$, let us define Sel-IND security game $\mathsf{Exp}_{\mathsf{ABE},\mathcal{A}}(1^\lambda)$ as follows.

1. $\mathcal{A}$ outputs prm and the challenge ciphertext attribute $X^\star \in \mathcal{X}_{\mathsf{prm}}$.
2. **Setup phase:** On input $1^\lambda, \mathsf{prm}$, the challenger samples $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{prm})$ and gives mpk to $\mathcal{A}$.
3. **Query phase:** During the game, $\mathcal{A}$ adaptively makes the following queries, in an arbitrary order. $\mathcal{A}$ can make unbounded many key queries, but can make only single challenge query.
   (a) **Key Queries:** $\mathcal{A}$ chooses an input $Y \in \mathcal{Y}_{\mathsf{prm}}$. For each such query, the challenger replies with $\mathsf{sk}_Y \leftarrow \mathsf{KeyGen}(\mathsf{msk}, Y)$.
   (b) **Challenge Query:** At some point, $\mathcal{A}$ submits a pair of equal length messages $(\mu_0, \mu_1) \in \mathcal{M}^2$ to the challenger. The challenger samples a random bit $b \leftarrow \{0,1\}$ and replies to $\mathcal{A}$ with $\mathsf{ct}_{X^\star} \leftarrow \mathsf{Enc}(\mathsf{mpk}, X^\star, \mu_b)$.

   We require that $R(X^\star, Y) = 1$ holds for any Y such that $\mathcal{A}$ makes a key query for Y in order to avoid trivial attacks.
4. **Output phase:** $\mathcal{A}$ outputs a guess bit b' as the output of the experiment.

We define the advantage $\mathsf{Adv}^{\mathsf{Sel\text{-}IND}}_{\mathsf{ABE},\mathcal{A}}(1^\lambda)$ of $\mathcal{A}$ in the above game as

$$\mathsf{Adv}^{\mathsf{Sel\text{-}IND}}_{\mathsf{ABE},\mathcal{A}}(1^\lambda) := \left|\Pr\Big[\mathsf{Exp}_{\mathsf{ABE},\mathcal{A}}(1^\lambda) = 1 | b = 0\Big] - \Pr\Big[\mathsf{Exp}_{\mathsf{ABE},\mathcal{A}}(1^\lambda) = 1 | b = 1\Big]\right|.$$

The ABE scheme ABE is said to satisfy Sel-IND security (or simply *selective security*) if for any stateful PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\mathsf{Adv}^{\mathsf{Sel\text{-}IND}}_{\mathsf{ABE},\mathcal{A}}(1^\lambda) = \mathsf{negl}(\lambda)$.

We also consider the very selective version of the security.

Definition 2.15 (VerSel-IND security for ABE). We define VerSel-IND security game similarly to Sel-IND security game except that the adversary $\mathcal{A}$ outputs the key queries $Y_1, \ldots, Y_Q$, where Q is the number of key queries made by $\mathcal{A}$, along with the challenge ciphertext attribute $X^\star$ in the beginning of the security game. We define the advantage $\mathsf{Adv}^{\mathsf{VerSel\text{-}IND}}_{\mathsf{ABE},\mathcal{A}}(1^\lambda)$ of the adversary $\mathcal{A}$ accordingly and say that the scheme satisfies VerSel-IND security if the quantity is negligible.

In the following, we recall definitions of various ABEs by specifying the relation.

Key-policy Attribute Based encryption (kpABE) for circuits. To define kpABE for circuits, we set $\mathcal{X} = \{0,1\}^*$ and $\mathcal{Y}$ as the set of all circuits and define $R(\mathbf{x}, C) = C(\mathbf{x})$. In this paper, we consider circuit class $\mathcal{C}_{\mathsf{inp},\mathsf{dep}}$ that consists of circuits with input length $\mathsf{inp} := \mathsf{inp}(\lambda)$ and depth $\mathsf{dep} := \mathsf{dep}(\lambda)$. To do so, we set $\mathsf{prm} = (1^{\mathsf{inp}}, 1^{\mathsf{dep}})$, $\mathcal{X}_{\mathsf{prm}} = \{0,1\}^{\mathsf{inp}}$, and $\mathcal{Y}_{\mathsf{prm}} = \mathcal{C}_{\mathsf{inp},\mathsf{dep}}$.

Ciphertext-policy Attribute Based encryption (cpABE) for circuits. To define cpABE for circuits, we set $\mathcal{Y} = \{0,1\}^*$ and $\mathcal{X}$ as the set of all circuits and define $R(C, \mathbf{x}) = C(\mathbf{x})$. In this paper, we consider circuit class $\mathcal{C}_{\mathsf{inp},\mathsf{dep}}$ that consists of circuits with input length $\mathsf{inp} := \mathsf{inp}(\lambda)$ and depth $\mathsf{dep} := \mathsf{dep}(\lambda)$. To do so, we set $\mathsf{prm} = (1^{\mathsf{inp}}, 1^{\mathsf{dep}})$, $\mathcal{X}_{\mathsf{prm}} = \mathcal{C}_{\mathsf{inp},\mathsf{dep}}$, and $\mathcal{Y}_{\mathsf{prm}} = \{0,1\}^{\mathsf{inp}}$.

ABE for Turing Machines. To define ABE for Turing machines, we set $\mathcal{X} = \{0,1\}^*$, $\mathcal{Y}$ to be set of all Turing machine, and define $R : \mathcal{X} \times \mathcal{Y} \to \{0,1\} \cup \{\bot\}$ as

$$R((\mathbf{x}, 1^t), M) = \begin{cases} 0 & \text{if } M \text{ accepts } \mathbf{x} \text{ in } t \text{ steps} \\ 1 & \text{otherwise.} \end{cases}$$

ABE for NL. To define ABE for NL, we set $\mathcal{X} = \{0,1\}^*$, $\mathcal{Y}$ to be set of all non-deterministic Turing machines with two tapes, one of which encodes the input and can only be read, whereas the other tape can be read as well as written. When we measure the space complexity of the computation, we consider the space being used for the latter tape. We define $R : \mathcal{X} \times \mathcal{Y} \to \{0,1\} \cup \{\bot\}$ as

$$R((\mathbf{x}, 1^t, 1^{2^s}), M) = \begin{cases} 0 & \text{if } M \text{ accepts } \mathbf{x} \text{ within } t \text{ steps and space } s \\ 1 & \text{otherwise.} \end{cases}$$

Note that here, s is in the exponent to reflect the idea that the space for the computation is logarithmically bounded.

We will use the kpABE scheme given by [BGG+14] for our constructions. The following theorem summarizes the properties of the scheme.

Theorem 2.16 (Properties of [BGG+14]). There exists a key-policy ABE scheme $\mathsf{ABE} = \mathsf{ABE}.(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ for function class $\mathcal{C}_{\ell,d}$ which is selectively secure under the LWE assumption and has the following properties. In particular:

Key Compactness. We have $|\mathsf{ABE.sk}_C| \leq \mathrm{poly}(\lambda, d)$ for any $C \in \mathcal{C}_{\ell,d}$, where $(\mathsf{ABE.mpk}, \mathsf{ABE.msk}) \leftarrow \mathsf{ABE.Setup}(1^\lambda)$ and $\mathsf{ABE.sk}_C \leftarrow \mathsf{ABE.KeyGen}(\mathsf{ABE.msk}, C)$. In particular, the length of the secret key is independent of the attribute length ℓ and the size of the circuit C.

Parameters Succinctness. We have $|\mathsf{ABE.mpk}|, |\mathsf{ABE.msk}| \leq \mathrm{poly}(\lambda, d, \ell)$ and $|\mathsf{ABE.ct}| \leq \mathrm{poly}(\lambda, d, \ell) + |\mu|$ for any $\mathbf{x} \in \mathcal{X}_\lambda$ and $\mu \in \mathcal{M}_\lambda$, where $(\mathsf{ABE.mpk}, \mathsf{ABE.msk}) \leftarrow \mathsf{ABE.Setup}(1^\lambda)$ and $\mathsf{ABE.ct} \leftarrow \mathsf{ABE.Enc}(\mathsf{ABE.mpk}, \mathbf{x}, \mu)$.

We will also use the cpABE scheme given by [Wee22]. The following theorem summarizes the properties of the scheme.

Theorem 2.17 (Properties of [Wee22]). There exists a ciphertext policy ABE scheme $\mathsf{cpABE} = (\mathsf{cpABE.Setup}, \mathsf{cpABE.KeyGen}, \mathsf{cpABE.Enc}, \mathsf{cpABE.Dec})$ for function class $\mathcal{C}_{\ell,d}$, which is very selectively secure under the evasive LWE and tensor LWE assumption and has the following properties. In particular:

Ciphertext Compactness. We have $|\mathsf{cpABE.ct}| \leq \mathrm{poly}(\lambda, d) + |\mu|$ for any $C \in \mathcal{C}_{\ell,d}$ and $\mu \in \mathcal{M}_\lambda$, where $(\mathsf{cpABE.mpk}, \mathsf{cpABE.msk}) \leftarrow \mathsf{cpABE.Setup}(1^\lambda)$ and $\mathsf{cpABE.ct} \leftarrow \mathsf{cpABE.Enc}(\mathsf{cpABE.mpk}, C, \mu)$. In particular, the size of the ciphertext is independent from the size of the circuit C and its input length.

Parameters Succinctness. We have $|\mathsf{cpABE.mpk}|, |\mathsf{cpABE.msk}|, |\mathsf{cpABE.sk}_{\mathbf{x}}| \leq \mathrm{poly}(\lambda, \ell, d)$ for any $\mathbf{x} \in \{0,1\}^\ell$, where $(\mathsf{cpABE.mpk}, \mathsf{cpABE.msk}) \leftarrow \mathsf{cpABE.Setup}(1^\lambda)$ and $\mathsf{cpABE.sk} \leftarrow \mathsf{cpABE.KeyGen}(\mathsf{cpABE.msk}, \mathbf{x})$.

2.5 Tensors

In this work, similarly to [Wee22], we use the tensor product techniques. Let $\mathbf{A} = (a_{i,j}) \in \mathbb{Z}_q^{m \times n}$ and $\mathbf{B} \in \mathbb{Z}_q^{s \times t}$. The tensor product is defined as:

$$\mathbf{A} \otimes \mathbf{B} \stackrel{\text{def}}{=} \begin{pmatrix} a_{1,1}\mathbf{B} & \cdots & a_{1,n}\mathbf{B} \\ \vdots & & \vdots \\ a_{m,1}\mathbf{B} & \cdots & a_{m,n}\mathbf{B} \end{pmatrix} \in \mathbb{Z}_q^{ms \times nt}.$$

Throughout the paper, we will heavily use the mixed-product equality, stated as follows. Let $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$, $\mathbf{B} \in \mathbb{Z}_q^{s \times t}$, $\mathbf{C} \in \mathbb{Z}_q^{n \times u}$ and $\mathbf{D} \in \mathbb{Z}_q^{t \times v}$,

$$(\mathbf{A} \otimes \mathbf{B}) \cdot (\mathbf{C} \otimes \mathbf{D}) = (\mathbf{AC}) \otimes (\mathbf{BD}) \in \mathbb{Z}_q^{ms \times uv}.$$

The mixed-product can be naturally generalized following

$$(\mathbf{A}^1 \otimes \cdots \otimes \mathbf{A}^k) \cdot (\mathbf{B}^1 \otimes \cdots \otimes \mathbf{B}^k) = (\mathbf{A}^1\mathbf{B}^1) \otimes \cdots \otimes (\mathbf{A}^k\mathbf{B}^k).$$

Note that we adopt the same convention as in [Wee22] where matrix multiplication takes precedence over tensor products, i.e. $\mathbf{A} \otimes \mathbf{BC} = \mathbf{A} \otimes (\mathbf{BC})$.

2.6 Lattice Preliminaries

Here, we recall some facts on lattices that are needed for the exposition of our construction. Throughout this section, n, m, and q are integers such that $n = \mathrm{poly}(\lambda)$ and $m \geq n\lceil \log q \rceil$. In the following, let $\mathsf{SampZ}(\gamma)$ be a sampling algorithm for the truncated discrete Gaussian distribution over $\mathbb{Z}$ with parameter $\gamma > 0$ whose support is restricted to $z \in \mathbb{Z}$ such that $|z| \leq \sqrt{n}\gamma$.

Let

$$\mathbf{g} = (2^0, 2^1, \ldots, 2^{\frac{m}{n+1}-1})^\intercal, \quad \mathbf{G} = \mathbf{I}_{n+1} \otimes \mathbf{g}^\intercal$$

be the gadget vector and the gadget matrix. For $\mathbf{p} \in \mathbb{Z}_q^n$, we write $\mathbf{G}^{-1}(\mathbf{p})$ for the m-bit vector $(\mathsf{bits}(\mathbf{p}[1]), \ldots, \mathsf{bits}(\mathbf{p}[n+1]))^\intercal$, where $\mathsf{bits}(\mathbf{p}[i])$ are $m/(n+1)$ bits for each $i \in [n+1]$. The notation extends column-wise to matrices and it holds that $\mathbf{G}\mathbf{G}^{-1}(\mathbf{P}) = \mathbf{P}$.

Trapdoors. Let us consider a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$. For all $\mathbf{V} \in \mathbb{Z}_q^{n \times m'}$, we let $\mathbf{A}^{-1}(\mathbf{V})$ be an output distribution of $\mathsf{SampZ}(\gamma)^{m \times m'}$ conditioned on $\mathbf{A} \cdot \mathbf{A}^{-1}(\mathbf{V}, \gamma) = \mathbf{V}$. A γ-trapdoor for $\mathbf{A}$ is a trapdoor that enables one to sample from the distribution $\mathbf{A}^{-1}(\mathbf{V}, \gamma)$ in time $\mathrm{poly}(n, m, m', \log q)$ for any $\mathbf{V}$. We slightly overload notation and denote a γ-trapdoor for $\mathbf{A}$ by $\mathbf{A}_\gamma^{-1}$. The following properties had been established in a long sequence of works [GPV08, CHKP10, ABB10a, ABB10b, MP12, BLP+13].

Lemma 2.18 (Properties of Trapdoors). Lattice trapdoors exhibit the following properties.

1. Given $\mathbf{A}_\tau^{-1}$, one can obtain $\mathbf{A}_{\tau'}^{-1}$ for any $\tau' \geq \tau$.
2. Given $\mathbf{A}_\tau^{-1}$, one can obtain $[\mathbf{A}\|\mathbf{B}]_\tau^{-1}$ and $[\mathbf{B}\|\mathbf{A}]_\tau^{-1}$ for any $\mathbf{B}$.
3. There exists an efficient procedure $\mathsf{TrapGen}(1^n, 1^m, q)$ that outputs $(\mathbf{A}, \mathbf{A}_{\tau_0}^{-1})$ where $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ for some $m = O(n \log q)$ and is 2^{-n}-close to uniform, where $\tau_0 = \omega(\sqrt{n \log q \log m})$.

Useful Lemmata.

Lemma 2.19 (tail and truncation of $\mathcal{D}_{\mathbb{Z},\gamma}$). There exists $B_0 \in \Theta(\sqrt{\lambda})$ such that

$$\Pr\Big[x \leftarrow \mathcal{D}_{\mathbb{Z},\gamma} : |x| > \gamma B_0(\lambda)\Big] \leq 2^{-\lambda} \quad \text{for all} \quad \gamma \geq 1 \text{ and } \lambda \in \mathbb{N}.$$

Lemma 2.20 (Smudging Lemma [WWW22]). Let λ be a security parameter. Take any $a \in \mathbb{Z}$ where $|a| \leq B$. Suppose $\gamma \geq B\lambda^{\omega(1)}$. Then the statistical distance between the distributions $\{z : z \leftarrow \mathcal{D}_{\mathbb{Z},\gamma}\}$ and $\{z + a : z \leftarrow \mathcal{D}_{\mathbb{Z},\gamma}\}$ is $\mathsf{negl}(\lambda)$.

Lemma 2.21 (Leftover Hash Lemma). Fix some $n, m, q \in \mathbb{N}$. The leftover hash lemma states that if $m \geq 2n \log q$, then for $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}$, $\mathbf{x} \leftarrow \{0,1\}^m$ and $\mathbf{y} \leftarrow \mathbb{Z}_q^n$ the statistical distance between $(\mathbf{A}, \mathbf{A}\cdot\mathbf{x})$ and $(\mathbf{A}, \mathbf{y})$ is negligible. More concretely, it is bounded by $q^n\sqrt{2^{1-m}}$.

2.6.1 Hardness Assumptions

Assumption 2.22 (The LWE Assumption). Let $n = n(\lambda)$, $m = m(\lambda)$, and $q = q(\lambda) > 2$ be integers and $\chi = \chi(\lambda)$ be a distribution over $\mathbb{Z}_q$. We say that the $\mathrm{LWE}(n, m, q, \chi)$ hardness assumption holds if for any PPT adversary $\mathcal{A}$ we have

$$|\Pr[\mathcal{A}(\mathbf{A}, \mathbf{s}^\intercal\mathbf{A} + \mathbf{e}^\intercal) \to 1] - \Pr[\mathcal{A}(\mathbf{A}, \mathbf{v}^\intercal) \to 1]| \leq \mathsf{negl}(\lambda)$$

where the probability is taken over the choice of the random coins by the adversary $\mathcal{A}$ and $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}$, $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, $\mathbf{e} \leftarrow \chi^m$, and $\mathbf{v} \leftarrow \mathbb{Z}_q^m$. We also say that $\mathrm{LWE}(n, m, q, \chi)$ problem is subexponentially hard if the above probability is bounded by $2^{-n^\epsilon} \cdot \mathsf{negl}(\lambda)$ for some constant $0 < \epsilon < 1$ for all PPT $\mathcal{A}$.

As shown by previous works [Reg09, BLP+13], if we set $\chi = \mathsf{SampZ}(\gamma)$, the $\mathrm{LWE}(n, m, q, \chi)$ problem is as hard as solving worst case lattice problems such as gapSVP and SIVP with approximation factor $\mathrm{poly}(n) \cdot (q/\gamma)$ for some $\mathrm{poly}(n)$. Since the best known algorithms for 2^k-approximation of gapSVP and SIVP run in time $2^{\tilde{O}(n/k)}$, it follows that the above $\mathrm{LWE}(n, m, q, \chi)$ with noise-to-modulus ratio 2^{-n^ϵ} is likely to be (subexponentially) hard for some constant ϵ.

Assumption 2.23 (Circular Small Secret LWE). [HLL23] Let n, m, m', q, χ, χ' be functions of λ and

$$\bar{\mathbf{A}}_{\mathsf{fhe}} \leftarrow \mathbb{Z}_q^{n\times m},\ \bar{\mathbf{A}}' \leftarrow \mathbb{Z}_q^{n\times m'},\ \mathbf{r} \leftarrow \mathcal{D}_{\mathbb{Z},\chi}^n,\ \mathbf{s} \leftarrow (\mathbf{r}^\intercal, -1)^\intercal,\ \ \mathbf{e}_{\mathsf{fhe}} \leftarrow \mathcal{D}_{\mathbb{Z},\chi}^m,\ \ \mathbf{e}' \leftarrow \mathcal{D}_{\mathbb{Z},\chi'}^{m'},$$

$$\mathbf{R} \leftarrow \{0,1\}^{m\times(n+1)\lceil\log_2 q\rceil m},\ \ \delta_{\mathsf{fhe}} \leftarrow \mathbb{Z}_q^m,\ \ \delta' \leftarrow \mathbb{Z}_q^{m'},\ \ \Delta \leftarrow \mathbb{Z}_q^{(n+1)\times(n+1)\lceil\log_2 q\rceil m}$$

The circular small-secret LWE assumption $\mathsf{csLWE}_{n,m,m',q,\chi,\chi'}$ states that

$$\left\{\left(1^\lambda, \begin{pmatrix}\bar{\mathbf{A}}_{\mathsf{fhe}}\\ \mathbf{r}^\intercal\bar{\mathbf{A}}_{\mathsf{fhe}} + \mathbf{e}_{\mathsf{fhe}}^\intercal\end{pmatrix}, \begin{pmatrix}\bar{\mathbf{A}}_{\mathsf{fhe}}\\ \mathbf{r}^\intercal\bar{\mathbf{A}}_{\mathsf{fhe}} + \mathbf{e}_{\mathsf{fhe}}^\intercal\end{pmatrix}\mathbf{R} - \mathsf{bits}(\mathbf{s})\otimes\mathbf{G}, \bar{\mathbf{A}}', \mathbf{r}^\intercal\bar{\mathbf{A}}' + (\mathbf{e}')^\intercal\right)\right\}_{\lambda\in\mathbb{N}}$$

$$\approx \left\{\left(1^\lambda, \begin{pmatrix}\bar{\mathbf{A}}_{\mathsf{fhe}}\\ \delta_{\mathsf{fhe}}^\intercal\end{pmatrix}, \quad \Delta, \quad \bar{\mathbf{A}}', \ (\delta_i)^\intercal\right)\right\}_{\lambda\in\mathbb{N}}$$

Assumption 2.24 (Evasive LWE). [Wee22, ARYY23] Let $n, m, t, m', q \in \mathbb{N}$ be parameters and λ be a security parameter. Let χ and χ' be parameters for Gaussian distributions. Let Samp be a PPT algorithm that outputs

$$\mathbf{S} \in \mathbb{Z}_q^{m'\times n}, \mathbf{P} \in \mathbb{Z}_q^{n\times t}, \mathsf{aux} \in \{0,1\}^*$$

on input 1^λ. For a PPT adversary Adv, we define the following advantage functions:

$$\mathcal{A}_{\mathsf{Adv}}^{\mathrm{PRE}}(\lambda) \stackrel{\mathrm{def}}{=} \Pr\big[\mathsf{Adv}_0(\mathbf{B}, \mathbf{SB} + \mathbf{E}, \mathbf{SP} + \mathbf{E}', \mathsf{aux}) = 1\big] - \Pr\big[\mathsf{Adv}_0(\mathbf{B}, \mathbf{C}_0, \mathbf{C}', \mathsf{aux}) = 1\big]$$

$$\mathcal{A}^{\mathrm{POST}}_{\mathsf{Adv}}(\lambda) \stackrel{\mathrm{def}}{=} \Pr[\mathsf{Adv}_1(\mathbf{B}, \mathbf{SB}+\mathbf{E}, \mathbf{K}, \mathsf{aux}) = 1] - \Pr[\mathsf{Adv}_1(\mathbf{B}, \mathbf{C}_0, \mathbf{K}, \mathsf{aux}) = 1]$$

where

$$(\mathbf{S}, \mathbf{P}, \mathsf{aux}) \leftarrow \mathsf{Samp}(1^\lambda),$$
$$\mathbf{B} \leftarrow \mathbb{Z}_q^{n\times m},$$
$$\mathbf{C}_0 \leftarrow \mathbb{Z}_q^{m'\times m}, \mathbf{C}' \leftarrow \mathbb{Z}_q^{m'\times t},$$
$$\mathbf{E} \leftarrow \mathcal{D}_{\mathbb{Z},\chi}^{m'\times m}, \mathbf{E}' \leftarrow \mathcal{D}_{\mathbb{Z},\chi'}^{m'\times t}$$
$$\mathbf{K} \leftarrow \mathbf{B}^{-1}(\mathbf{P}) \text{ with standard deviation } O(\sqrt{m\log(q)}).$$

We say that the *evasive* LWE (EvLWE) assumption holds if for every PPT Samp and Adv_1, there exists another PPT Adv_0 and a polynomial $Q(\cdot)$ such that

$$\mathcal{A}^{\mathrm{PRE}}_{\mathsf{Adv}_0}(\lambda) \geq \mathcal{A}^{\mathrm{POST}}_{\mathsf{Adv}_1}(\lambda)/Q(\lambda) - \mathsf{negl}(\lambda).$$

Remark 2.25. In the above definition, all the LWE error terms are chosen from the same distribution $D_{\mathbb{Z},\chi}$. However, in our security proof, we often consider the case where some of LWE error terms are chosen from $D_{\mathbb{Z},\chi}$ and others from $D_{\mathbb{Z},\chi'}$ with different $\chi \gg \chi'$. The evasive LWE assumption with such a mixed noise distribution is implied by the evasive LWE assumption with all LWE error terms being chosen from $D_{\mathbb{Z},\chi}$ as above definition, since if the precondition is satisfied for the latter case, that for the former case is also satisfied. To see this, it suffices to observe that we can convert the distribution from $D_{\mathbb{Z},\chi'}$ into that from $D_{\mathbb{Z},\chi}$ by adding extra Gaussian noise.

In the security proof of our constructions, we sometimes want to include information dependent on $\mathbf{S}$ into the auxiliary information. However, this makes the corresponding evasive LWE assumption stronger and not desirable. The following lemma from [ARYY23] allows us to do this without strengthening the assumption under certain conditions. In the lemma, we separate the auxiliary information into two parts aux_1 and aux_2, where aux_1 is typically the part dependent on $\mathbf{S}$. The lemma roughly says that if aux_1 is pseudorandom, then we can apply the evasive LWE with respect to a modified sampler whose aux_1 is replaced with a random string to derive the conclusion on postcondition distribution.

Lemma 2.26 (Lemma 3.4 in [ARYY23]). Let $n, m, t, m', q \in \mathbb{N}$ be parameters and λ be a security parameter. Let χ and χ' be Gaussian parameters. Let Samp be a PPT algorithm that outputs

$$\mathbf{S} \in \mathbb{Z}_q^{m'\times n}, \mathsf{aux} = (\mathsf{aux}_1, \mathsf{aux}_2) \in \mathcal{S} \times \{0,1\}^* \text{ and } \mathbf{P} \in \mathbb{Z}_q^{n\times t}$$

for some set $\mathcal{S}$. Furthermore, we assume that there exists a public deterministic poly-time algorithm $\mathsf{Reconstruct}$ that allows to derive $\mathbf{P}$ from aux_2, i.e. $\mathbf{P} = \mathsf{Reconstruct}(\mathsf{aux}_2)$.

We introduce the following advantage functions:

$$\mathcal{A}^{\mathrm{PRE}'}_{\mathsf{Adv}}(\lambda) \stackrel{\mathrm{def}}{=} \Pr\left[\mathsf{Adv}(\mathbf{B}, \mathbf{SB}+\mathbf{E}, \mathbf{SP}+\mathbf{E}', \mathsf{aux}_1, \mathsf{aux}_2) = 1\right] - \Pr\left[\mathsf{Adv}(\mathbf{B}, \mathbf{C}_0, \mathbf{C}', \mathbf{c}, \mathsf{aux}_2) = 1\right]$$

$$\mathcal{A}^{\mathrm{POST}'}_{\mathsf{Adv}}(\lambda) \stackrel{\mathrm{def}}{=} \Pr[\mathsf{Adv}(\mathbf{B}, \mathbf{SB}+\mathbf{E}, \mathbf{K}, \mathsf{aux}_1, \mathsf{aux}_2) = 1] - \Pr[\mathsf{Adv}(\mathbf{B}, \mathbf{C}_0, \mathbf{K}, \mathbf{c}, \mathsf{aux}_2) = 1]$$

where

$$(\mathbf{S}, \mathsf{aux} = (\mathsf{aux}_1, \mathsf{aux}_2), \mathbf{P}) \leftarrow \mathsf{Samp}(1^\lambda),$$
$$\mathbf{B} \leftarrow \mathbb{Z}_q^{n\times m}$$
$$\mathbf{C}_0 \leftarrow \mathbb{Z}_q^{m'\times m}, \mathbf{C}' \leftarrow \mathbb{Z}_q^{m'\times t}, \mathbf{c} \leftarrow \mathcal{S}$$
$$\mathbf{E} \leftarrow \mathcal{D}_{\mathbb{Z},\chi}^{m'\times m}, \mathbf{E}' \leftarrow \mathcal{D}_{\mathbb{Z},\chi}^{m'\times t}$$
$$\mathbf{K} \leftarrow \mathbf{B}^{-1}(\mathbf{P}) \text{ with standard deviation } O(\sqrt{m\log(q)}).$$

Then, under the evasive LWE (cited above in Assumption 2.24) with respect to Samp' that outputs $(\mathbf{S}, (\mathbf{c}, \mathsf{aux}_2), \mathbf{P})$ for random $\mathbf{c}$, if $\mathcal{A}^{\mathrm{PRE}'}_{\mathsf{Adv}}(\lambda)$ is negligible for any PPT adversary Adv, so is $\mathcal{A}^{\mathrm{POST}'}_{\mathsf{Adv}}(\lambda)$ for any PPT adversary Adv.

Assumption 2.27 (Tensor LWE). [Wee22] Let $n, m, q, \ell, Q \in \mathbb{N}$ be parameters and $\gamma, \chi > 0$ be Gaussian parameters. For all $\mathbf{x}_1, \cdots, \mathbf{x}_Q \in \{0,1\}^\ell$, we have

$$\mathbf{A}, \left\{\mathbf{s}^\intercal(\mathbf{I}_n \otimes \mathbf{r})(\mathbf{A} - \mathbf{x}_i^\intercal \otimes \mathbf{G}) + \mathbf{e}_i^\intercal, \mathbf{r}_i\right\}_{i\in[Q]} \approx_c \mathbf{A}, \left\{\mathbf{c}_i^\intercal, \mathbf{r}_i\right\}_{i\in[Q]}$$

where $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times \ell m}, \mathbf{s} \leftarrow \mathbb{Z}_q^{mn}, \mathbf{e}_i \leftarrow \mathcal{D}_{\mathbb{Z},\chi}^{\ell m}, \mathbf{r}_i \leftarrow \mathcal{D}_{\mathbb{Z},\gamma}^{m}, \mathbf{c}_i \leftarrow \mathbb{Z}_q^{\ell m}$.

2.7 GSW Homomorphic Encryption and Evaluation

We recall the format (without the distribution) of the (leveled fully) homomorphic encryption [GSW13] and the correctness property. We adapt the syntax from [HLL23].

Lemma 2.28. The leveled FHE scheme works as follows:

- The keys are

$$\text{(public)} \quad \mathbf{A}_{\mathsf{fhe}} = \begin{pmatrix} \bar{\mathbf{A}}_{\mathsf{fhe}} \\ \bar{\mathbf{s}}^\intercal \bar{\mathbf{A}}_{\mathsf{fhe}} + \mathbf{e}_{\mathsf{fhe}}^\intercal \end{pmatrix} \in \mathbb{Z}_q^{(n+1)\times m}, \quad \text{(secret)} \quad \mathbf{s}^\intercal = (\bar{\mathbf{s}}^\intercal, -1),$$

  where $\bar{\mathbf{s}} \in \mathbb{Z}^n$,$\bar{\mathbf{A}}_{\mathsf{fhe}} \in \mathbb{Z}_q^{n\times m}$, and $\mathbf{e}_{\mathsf{fhe}}^\intercal \in \mathbb{Z}^m$.

- A ciphertext of $x \in \{0,1\}$ is $\mathbf{X} = \mathbf{A}_{\mathsf{fhe}}\mathbf{R} - x\mathbf{G} \in \mathbb{Z}_q^{(n+1)\times m}$, where $\mathbf{R} \in \mathbb{Z}_q^{m\times m}$ is the encryption randomness. The decryption equation is

$$\mathbf{s}^\intercal \mathbf{X} = -\mathbf{e}_{\mathsf{fhe}}^\intercal \mathbf{R} - x\mathbf{s}^\intercal \mathbf{G} \in \mathbb{Z}_q^m,$$

  which can be used to extract x via multiplication by $\mathbf{G}^{-1}(\lfloor q/2 \rfloor \iota_{n+1})$.

- There is an efficient algorithm

$$\text{MakeHEvalCkt}(1^n, 1^m, q, C) = \mathsf{HEval}_C$$

  that takes as input n, m, q and a circuit $C : \{0,1\}^L \to \{0,1\}$ and outputs a circuit

$$\mathsf{HEval}_C(\mathbf{X}_1, \ldots, \mathbf{X}_L) = \mathbf{C}$$

  taking L ciphertexts as input and outputting a new ciphertext $\mathbf{C}$.

  - The depth of HEval_C is $dO(\log m \log\log q)$, where d is the depth of C.
  - Suppose $\mathbf{X}_\ell = \mathbf{A}_{\mathsf{fhe}}\mathbf{R}_\ell - \mathbf{x}[\ell]\mathbf{G}$ for $\ell \in [L]$ with $\mathbf{x} \in \{0,1\}^L$, then

$$\mathbf{C} = \mathbf{A}_{\mathsf{fhe}}\mathbf{R}_C - C(\mathbf{x})\mathbf{G},$$

    where $\left\|\mathbf{R}_C^\intercal\right\| \le (m+2)^d \max_{\ell\in[L]} \left\|\mathbf{R}_\ell^\intercal\right\|$.

Additionally, in the circular version, ciphertexts of $\mathsf{bits}(\mathbf{s})$ are published.

Lemma 2.29. (homomorphic evaluation for vector-valued functions [HLL23]) There is an efficient algorithm

$$\text{MakeVEvalCkt}(1^n, 1^m, q, C) = \mathsf{VEval}_C$$

that takes as input n, m, q and a vector-valued circuit $C : \{0,1\}^L \to \mathbb{Z}_q^{1\times m'}$ and outputs a circuit

$$\mathsf{VEval}_C(\mathbf{X}_1, ..., \mathbf{X}_L) = \mathbf{C},$$

taking L ciphertexts as input and outputting a new ciphertext $\mathbf{C}$ of different format.

- The depth of VEval_C is $d \cdot O(\log m \log\log q) + O(\log^2 \log q)$ for C of depth d.

- Suppose $\mathbf{X}_\ell = \mathbf{A}_{\mathsf{fhe}}\mathbf{R}_\ell - \mathbf{x}[\ell]\mathbf{G}$ for $\ell \in [L]$ with $\mathbf{x} \in \{0,1\}^L$, then

$$\mathbf{C} = \mathbf{A}_{\mathsf{fhe}}\mathbf{R}_C - \begin{pmatrix} \mathbf{0}_{n\times m'} \\ C(\mathbf{x}) \end{pmatrix} \in \mathbb{Z}_q^{(n+1)\times m'},$$

  where $\left\|\mathbf{R}_C^\intercal\right\| \le (m+2)^d \lceil \log q \rceil \max_{\ell\in[L]} \left\|\mathbf{R}_\ell^\intercal\right\|$. The new decryption equation is

$$\mathbf{s}^\intercal \mathbf{C} = -\mathbf{e}_{\mathsf{fhe}}^\intercal \mathbf{R}_C + C(\mathbf{x}) \in \mathbb{Z}_q^{1\times m'}.$$

2.8 BGG+ Homomorphic Evaluation Procedures

In this section we describe the properties of the attribute encoding and its homomorphic evaluation. We adapt the syntax from [HLL23].

- For L-bit input, the public parameter is $\mathbf{A}_{\mathsf{att}} \in \mathbb{Z}_q^{(n+1)\times(L+1)m}$.
- The encoding of $\mathbf{x} \in \{0,1\}^L$ is

$$\mathbf{s}^\intercal(\mathbf{A}_{\mathsf{att}} - (1, \mathbf{x}^\intercal) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{att}}^\intercal,$$

  where $\mathbf{s}^\intercal = (\bar{\mathbf{s}}^\intercal, -1)$ with $\bar{\mathbf{s}} \in \mathbb{Z}^n$ and $\mathbf{e}_{\mathsf{att}}^\intercal \in \mathbb{Z}^{(L+1)m}$.
- There are efficient deterministic algorithms [BGG+14]

$$\mathsf{EvalC}(\mathbf{A}_{\mathsf{att}}, C) = \mathbf{H}_C \quad \text{and} \quad \mathsf{EvalCX}(\mathbf{A}_{\mathsf{att}}, C, \mathbf{x}) = \mathbf{H}_{C,\mathbf{x}}$$

  that take as input $\mathbf{A}_{\mathsf{att}}$, a circuit $C : \{0,1\}^L \to \{0,1\}$, and (for EvalCX) some $\mathbf{x} \in \{0,1\}^L$, and output some matrix in $\mathbb{Z}^{(L+1)m\times m}$.
  - Suppose C is of depth d, then $\left\|\mathbf{H}_C^\intercal\right\|, \|\mathbf{H}_{C,\mathbf{x}}^\intercal\| \le (m+2)^d$.
  - They satisfy encoding homomorphism, $(\mathbf{A}_{\mathsf{att}} - (1,\mathbf{x}^\intercal)\otimes\mathbf{G})\mathbf{H}_{C,\mathbf{x}} = \mathbf{A}_{\mathsf{att}}\mathbf{H}_C - C(\mathbf{x})\mathbf{G}$.
- There are efficient deterministic algorithms [BTVW17]

$$\mathsf{MEvalC}(\mathbf{A}_{\mathsf{att}}, C) = \mathbf{H}_C \quad \text{and} \quad \mathsf{MEvalCX}(\mathbf{A}_{\mathsf{att}}, C, \mathbf{x}) = \mathbf{H}_{C,x}$$

  that take as input $\mathbf{A}_{\mathsf{att}}$, a matrix-valued circuit $C : \{0,1\}^L \to \mathbb{Z}_q^{n+1\times m'}$, and (for $\mathsf{MEvalCX}$) some $\mathbf{x} \in \{0,1\}^L$, and output some matrix in $\mathbb{Z}^{(L+1)m\times m'}$.
  - Suppose C is of depth d, then $\left\|\mathbf{H}_C^\intercal\right\|, \|\mathbf{H}_{C,\mathbf{x}}^\intercal\| \le (m+2)^d \lceil \log q \rceil$.
  - The matrix encoding homomorphism is $(\mathbf{A}_{\mathsf{att}} - (1,\mathbf{x}^\intercal)\otimes\mathbf{G})\mathbf{H}_{C,\mathbf{x}} = \mathbf{A}_{\mathsf{att}}\mathbf{H}_C - C(\mathbf{x})$.

Dual-Use Technique and Extension. In [BTVW17], the attribute encoded with secret $\mathbf{s}^\intercal$ is FHE ciphertexts under key $\mathbf{s}^\intercal$ (the same, "dual-use") and the circuit being $\mathsf{MEvalCX}$'ed is some HEval_C. This leads to automatic decryption. Let C be a circuit with Boolean output, $\mathbf{x}$ an input, $\mathbf{X}$ a bunch of FHE ciphertexts of $\mathsf{bits}(\mathbf{x})$ under $\mathbf{s}^\intercal$, and $\mathbf{e}_{\mathsf{att}}$, $\mathbf{e}'$, $\mathbf{e}''$ some unspecified noises, then

$$\begin{aligned}
&\mathbf{s}^\intercal(\mathbf{A}_{\mathsf{att}} - (1, \mathsf{bits}(\mathbf{X})) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{att}}^\intercal \cdot \mathbf{H}_{\mathsf{HEval}_C,\mathbf{X}} \\
&= \mathbf{s}^\intercal \mathbf{A}_{\mathsf{att}} \mathbf{H}_{\mathsf{HEval}_C} - \mathbf{s}^\intercal \mathsf{HEval}_C(\mathbf{X}) + (\mathbf{e}')^\intercal \quad (\mathsf{MEvalCX}) \\
&= \mathbf{s}^\intercal \mathbf{A}_{\mathsf{att}} \mathbf{H}_{\mathsf{HEval}_C} - \mathbf{s}^\intercal C(\mathbf{x})\mathbf{G} + (\mathbf{e}'')^\intercal \quad (\mathsf{HEval\ decryption}) \\
&= \mathbf{s}^\intercal(\mathbf{A}_{\mathsf{att}} \mathbf{H}_{\mathsf{HEval}_C} - C(\mathbf{x})\mathbf{G}) + (\mathbf{e}'')^\intercal.
\end{aligned}$$

To extend the dual-use technique to vector-valued circuits, let the codomain of C be $\mathbb{Z}_q^{1\times m'}$, then VEval_C is $\mathbb{Z}_q^{(n+1)\times m'}$-valued and

$$\begin{aligned}
&\mathbf{s}^\intercal(\mathbf{A}_{\mathsf{att}} - (1, \mathsf{bits}(\mathbf{X})) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{att}}^\intercal \cdot \mathbf{H}_{\mathsf{VEval}_C, \mathbf{X}} \\
&= \mathbf{s}^\intercal \mathbf{A}_{\mathsf{att}} \mathsf{VEval}_C - \mathbf{s}^\intercal \mathsf{VEval}_C(\mathbf{X}) + (\mathbf{e}')^\intercal \quad (\mathsf{MEvalCX}) \\
&= \mathbf{s}^\intercal \mathbf{A}_{\mathsf{att}} \mathsf{VEval}_C - C(\mathbf{x}) + (\mathbf{e}'')^\intercal \quad (\mathsf{VEval\ decryption}).
\end{aligned}$$

Extension to circular encryption means setting $\mathbf{x} = \mathbf{s}$, for which we say $\mathbf{S}$, $\mathbf{A}_{\mathsf{circ}}$ in place of $\mathbf{X}$, $\mathbf{A}_{\mathsf{att}}$.

3 Bootstrapping Randomized Homomorphic Evaluation

In this section, we show how to achieve unbounded homomorphism for a randomized attribute encoding of the form

$$\mathbf{c}_{\mathsf{att}}^\intercal = \mathbf{s}^\intercal(\mathbf{I}_{n+1} \otimes \mathbf{r})(\mathbf{A}_{\mathsf{att}} - (1, \mathbf{x}^\intercal) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{att}}^\intercal = \mathbf{s}_{\mathbf{r}}^\intercal(\mathbf{A}_{\mathsf{att}} - (1, \mathbf{x}^\intercal) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{att}}^\intercal$$

where $\mathbf{s} \in \mathbb{Z}^{m(n+1)}, \mathbf{r} \in \mathbb{Z}^m, \mathbf{A}_{\mathsf{att}} \in \mathbb{Z}_q^{(n+1)\times m}, x \in \{0,1\}^{\mathrm{L}}, \mathbf{e}_{\mathsf{att}}^\intercal \in \mathbb{Z}^m$. We let $\|\mathbf{e}_{\mathsf{att}}\|$ and $\|(\mathbf{s}^\intercal(\mathbf{I}_{n+1} \otimes \mathbf{r}))^\intercal\|$ be bounded by B. Here B denotes the bound of removable noise. We achieve unbounded homomorphism for the above randomized attribute encoding using the following steps.

1. **Noise removal for randomized encoding.** First we transform $\mathbf{c}_{\mathsf{att}}^\intercal = \mathbf{s}_{\mathbf{r}}^\intercal(\mathbf{A}_{\mathsf{att}} - (1, \mathbf{x}^\intercal) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{att}}^\intercal$ into a noiseless encoding using the procedure $\mathsf{RemoveNoise}(\cdot)$ from HLL to achieve $\mathsf{RemoveNoise}(\mathbf{c}_{\mathsf{att}}^\intercal) = \mathsf{RndPad}_{\mathbf{A}}(\mathbf{s}_{\mathbf{r}}) - C(\mathbf{x})\mathbf{s}_{\mathbf{r}}^\intercal\mathbf{G}$ where $\mathsf{RndPad}_{\mathbf{A}}(\mathbf{s}_{\mathbf{r}}) = \mathsf{RemoveNoise}(\mathbf{s}_{\mathbf{r}}^\intercal\mathbf{A})$.
2. **Structure restoration.** Here we transform the noiseless encoding achieved above to an attribute encoding with a smaller noise, following the blueprint of [HLL23].

Before describing the above steps in detail, we first state a few tools, theorems and lemmas that will be useful later.

3.1 Preparation

Rounding Function. Let $\mathsf{Rnd} : \mathbb{Z}_q \to \mathbb{Z}_q$ be a rounding function defined as $\mathsf{Rnd}(x) = (q/p)\lfloor p/q \cdot x \rceil$, where p divides q. We make the following observations for the function Rnd.

Claim 3.1. For $x \in \mathbb{Z}_q$, $\mathsf{Rnd}(x) = x + e_{\mathsf{Rnd}}$ where $|e_{\mathsf{Rnd}}| \leq \frac{q}{2p}$.

Proof. The rounding function $\mathrm{Rnd}(x)$ first rounds $\frac{p}{q} \cdot x$ to the nearest integer, then scales it up to $\mathbb{Z}_q$ by multiplying with $\frac{q}{p}$. Let m be the nearest integer to $\frac{p}{q} \cdot x$. Then, $\mathrm{Rnd}(x) = m \cdot \frac{q}{p}$. The error e_{Rnd} introduced by the rounding process is defined as $e_{\mathrm{Rnd}} = \mathrm{Rnd}(x) - x$.
When $\frac{p}{q} \cdot x$ is rounded to m, the error introduced is at most $\frac{1}{2}$. Scaling this error up to $\mathbb{Z}_q$ by multiplying with $\frac{q}{p}$, the maximum error becomes $\frac{q}{p} \cdot \frac{1}{2} = \frac{q}{2p}$. Therefore, we have $|e_{\mathrm{Rnd}}| \leq \frac{q}{2p}$. □

Claim 3.2. For random $x, x' \in \mathbb{Z}_q$ such that $|x - x'|$ is negligibly smaller than $\frac{q}{p}$, we have $\mathsf{Rnd}(x) - \mathsf{Rnd}(x') = 0$ with all but negligible probability.

Proof. Let ϵ represent the negligible quantity, such that $|x - x'| \leq \epsilon \cdot \frac{q}{p}$. The function Rnd operates by scaling x to the $\mathbb{Z}_p$ space, rounding it, and then scaling back to $\mathbb{Z}_q$. This creates intervals in $\mathbb{Z}_q$ of size $\frac{q}{p}$ that map to each integer in $\mathbb{Z}_p$. The probability that x and x' fall into different rounding intervals is related to how close x is to the boundary of a rounding interval. If x and x' are within $\epsilon \times \frac{q}{p}$ of each other, for a small ϵ, and they straddle a rounding boundary, then they round to different values.
Given a uniform distribution of x and x' over $\mathbb{Z}_q$, the probability of a number falling within $\epsilon \times \frac{q}{p}$ of a rounding

boundary is approximately 2ϵ because there are two boundaries (upper and lower) for each interval, and each boundary has a "risk zone" of $\epsilon \times \frac{q}{p}$ around it.
Therefore, $\Pr[\mathsf{Rnd}(x) \neq \mathsf{Rnd}(x') \mid |x - x'| \leq \epsilon \cdot (q/p)] \leq 2\epsilon$, which is negligible. □

Theorem 3.3 (Noise Removal: [HLL23] Construction 1). There exists a deterministic procedure $\mathsf{RemoveNoise}(\cdot)$ such that for an input $\mathbf{u}^\intercal = \mathbf{s}^\intercal(\mathbf{A} - x\mathbf{G}) + \mathbf{e} \in \mathbb{Z}_q^{1\times m}$, where $\mathbf{A} \in \mathbb{Z}_q^{n+1\times m}, x \in \{0,1\}, \|\mathbf{s}\| \leq B, \|\mathbf{e}\| \leq B$,

$$\mathsf{RemoveNoise}(\mathbf{u}^\intercal) = \mathsf{RndPad}_{\mathbf{A}}(\mathbf{s}) - x\mathbf{s}^\intercal\mathbf{G} \in \mathbb{Z}_q^{1\times m}$$

where $\mathsf{RndPad}_{\mathbf{A}}(\mathbf{s}) = \mathsf{RemoveNoise}(\mathbf{s}^\intercal\mathbf{A})$.

Lemma 3.4. A canonical Boolean circuit of $\mathsf{RndPad}_{\mathbf{A}}(\cdot)$ is of depth $O(\log n \log\log q)$ and can be efficiently generated from $\mathbf{A} \in \mathbb{Z}_q^{(n+1)\times m}$.

Theorem 3.5 (Bootstrapping: [HLL23] Theorem 12). It works as follows:

- The secret is $\mathbf{s} = (\bar{\mathbf{s}}^\intercal, -1)^\intercal \in \mathbb{Z}^{n+1}$ with $\mathsf{bits}(\mathbf{s}) \in \{0,1\}^{(n+1)\lceil\log q\rceil}$. The circular ciphertext is

$$\mathbf{S} = \begin{pmatrix} \bar{\mathbf{A}}_{\mathsf{fhe}} \\ \bar{\mathbf{s}}^\intercal\bar{\mathbf{A}}_{\mathsf{fhe}} + \mathbf{e}_{\mathsf{fhe}}^\intercal \end{pmatrix} (\mathbf{R}_1, \cdots, \mathbf{R}_{(n+1)\lceil\log q\rceil}) - \mathsf{bits}(\mathbf{s}) \otimes \mathbf{G} \in \mathbb{Z}_q^{(n+1)\times m(n+1)\lceil\log q\rceil}$$

  where $\bar{\mathbf{A}}_{\mathsf{fhe}} \in \mathbb{Z}_q^{(n+1)\times m}$, $\mathbf{e}_{\mathsf{fhe}}^\intercal \in \mathbb{Z}^m$, and $\mathbf{R}_i \in \mathbb{Z}_q^{m\times m}$ for $1 \leq i \leq (n+1)\lceil\log q\rceil$.
  Let $\mathrm{L_S} = m(n+1)^2\lceil\log q\rceil^2$, so that $\mathsf{bits}(\mathbf{S}) \in \{0,1\}^{1\times \mathrm{L_S}}$.

- The circular encoding is $\mathbf{s}^\intercal(\mathbf{A}_{\mathsf{circ}} - (1, \mathsf{bits}(\mathbf{S})) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{circ}}^\intercal$ where $\mathbf{A}_{\mathsf{circ}} \in \mathbb{Z}_q^{(n+1)\times(\mathrm{L_S}+1)m}$ and $\mathbf{e}_{\mathsf{circ}} \in \mathbb{Z}^{(\mathrm{L_S}+1)m}$.

- There are efficient deterministic algorithms

$$\mathsf{EvalRndPad}(\mathbf{A}_{\mathsf{circ}}, \mathbf{A}) = \mathrm{H}_{\mathbf{A}}^{\mathsf{RndPad}} \quad \text{and} \quad \mathsf{EvalRndPadS}(\mathbf{A}_{\mathsf{circ}}, \mathbf{A}, \mathbf{S}) = \mathrm{H}_{\mathbf{A},\mathbf{S}}^{\mathsf{RndPad}}$$

  such that

$$\left\|(\mathrm{H}_{\mathbf{A}}^{\mathsf{RndPad}})^\intercal\right\|, \left\|(\mathrm{H}_{\mathbf{A},\mathbf{S}}^{\mathsf{RndPad}})^\intercal\right\| \leq 2^{O(\log^5\lambda)}.$$

  Moreover, when $\mathbf{S}$ is indeed of the correct form for $\mathsf{RndPad}_{\mathbf{A}}(\cdot)$ defined in Theorem 3.3,

$$\begin{aligned}&\mathbf{s}^\intercal(\mathbf{A}_{\mathsf{circ}} - (1, \mathsf{bits}(\mathbf{S})) \otimes \mathbf{G})\mathrm{H}_{\mathbf{A},\mathbf{S}}^{\mathsf{RndPad}} \\ &= \mathbf{s}^\intercal\mathbf{A}_{\mathsf{circ}}\mathrm{H}_{\mathbf{A}}^{\mathsf{RndPad}} - \mathsf{RndPad}_{\mathbf{A}}(\mathbf{s}) + \mathbf{e}_{\mathsf{fhe}}^\intercal\mathbf{R}_{\mathsf{RndPad}_{\mathbf{A}}}\end{aligned}$$

  where $\left\|\mathbf{R}_{\mathsf{RndPad}_{\mathbf{A}}}^\intercal\right\| \leq 2^{O(\log^4\lambda)}$.

Theorem 3.6 (Unbounded Homomorphic Evaluation: [HLL23] Construction 2). Let $C : \{0,1\}^L \to \{0,1\}$ and $\mathbf{x} \in \{0,1\}^L$. Suppose

$$\mathbf{s} = (\bar{\mathbf{s}}^\intercal, -1)^\intercal, \ , \mathbf{R} \in \{0,1\}^{m\times m(n+1)\log q}, \ \mathbf{S} = \begin{pmatrix} \bar{\mathbf{A}}_{\mathsf{fhe}} \\ \bar{\mathbf{s}}^\intercal\bar{\mathbf{A}}_{\mathsf{fhe}} + \mathbf{e}_{\mathsf{fhe}}^\intercal \end{pmatrix} \mathbf{R} - \mathsf{bits}(\mathbf{s}) \otimes \mathbf{G}$$

$$\mathbf{c}_{\mathsf{att}}^\intercal = \mathbf{s}^\intercal(\mathbf{A}_{\mathsf{att}} + (1, \mathbf{x}) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{att}}^\intercal, \ \mathbf{c}_{\mathsf{circ}}^\intercal = \mathbf{s}^\intercal(\mathbf{A}_{\mathsf{circ}} - (1, \mathsf{bits}(\mathbf{S})) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{circ}}^\intercal,$$

where $\|\mathbf{s}\| \leq B, \|\mathbf{e}_{\mathsf{att}}\|, \left\|\mathbf{e}_{\mathsf{fhe}}^\intercal\right\|, \|\mathbf{e}_{\mathsf{circ}}\| \leq 2^{-\lambda}B$.

There are two efficient algorithms

$$\mathsf{UEvalC}(\mathbf{A}_{\mathsf{att}}, \mathbf{A}_{\mathsf{circ}}, C) = \mathbf{A}_C, \qquad \mathsf{UEvalCX}(\mathbf{A}_{\mathsf{att}}, \mathsf{ct}_{\mathsf{att}}^\intercal, \mathbf{A}_{\mathsf{circ}}, \mathsf{ct}_{\mathsf{circ}}^\intercal, C, \mathbf{x}, \mathbf{S}) = \mathbf{c}_{C,\mathbf{x}}^\intercal$$

satisfying, with all but negligible probability, $\mathbf{c}_{C,\mathbf{x}}^\intercal = \mathbf{s}^\intercal(\mathbf{A}_C - C(\mathbf{x})\mathbf{G}) + \mathbf{e}_{C,\mathbf{x}}^\intercal$ and

$$\|\mathbf{e}_{C,\mathbf{x}}\| \leq (\|\mathbf{e}_{\mathsf{fhe}}\| + \|\mathbf{e}_{\mathsf{circ}}\|) \cdot 2^{\mathrm{poly}(\log q)} \leq B.$$

3.2 Noise Removal for Randomized Encoding

Here we use the $\mathsf{RemoveNoise}(\cdot)$ procedure from Theorem 3.3 to remove noise from a randomized attribute encoding $\mathbf{c}_{\mathsf{att}}^\intercal = \mathbf{s}_{\mathbf{r}}^\intercal(\mathbf{A} - C(\mathbf{x})\mathbf{G}) + \mathbf{e}^\intercal \in \mathbb{Z}_q^{1\times m}$ with $\|\mathbf{s}_{\mathbf{r}}\|, \|\mathbf{e}\| \le B$. Here we have $\mathbf{s}_{\mathbf{r}}^\intercal = \mathbf{s}^\intercal(\mathbf{I}_{n+1} \otimes \mathbf{r})$, $\mathbf{s} \in \mathbb{Z}^{m(n+1)}$, $\mathbf{r} \in \mathbb{Z}^m$, $\mathbf{A} \in \mathbb{Z}_q^{(n+1)\times m}$, and $\mathbf{e} \in \mathbb{Z}^m$.
We have the following using Theorem 3.3.

$$\mathsf{RemoveNoise}(\mathbf{s}_{\mathbf{r}}^\intercal(\mathbf{A} - C(\mathbf{x})\mathbf{G}) + \mathbf{e}^\intercal) = \mathsf{RndPad}_{\mathbf{A}}(\mathbf{s}_{\mathbf{r}}) - C(\mathbf{x})\mathbf{s}_{\mathbf{r}}^\intercal\mathbf{G}.$$

3.3 Randomized Bootstrapping a.k.a Structure Restoration

To support unbounded evaluation, we transform the noiseless encoding achieved in Section 3.2 back to an attribute encoding with smaller noise, following the blueprint of [HLL23]. In particular, we want to compute $\mathbf{s}_{\mathbf{r}}^\intercal\mathbf{A}'_C - \mathsf{RndPad}_{\mathbf{A}}(\mathbf{s}_{\mathbf{r}}) + (\mathbf{e}')^\intercal$ for some publicly computable matrix $\mathbf{A}'_C$ and a small noise $\mathbf{e}'$ whose norm is within the bound of removable noise. Note that, as discussed in the technical overview, this cannot be directly computed during the key generation or encryption process as it requires the knowledge of both the keygen randomness $\mathbf{r}$ and the encryption randomness $\mathbf{s}$. We divide the structure restoration into two parts.

Step 1: Computing Advice for HLL. We define a new algorithm $\mathsf{ComputeAdvice}_{\mathbf{r}}$ that enables us to compute an extra *noisy* FHE circular encryption $\mathbf{S}'_{\mathbf{r}} = \mathsf{hct}_{\mathbf{s}_{\mathbf{r}}}(\mathbf{s}_{\mathbf{r}}) + \mathbf{err}_{\mathbf{S}}$ using key $\mathbf{s}_{\mathbf{r}}$ and a circular encoding $\mathbf{E}'_{\mathbf{r}} = \mathbf{s}_{\mathbf{r}}^\intercal(\mathbf{A}_{\mathsf{circ}} - (1, \mathsf{bits}(\mathbf{S}'_{\mathbf{r}})) \otimes \mathbf{G}) + \mathbf{e}_{\mathbf{E}}^\intercal$ where $\mathbf{s}_{\mathbf{r}}^\intercal = \mathbf{s}^\intercal(\mathbf{I}_{n+1} \otimes \mathbf{r}) = (\bar{\mathbf{s}}_{\mathbf{r}}^\intercal, -1)$[3] is the FHE secret key.

Step 2: HLL Bootstrapping. We use $\mathbf{S}'_{\mathbf{r}}$ and $\mathbf{E}'_{\mathbf{r}}$ obtained above to compute $\mathbf{s}_{\mathbf{r}}^\intercal\mathbf{A} - \mathsf{RndPad}_{\mathbf{A}}(\mathbf{s}_{\mathbf{r}})$ similarly to HLL.

Notations. We set $m_{\mathrm{T}} = m(m(n+1)\lceil\log q\rceil + \lambda)$, $\mathrm{L}_{\mathrm{T}} = m(n+1)(m(n+1)\lceil\log q\rceil + \lambda)\lceil\log q\rceil$, $m_{\mathrm{S}} = m(n+1)\lceil\log q\rceil$, $\mathrm{L}_{\mathrm{S}} = m(n+1)^2\lceil\log q\rceil^2$, and $\ell = \mathrm{L}_{\mathrm{S}}(1 + m\log q)$.

Ingredients. We require the following tools.

1. A rounding function $\mathsf{Rnd} : \mathbb{Z}_q \to \mathbb{Z}_q$ where $\mathsf{Rnd}(x) = \frac{q}{p}\left\lfloor\frac{px}{q}\right\rceil$, where p is an integer that divides q.
2. Three pseudorandom functions: $\mathsf{PRF}_1 : \{0,1\}^\lambda \times \mathbb{Z}^m \to [-\sigma', \sigma']^{1\times m}$, $\mathsf{PRF}_2 : \{0,1\}^\lambda \times \mathbb{Z}^m \to \{0,1\}^{m\times m_{\mathrm{S}}}$ and $\mathsf{PRF}_3 : \{0,1\}^\lambda \times \mathbb{Z}^m \to [-\sigma', \sigma']^{1\times(\mathrm{L}_{\mathrm{S}}+1)m}$.

Next, we describe our $\mathsf{ComputeAdvice}_{\mathbf{r}}$ procedure.

Step 1: Procedure $\mathsf{ComputeAdvice}_{\mathbf{r}}(\bar{\mathbf{A}}_{\mathsf{fhe}}, \mathbf{A}_{\mathsf{circ}}, \mathbf{A}_{\mathsf{path}}, \mathbf{T}, \mathbf{D}, \mathbf{c}_1^\intercal)$. The algorithm $\mathsf{ComputeAdvice}_{\mathbf{r}}$, with $\mathbf{r} \in \mathbb{Z}_q^m$ hardwired, has the following functionality.
Input: It takes as input $(\bar{\mathbf{A}}_{\mathsf{fhe}}, \mathbf{A}_{\mathsf{circ}}, \mathbf{A}_{\mathsf{path}}, \mathbf{T}, \mathbf{D}, \mathbf{c}_1^\intercal)$, where $\bar{\mathbf{A}}_{\mathsf{fhe}} \in \mathbb{Z}_q^{n\times m}$, $\mathbf{A}_{\mathsf{circ}} \in \mathbb{Z}_q^{(n+1)\times(\mathrm{L}_{\mathrm{S}}+1)m}$,

$$\mathbf{A}_{\mathsf{path}} \in \mathbb{Z}_q^{(n+1)\times(\mathrm{L}_{\mathrm{T}}+1)m},\ \mathbf{T} = \begin{pmatrix} \bar{\mathbf{A}}'_{\mathsf{fhe}} \\ \bar{\mathbf{t}}^\intercal\bar{\mathbf{A}}'_{\mathsf{fhe}} + (\mathbf{e}'_{\mathsf{fhe}})^\intercal \end{pmatrix}\mathbf{R} - \mathsf{bits}(\mathbf{s}, \mathsf{sd}) \otimes \mathbf{G}, \in \mathbb{Z}_q^{(n+1)\times m_{\mathrm{T}}}$$

$$\mathbf{D} = \mathbf{t}^\intercal(\mathbf{A}_{\mathsf{path}} - (1, \mathsf{bits}(\mathbf{T})) \otimes \mathbf{G}) + \mathbf{e}_{\mathbf{D}}^\intercal \in \mathbb{Z}_q^{(\mathrm{L}_{\mathrm{T}}+1)m},\ \mathbf{c}_1^\intercal = \mathbf{t}^\intercal(\mathbf{A}_{\mathbf{r}}) + \mathbf{e}_1^\intercal \in \mathbb{Z}_{q'}^\ell$$

for $\mathbf{t} = (\bar{\mathbf{t}}^\intercal, -1)^\intercal \in \mathbb{Z}^{n+1}$, $\bar{\mathbf{A}}'_{\mathsf{fhe}} \in \mathbb{Z}_q^{(n+1)\times m}$, $\mathbf{e}'_{\mathsf{fhe}} \in \mathbb{Z}^m$, $\mathbf{R} \in \{0,1\}^{m\times m_{\mathrm{T}}}$, and $\mathbf{A}_{\mathbf{r}} = \mathbf{A}_{\mathsf{path}}\mathbf{H}_{\mathrm{F}} \in \mathbb{Z}_q^{(n+1)\times\ell}$. Here $\mathbf{H}_{\mathrm{F}}$ is a short matrix computed w.r.t. the function F defined in Figure 1.

[3] We sample $\mathbf{s}$ and $\mathbf{r}$ in such a way that $\mathbf{s}_{\mathbf{r}}^\intercal = (\bar{\mathbf{s}}_{\mathbf{r}}^\intercal, -1)$ holds.

Output: It outputs $(\mathbf{S}'_{\mathbf{r}} \in \mathbb{Z}_q^{(n+1)\times m_S}, \mathbf{E}'_{\mathbf{r}} \in \mathbb{Z}_q^{1\times(L_S+1)m})$, where

$$\mathbf{S}'_{\mathbf{r}} = \begin{pmatrix} \bar{\mathbf{A}}_{\mathsf{fhe}} \\ \bar{\mathbf{s}}_{\mathbf{r}}^{\mathsf{T}}\bar{\mathbf{A}}_{\mathsf{fhe}} + \mathbf{e}_{\mathsf{fhe}}^{\mathsf{T}} \end{pmatrix} \mathbf{R} - \mathsf{bits}(\mathbf{s}_{\mathbf{r}}) \otimes \mathbf{G} + \mathbf{err}_{\mathbf{S}}, \quad \mathbf{E}'_{\mathbf{r}} = \mathbf{s}_{\mathbf{r}}^{\mathsf{T}}(\mathbf{A}_{\mathsf{circ}} - (1, \mathsf{bits}(\mathbf{S}'_{\mathbf{r}})) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{circ}}^{\mathsf{T}}$$

for $\mathbf{s}_{\mathbf{r}}^{\mathsf{T}} = (\bar{\mathbf{s}}_{\mathbf{r}}^{\mathsf{T}}, -1) \in \mathbb{Z}^{n+1}$, $\mathbf{e}_{\mathsf{fhe}} \in \mathbb{Z}^m$, $\mathbf{err}_{\mathbf{S}}$[4] $\in \mathbb{Z}^{(n+1)\times m_S}$, and $\mathbf{e}_{\mathsf{circ}} \in \mathbb{Z}_q^{1\times(L_S+1)m}$.

If the terms $\mathbf{e}'_{\mathsf{fhe}}, \mathbf{e}_{\mathbf{D}}$ and $\mathbf{e}_1$ are short, then $\mathbf{e}_{\mathsf{fhe}}$, $\mathbf{err}_{\mathbf{S}}$, and $\mathbf{e}_{\mathsf{circ}}$ are short. This will be formally proven in Lemma 3.7. The algorithm $\mathsf{ComputeAdvice}_{\mathbf{r}}$ works as follows.

1. *Defining function to evaluate.* Define function $\mathrm{F} = \mathrm{F}_{\mathbf{r}}(\cdot,\cdot)$ as in Figure 1. Using the fact that the computation of modular inner product and the PRF is in NC_1, we analyse the depth of F as follows.
   Step 1 and 3 can be implemented by a circuit of depth $O(\log m \log\log q + \log n \log\log q)$. Step 2 can be implemented by a circuit of depth $O(\log n \log\log q) + O(\log(m \log q)) \le O(\log m \log\log q + \log n \log\log q)$, where the first component is w.r.t. inner product and second for the addition. Similarly step 4 can be implemented by a circuit of depth $O(\log n \log\log q)$. Thus the entire computation can be implemented with depth $d_{\mathrm{F}} = O(\log m \log\log q + \log n \log\log q)$.

2. *Defining circuits for homomorphic evaluation.* Here we define homomorphic evaluation circuits for function F.

   – Define $\mathsf{VEval}_{\mathrm{F}} = \mathsf{MakeVEvalCkt}(n, m, q, \mathrm{F}) \in \mathbb{Z}_q^{n+1\times\ell}$. From Lemma 2.29, the depth of $\mathsf{VEval}_{\mathrm{F}}$ is

   $$(d_{\mathrm{F}} O(\log m \log\log q) + O(\log^2 \log q)) = O(\log^2 m \log^2 \log q + \log m \log n \log^2 \log q).$$

   – Compute $\mathbf{H}_{\mathrm{F}} = \mathsf{MEvalC}(\mathbf{A}_{\mathsf{path}}, \mathsf{VEval}_{\mathrm{F}})$, $\mathbf{H}_{\mathrm{F},\mathbf{T}} = \mathsf{MEvalCX}(\mathbf{A}_{\mathsf{path}}, \mathsf{VEval}_{\mathrm{F}}, \mathbf{T}) \in \mathbb{Z}^{(L_T+1)m\times\ell}$.

   Using the depth bound from Section 2.8, we have

   $$\begin{aligned} \|(\mathbf{H}_{\mathrm{F}})^{\mathsf{T}}\|, \|(\mathbf{H}_{\mathrm{F},\mathbf{T}})^{\mathsf{T}}\| &\le (m+2)^{\mathsf{d}_{\mathsf{VEval}_{\mathrm{F}}}} \lceil \log q \rceil \\ &= (m+2)^{O(\log^2 m \log^2 \log q + \log m \log n \log^2 \log q)} \lceil \log q \rceil \\ &\le 2^{\log^5 \lambda}. \end{aligned}$$

3. *Compute the circuit homomorphically on* $\mathbf{D}$. Here we use the homomorphic evaluation circuits to compute on $\mathbf{D}$.

   $$\begin{aligned} \mathbf{D} \cdot \mathbf{H}_{\mathrm{F},\mathbf{T}} &= (\mathbf{t}^{\mathsf{T}}(\mathbf{A}_{\mathsf{path}} - (1, \mathsf{bits}(\mathbf{T})) \otimes \mathbf{G}) + \mathbf{e}_{\mathrm{D}}^{\mathsf{T}})\mathbf{H}_{\mathrm{F},\mathbf{T}} \\ &= \mathbf{t}^{\mathsf{T}}\mathbf{A}_{\mathsf{path}}\mathbf{H}_{\mathrm{F}} - \mathbf{t}^{\mathsf{T}}\mathsf{VEval}_{\mathrm{F}}(\mathsf{bits}(\mathbf{T})) + \mathbf{e}_{\mathrm{D}}^{\mathsf{T}}\mathbf{H}_{\mathrm{F},\mathbf{T}} \\ &= \mathbf{t}^{\mathsf{T}}\mathbf{A}_{\mathbf{r}} - \mathrm{F}(\mathbf{s}, \mathsf{sd}) + (\mathbf{e}'_{\mathsf{fhe}})^{\mathsf{T}}\mathbf{R}_{\mathrm{F}} + \mathbf{e}_{\mathrm{D}}^{\mathsf{T}}\mathbf{H}_{\mathrm{F},\mathbf{T}} \\ &= \mathbf{t}^{\mathsf{T}}\mathbf{A}_{\mathbf{r}} - (\mathbf{S}_{\mathbf{r}}, \mathbf{E}_{\mathbf{r}}) + \mathbf{e}_{\mathrm{F}}^{\mathsf{T}}. \end{aligned}$$

   where $\mathbf{e}_{\mathrm{F}}^{\mathsf{T}} = (\mathbf{e}'_{\mathsf{fhe}})^{\mathsf{T}}\mathbf{R}_{\mathrm{F}} + \mathbf{e}_{\mathrm{D}}^{\mathsf{T}}\mathbf{H}_{\mathrm{F},\mathbf{T}}$ and by Lemma 2.29, we have

   $$\begin{aligned} \left\|\mathbf{R}_{\mathrm{F}}^{\mathsf{T}}\right\| &\le (m+2)^{d_{\mathrm{F}}} \lceil \log q \rceil \cdot \mathsf{max}_{i\in[m(n+1)\lceil\log q\rceil+\lambda]} \left\|\mathbf{R}_i^{\mathsf{T}}\right\| \\ &\le (m+2)^{d_{\mathrm{F}}} \lceil \log q \rceil \cdot m = (m+2)^{d_{\mathrm{F}}} \lceil \log q \rceil \cdot 3(n+1) \lceil \log q \rceil \\ &\le (m+2)^{d_{\mathrm{F}}} O(\log q) \le 2^{O(\log^4 \lambda)}. \end{aligned}$$

[4] This is an extra noise term in the fhe ciphertext. However, we bound it such that it is still correctly decryptable by secret key $\mathbf{s}_{\mathbf{r}}$.

Function F

Hardwired constants: $\mathbf{r} \in \mathcal{D}^m_{\mathbb{Z},\sigma_{\mathbf{r}}}$.
On input $(\mathbf{s} \in \mathbb{Z}^{m(n+1)}, \mathsf{sd} \in \{0,1\}^\lambda)$, proceed as follows:

(a) Set $\mathbf{s}_\mathbf{r}^\intercal := \mathbf{s}^\intercal(\mathbf{I}_{n+1} \otimes \mathbf{r})$ and parse $\mathbf{s}_\mathbf{r} = (\bar{\mathbf{s}}_\mathbf{r}^\intercal, -1)^\intercal$.

(b) Compute $\underline{\mathbf{a}}_\mathbf{r}^\intercal = \bar{\mathbf{s}}_\mathbf{r}^\intercal \bar{\mathbf{A}}_{\mathsf{fhe}} + \mathsf{PRF}_1(\mathsf{sd}, \mathbf{r})$.

(c) Compute $\mathbf{S}_\mathbf{r} = \begin{pmatrix} \bar{\mathbf{A}}_{\mathsf{fhe}} \\ \underline{\mathbf{a}}_\mathbf{r}^\intercal \end{pmatrix} \mathsf{PRF}_2(\mathsf{sd}, \mathbf{r}) - \mathsf{bits}(\mathbf{s}_\mathbf{r}) \otimes \mathbf{G} \in \mathbb{Z}_q^{(n+1)\times m(n+1)\lceil \log q \rceil}$.
Parse $\mathbf{S}_\mathbf{r} \in \mathbb{Z}_q^{1\times m(n+1)^2\lceil \log q \rceil}$ and set $\bar{\mathbf{S}}_\mathbf{r} = \mathsf{Rnd}(\mathbf{S}_\mathbf{r})$.

(d) Compute $\mathbf{E}_\mathbf{r} := \mathbf{s}_\mathbf{r}^\intercal(\mathbf{A}_{\mathsf{circ}} - (1, \mathsf{bits}(\bar{\mathbf{S}}_\mathbf{r})) \otimes \mathbf{G}) + \mathsf{PRF}_3(\mathsf{sd}, \mathbf{r}) \in \mathbb{Z}_q^{1\times(\mathrm{L_S}+1)m}$.

(e) Output $(\mathbf{S}_\mathbf{r}, \mathbf{E}_\mathbf{r}) \in \mathbb{Z}_q^{1\times\ell}$.

Figure 1: Function F

4. *Cancel out masking term.* Compute $\mathbf{c}_1^\intercal - \mathbf{D} \cdot \mathbf{H}_{\mathrm{F},\mathbf{T}}$.

$$\begin{aligned} &\mathbf{c}_1^\intercal - \mathbf{D} \cdot \mathbf{H}_{\mathrm{F},\mathbf{T}} \\ &= \mathbf{t}^\intercal(\mathbf{A}_\mathbf{r}) + \mathbf{e}_1^\intercal - (\mathbf{t}^\intercal \mathbf{A}_\mathbf{r} - (\mathbf{S}_\mathbf{r}, \mathbf{E}_\mathbf{r}) + \mathbf{e}_\mathrm{F}^\intercal) \\ &= (\mathbf{S}_\mathbf{r}, \mathbf{E}_\mathbf{r}) + \mathbf{e}^\intercal = (\mathbf{S}_\mathbf{r} + \mathbf{e}_\mathbf{S}^\intercal\ ,\ \mathbf{E}_\mathbf{r} + \mathbf{e}_\mathbf{E}^\intercal) \end{aligned}$$

where $\mathbf{e}^\intercal = \mathbf{e}_1^\intercal - \mathbf{e}_\mathrm{F}^\intercal \in \mathbb{Z}^{1\times\ell}$ and $\|\mathbf{e}_\mathbf{S}\|, \|\mathbf{e}_\mathbf{E}\| \le \|\mathbf{e}\|$.

5. *Output* Output $\mathbf{S}'_\mathbf{r} = \mathsf{Rnd}(\mathbf{S}_\mathbf{r} + \mathbf{e}_\mathbf{S}^\intercal)$ and $\mathbf{E}'_\mathbf{r} = \mathbf{E}_\mathbf{r} + \mathbf{e}_\mathbf{E}^\intercal$.

We encapsulate the property of our algorithm $\mathsf{ComputeAdvice}_\mathbf{r}$ in the following lemma.

Lemma 3.7 ($\mathsf{ComputeAdvice}_\mathbf{r}$)**.** If $\sigma', \|\mathbf{e}'_{\mathsf{fhe}}\|, \|\mathbf{e}_\mathbf{D}\| \le 2^{-2\lambda}B', \|\mathbf{e}_1\| \le 2^{-\lambda/2}B',\ q = B'p\lambda^{\omega(1)}, \|\mathbf{s}_\mathbf{r}\| \le 2^{-\lambda}B'$, for a bound $B' = 2^{-4\lambda}B$ where $B < q/4$ is the bound on the removable error, then for procedure $\mathsf{ComputeAdvice}_\mathbf{r}$ defined above, the output $(\mathbf{S}'_\mathbf{r}, \mathbf{E}'_\mathbf{r})$ is suitable for HLL Bootstrapping. Specifically, we have

- $\bar{\mathbf{S}}_\mathbf{r} = \mathbf{S}'_\mathbf{r}$ with overwhelming probability.
- $\bar{\mathbf{S}}_\mathbf{r}$ and $\mathbf{S}'$ are valid fhe ciphertexts encrypting $\mathsf{bits}(\mathbf{s}_\mathbf{r})$ using key $\mathbf{s}_\mathbf{r}$ with decryption error $\le 2^{-\lambda}B$
- The error in $\mathbf{E}'_\mathbf{r}$ is bounded as $\|\mathbf{e}_{\mathsf{circ}}\| \le 2^{-\lambda}B$.

Proof. We show that the output of $\mathsf{ComputeAdvice}_\mathbf{r}$ is valid for HLL bootstrapping.

First we note that since $\|\mathbf{e}'_{\mathsf{fhe}}\|, \|\mathbf{e}_\mathbf{D}\| \le 2^{-2\lambda}B'$ and $\|\mathbf{R}_\mathrm{F}^\intercal\|, \|(\mathrm{H}_{\mathrm{F},\mathbf{T}})^\intercal\| \le 2^{\mathrm{poly}(\log\lambda)}$, from Step 3 of $\mathsf{ComputeAdvice}_\mathbf{r}$ we have $\|\mathbf{e}_\mathrm{F}\| \le 2^{-3\lambda/2}B'$. Also, since $\|\mathbf{e}_1\| \le 2^{-\lambda/2}B'$, from Step 4 we have $\|\mathbf{e}_\mathbf{S}\|, \|\mathbf{e}_\mathbf{E}\| \le \|\mathbf{e}\| \le \|\mathbf{e}_1\| + \|\mathbf{e}_\mathrm{F}\| \le B'$.

- Equality of $\bar{\mathbf{S}}_\mathbf{r}$ and $\mathbf{S}'_\mathbf{r}$. We have $\bar{\mathbf{S}}_\mathbf{r} = \mathsf{Rnd}(\mathbf{S}_\mathbf{r})$ and $\mathbf{S}'_\mathbf{r} = \mathsf{Rnd}(\mathbf{S}_\mathbf{r} + \mathbf{e}_\mathbf{S}^\intercal)$. From Claim 3.2, we know that $\mathsf{Rnd}(\mathbf{S}_\mathbf{r}) = \mathsf{Rnd}(\mathbf{S}_\mathbf{r} + \mathbf{e}_\mathbf{S}^\intercal)$ with overwhelming probability if $\|(\mathbf{S}_\mathbf{r} - (\mathbf{S}_\mathbf{r} + \mathbf{e}_\mathbf{S}^\intercal))^\intercal\| = \|\mathbf{e}_\mathbf{S}\| \le \epsilon\frac{q}{p}$ for some negligible parameter ϵ. Note that $\|\mathbf{e}_\mathbf{S}\| \le B' \le \frac{1}{\lambda^{\omega(1)}}q/p$. Thus we have $\mathsf{Rnd}(\mathbf{S}_\mathbf{r}) = \mathsf{Rnd}(\mathbf{S}_\mathbf{r} + \mathbf{e}_\mathbf{S}^\intercal)$ with overwhelming probability.

- Validity of $\bar{\mathbf{S}}_\mathbf{r}$ and $\mathbf{S}'_\mathbf{r}$.

  First we show that $\bar{\mathbf{S}}_\mathbf{r} = \mathbf{S}_\mathbf{r} + \mathbf{err}_1 \in \mathbb{Z}_q^{(n+1)\times m(n+1)\lceil \log q\rceil}$ is a valid fhe encryption of $\mathbf{s}_\mathbf{r}$. Consider the decryption equation

$$\begin{aligned}\mathbf{s}_\mathbf{r}^\intercal \bar{\mathbf{S}}_\mathbf{r} &= \mathbf{s}_\mathbf{r}^\intercal \mathbf{S}_\mathbf{r} + \mathbf{s}_\mathbf{r}^\intercal \mathbf{err}_1 \\ &= -\mathsf{PRF}_1(\mathsf{sd},\mathbf{r})\mathsf{PRF}_2(\mathsf{sd},\mathbf{r}) - \mathbf{s}_\mathbf{r}(\mathsf{bits}(\mathbf{s}_\mathbf{r}) \otimes \mathbf{G}) + \mathbf{s}_\mathbf{r}^\intercal \mathbf{err}_1 \\ &= -\mathbf{s}_\mathbf{r}(\mathsf{bits}(\mathbf{s}_\mathbf{r}) \otimes \mathbf{G}) + \mathbf{err}_{\mathsf{fhe}}^\intercal.\end{aligned}$$

  which can be used to extract $\mathsf{bits}(\mathbf{s}_\mathbf{r})$ via tensoring by $\mathbf{G}^{-1}(\lfloor q/2 \rfloor\, \iota_{n+1})$ if $\|\mathbf{err}_{\mathsf{fhe}}\| \le 2^{-\lambda}B < q/4$, where $\mathbf{err}_{\mathsf{fhe}} = \mathbf{s}_\mathbf{r}^\intercal \mathbf{err}_1 - \mathsf{PRF}_1(\mathsf{sd},\mathbf{r})\mathsf{PRF}_2(\mathsf{sd},\mathbf{r})$.
  Note that Claim 3.1 implies $\|\mathbf{err}_1\| \le \frac{q}{2p} \le B'\lambda^{\omega(1)}$. Then, we have $\|\mathbf{err}_{\mathsf{fhe}}\| \le B' \cdot B'\lambda^{\omega(1)} + O(m)\sigma' \le \lambda^{\omega(1)}(B')^2 + O(m)2^{-\lambda}B' \le 2^{-\lambda}B$.
  Next, we show that $\mathbf{S}'_\mathbf{r} = \mathbf{S}_\mathbf{r} + \mathbf{e}_\mathbf{S} + \mathbf{err}_2 \in \mathbb{Z}_q^{(n+1)\times m(n+1)\lceil \log q\rceil}$ is a valid fhe encryption of $\mathbf{s}_\mathbf{r}$. Consider the decryption equation

$$\begin{aligned}\mathbf{s}_\mathbf{r}^\intercal \mathbf{S}'_\mathbf{r} &= \mathbf{s}_\mathbf{r}^\intercal \mathbf{S}_\mathbf{r} + \mathbf{s}_\mathbf{r}^\intercal(\mathbf{e}_\mathbf{S} + \mathbf{err}_2) \\ &= -\mathbf{s}_\mathbf{r}^\intercal(\mathsf{bits}(\mathbf{s}_\mathbf{r}) \otimes \mathbf{G}) + (\mathbf{err}'_{\mathsf{fhe}})^\intercal.\end{aligned}$$

  which can be used to extract $\mathsf{bits}(\mathbf{s}_\mathbf{r})$ via tensoring by $\mathbf{G}^{-1}(\lfloor q/2 \rfloor\, \iota_{n+1})$ if $\|\mathbf{err}'_{\mathsf{fhe}}\| \le 2^{-\lambda}B < q/4$, where $(\mathbf{err}'_{\mathsf{fhe}})^\intercal = \mathbf{s}_\mathbf{r}^\intercal(\mathbf{e}_\mathbf{S} + \mathbf{err}_2) - \mathsf{PRF}_1(\mathsf{sd},\mathbf{r})\mathsf{PRF}_2(\mathsf{sd},\mathbf{r})$ and $\|\mathbf{err}'_{\mathsf{fhe}}\| \le 2^{-\lambda}B$. The analysis is similar to $\|\mathbf{e}_{\mathsf{fhe}}\|$, hence omitted.

- Error bound in $\mathbf{E}'_\mathbf{r}$. We have $\mathbf{E}'_\mathbf{r} = \mathbf{s}_\mathbf{r}^\intercal(\mathbf{A}_{\mathsf{circ}} - (1, \mathsf{bits}(\bar{\mathbf{S}}_\mathbf{r})) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{circ}}^\intercal$, where $\mathbf{e}_{\mathsf{circ}}^\intercal = \mathsf{PRF}_3(\mathsf{sd},\mathbf{r}) + \mathbf{e}_\mathbf{E}^\intercal$. We have $\|\mathbf{e}_{\mathsf{circ}}\| \le 2^{-\lambda}B' + B' \le 2^{-\lambda}B$.

□

Step 2: HLL Bootstrapping On input $\mathbf{A}_{\mathsf{circ}}$, $\mathbf{S}'_\mathbf{r} = \mathsf{hct}_{\mathbf{s}_\mathbf{r}}(\mathbf{s}_\mathbf{r}) + \mathbf{err}_\mathbf{S}$, and $\mathbf{E}'_\mathbf{r} = \mathbf{s}_\mathbf{r}^\intercal(\mathbf{A}_{\mathsf{circ}} - (1, \mathsf{bits}(\mathbf{S}'_\mathbf{r})) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{circ}}^\intercal$[5], it does the following[6].

1. *Defining circuits for homomorphic evaluation*. Define the following evaluation circuits/matrices corresponding to the function $\mathsf{RndPad}_\mathbf{A}(\cdot)$.
   - Define $\mathsf{VEval}_{\mathsf{RndPad}_\mathbf{A}} = \mathsf{MakeVEvalCkt}(\mathsf{RndPad}_\mathbf{A})$.
   - Compute $\mathrm{H}_\mathbf{A}^{\mathsf{RndPad}} = \mathsf{MEvalC}(\mathbf{A}_{\mathsf{circ}}, \mathsf{VEval}_{\mathsf{RndPad}_\mathbf{A}}) \in \mathbb{Z}^{(\mathrm{L_S}+1)m\times m}$ and $\mathrm{H}_{\mathbf{A},\mathbf{S}'_\mathbf{r}}^{\mathsf{RndPad}} = \mathsf{MEvalCX}(\mathbf{A}_{\mathsf{circ}}, \mathsf{VEval}_{\mathsf{RndPad}_\mathbf{A}}, \mathbf{S}'_\mathbf{r}) \in \mathbb{Z}^{(\mathrm{L_S}+1)m\times m}$.

   Here $\left\|(\mathrm{H}_\mathbf{A}^{\mathsf{RndPad}})^\intercal\right\|, \left\|(\mathrm{H}_{\mathbf{A},\mathbf{S}'_\mathbf{r}}^{\mathsf{RndPad}})^\intercal\right\| \le 2^{O(\log^5 \lambda)}$.
2. *Compute the matrix homomorphically on* $\mathbf{E}'_\mathbf{r}$. We have

$$\begin{aligned}\mathbf{E}'_\mathbf{r} \cdot \mathrm{H}_{\mathbf{A},\mathbf{S}'_\mathbf{r}}^{\mathsf{RndPad}} &= \left(\mathbf{s}_\mathbf{r}^\intercal(\mathbf{A}_{\mathsf{circ}} - (1, \mathsf{bits}(\mathbf{S}'_\mathbf{r})) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{circ}}^\intercal\right) \mathrm{H}_{\mathbf{A},\mathbf{S}'_\mathbf{r}}^{\mathsf{RndPad}} \\ &= \mathbf{s}_\mathbf{r}^\intercal \mathbf{A}_{\mathsf{circ}} \mathrm{H}_\mathbf{A}^{\mathsf{RndPad}} - \mathsf{RndPad}_\mathbf{A}(\mathbf{s}_\mathbf{r}) + \mathbf{err}^\intercal\end{aligned}$$

[5]We replace $\bar{\mathbf{S}}_\mathbf{r}$ with $\mathbf{S}'_\mathbf{r}$ in the $\mathbf{E}'_\mathbf{r}$, since $\bar{\mathbf{S}}_\mathbf{r} = \mathbf{S}'_\mathbf{r}$ with overwhelming probability.

[6]Note that this doesn't straightforward follows from HLL. The error terms are different in the circular ciphertext $\mathbf{S}'_\mathbf{r}$. However, we can still get the final error with the desired bounds as analysed below.

where, using the fact that depth of RndPad is $O(\log n \log\log q)$, we have

$$\begin{aligned}\|\mathbf{err}\| \leq & (m+2)^{O(\log n \log\log q)} \cdot m \cdot (\|\mathsf{PRF}_1(\mathsf{sd},\mathbf{r})^\intercal\| + \|(\mathbf{s}_\mathbf{r}^\intercal(\mathbf{e}_\mathbf{S}+\mathbf{err}_2))^\intercal\|) \\ & + \|\mathbf{e}_{\mathsf{circ}}\| \cdot 2^{O(\log^5 \lambda)}.\end{aligned}$$

Assuming $\|\mathsf{PRF}_1(\mathsf{sd},\mathbf{r})^\intercal\|, \|(\mathbf{s}_\mathbf{r}^\intercal(\mathbf{e}_\mathbf{S}+\mathbf{err}_2))^\intercal\|, \|\mathbf{e}_{\mathsf{circ}}\| \leq 2^{-\lambda}B$, we get $\|\mathbf{err}\| \leq 2^{-\lambda/2}B$.

3. *Output* $\mathbf{s}_\mathbf{r}^\intercal \mathbf{A}_C - \mathsf{RndPad}_\mathbf{A}(\mathbf{s}_\mathbf{r}) + \mathbf{err}^\intercal$, where $\mathbf{A}_C = \mathbf{A}_{\mathsf{circ}}\mathrm{H}_\mathbf{A}^{\mathsf{RndPad}}$.

Restoring the structure. We note that combining the outputs from Section 3.2 and 3.3, we get

$$\mathbf{s}_\mathbf{r}^\intercal \mathbf{A}_C - C(\mathbf{x})\mathbf{s}_\mathbf{r}^\intercal \mathbf{G} + \mathbf{err}^\intercal, \quad \text{where } \|\mathbf{err}\| \leq 2^{-\lambda/2}B.$$

4 Ciphertext Policy ABE for Unbounded Depth Circuits

In this section we present CP-ABE for unbounded depth and size circuits.

4.1 Construction

We construct a CPABE scheme for circuit class $\mathcal{C} = \{C : \{0,1\}^{\mathrm{L}} \to \{0,1\}\}$.

$\mathsf{Setup}(1^\lambda, 1^{\mathrm{L}})$. The setup algorithm does the following.

1. Set $\mathrm{L_S} = m(n+1)^2 \lceil \log q \rceil^2, \mathrm{L_T} = m(n+1)(m(n+1)\lceil \log q \rceil + \lambda)\lceil \log q \rceil$, and $\ell = \mathrm{L_S}(1+m\log q)$.
2. Sample: $(\mathbf{B},\tau) \leftarrow \mathsf{TrapGen}(1^{(n+1)(m+2)}, 1^{(n+1)(m+2)w}, q)$, $\bar{\mathbf{A}}_{\mathsf{fhe}} \leftarrow \mathbb{Z}_q^{n\times m}$, $\mathbf{A}_0 \leftarrow \mathbb{Z}_q^{(n+1)\times m}$, $\mathbf{A}_{\mathsf{path}} \leftarrow \mathbb{Z}_q^{(n+1)\times(\mathrm{L_T}+1)m}$, $\mathbf{A}_{\mathsf{att}} \leftarrow \mathbb{Z}_q^{(n+1)\times(\mathrm{L}+1)m}$, $\mathbf{A}_{\mathsf{circ}} \leftarrow \mathbb{Z}_q^{(n+1)\times(\mathrm{L_S}+1)m}$, $\mathbf{u} \leftarrow \mathcal{D}^m_{\mathbb{Z},\sigma_\mathbf{r}}$, where $w \in O(\log q)$.
3. Output $\mathsf{mpk} := (\bar{\mathbf{A}}_{\mathsf{fhe}}, \mathbf{A}_0, \mathbf{A}_{\mathsf{path}}, \mathbf{A}_{\mathsf{att}}, \mathbf{A}_{\mathsf{circ}}, \mathbf{B}, \mathbf{u})$[7] and $\mathsf{msk} := \tau$.

$\mathsf{KeyGen}(\mathsf{msk}, \mathbf{x})$. The key generation algorithm does the following.

1. For $1 \leq j \leq m-1$, sample $\mathbf{r}_j \leftarrow \mathcal{D}_{\mathbb{Z},\sigma_\mathbf{r}}$. Set $\mathbf{r} = (\mathbf{r}_1, \ldots, \mathbf{r}_m)$ such that $\sum_{j\in[m]} \mathbf{r}_j = 1$.
2. Define function F as in Figure 1 using $\mathbf{r}$, $\mathsf{VEval}_\mathrm{F} = \mathsf{MakeVEvalCkt}(n,m,q,\mathrm{F}) \in \mathbb{Z}_q^{n+1\times\ell}$ and compute $\mathbf{H}_\mathrm{F} = \mathsf{MEvalC}(\mathbf{A}_{\mathsf{path}}, \mathsf{VEval}_\mathrm{F})$. Set $\mathbf{A}_\mathbf{r} = \mathbf{A}_{\mathsf{path}}\mathbf{H}_\mathrm{F}$.
3. Sample
$$\mathbf{K} \leftarrow \mathbf{B}_\tau^{-1}\begin{pmatrix}\mathbf{A}_0\mathbf{r} \\ (\mathbf{A}_{\mathsf{att}} - (1,\mathbf{x}^\intercal)\otimes\mathbf{G})\otimes\mathbf{r} \\ \mathbf{A}_\mathbf{r}\end{pmatrix}$$
4. Output $\mathsf{sk} := (\mathbf{K},\mathbf{r})$.

$\mathsf{Enc}(\mathsf{mpk}, C, \mu)$. The encryption algorithm does the following.

1. Sample $\mathbf{s}_0 \leftarrow \mathcal{D}^{(n+1)}_{\mathbb{Z},\sigma_\mathbf{s}}$, $\mathbf{s}_j \leftarrow \mathcal{D}^m_{\mathbb{Z},\sigma_\mathbf{s}}$ for $1 \leq j \leq n$, $\bar{\mathbf{t}} \leftarrow \mathcal{D}^n_{\mathbb{Z},\sigma_\mathbf{s}}$, $\mathbf{e}'_0 \leftarrow \mathcal{D}^{(n+1)w}_{\mathbb{Z},\sigma_0}$, $\mathbf{e}'_{\mathsf{att}} \leftarrow \mathcal{D}^{m(n+1)w}_{\mathbb{Z},\sigma_{\mathsf{att}}}$, $\mathbf{e}'_1 \leftarrow \mathcal{D}^{(n+1)w}_{\mathbb{Z},\sigma_1}$ $\mathbf{e}_{\mathsf{msg}} \leftarrow \mathcal{D}^m_{\mathbb{Z},\sigma_{\mathsf{msg}}}$. Set $\mathbf{s} = (\mathbf{s}_1^\intercal, \ldots, \mathbf{s}_n^\intercal, -\mathbf{1}_\mathbf{m})^\intercal$, $\mathbf{t} = (\bar{\mathbf{t}}^\intercal, -1)^\intercal, \mathbf{e}_\mathbf{B}^\intercal = ((\mathbf{e}'_0)^\intercal, (\mathbf{e}'_{\mathsf{att}})^\intercal, (\mathbf{e}'_1)^\intercal)$.
2. Compute
$$\mathbf{c}_\mathbf{B}^\intercal := (\mathbf{s}_0^\intercal \mid \mathbf{s}^\intercal \mid \mathbf{t}^\intercal)\mathbf{B} + \mathbf{e}_\mathbf{B}^\intercal, \quad \mathsf{ct}_{\mathsf{msg}}^\intercal := \mathbf{s}_0^\intercal\mathbf{A}_0 + \mathbf{s}^\intercal(\mathbf{A}_C\mathbf{u}\otimes\mathbf{I}_m) + \mu\cdot\mathbf{g}^\intercal + \mathbf{e}_{\mathsf{msg}}^\intercal$$
where $\mathbf{A}_C \leftarrow \mathsf{UEvalC}(\mathbf{A}_{\mathsf{circ}}, \mathbf{A}_{\mathsf{att}}, C)$.

[7] All the algorithms take mpk implicitly.

3. For $\mathsf{sd} \leftarrow \{0,1\}^\lambda$ and $\mathbf{R} = (\mathbf{R}_1, \ldots, \mathbf{R}_{m(n+1)\lceil \log q \rceil + \lambda})$, compute fhe ciphertext as follows.

$$\bar{\mathbf{A}}'_{\mathsf{fhe}} \leftarrow \mathbb{Z}_q^{n \times m}, \;\; \mathbf{e}'_{\mathsf{fhe}} \leftarrow \mathcal{D}^m_{\mathbb{Z},\sigma'}, \;\; \mathbf{A}_{\mathbf{t}} := (\bar{\mathbf{A}}'_{\mathsf{fhe}} \;\; \bar{\mathbf{t}}^\intercal \bar{\mathbf{A}}'_{\mathsf{fhe}} + (\mathbf{e}'_{\mathsf{fhe}})^\intercal)^\intercal,$$

$$\mathbf{T} = \mathbf{A}_{\mathbf{t}}\mathbf{R} - \mathsf{bits}(\mathbf{s}, \mathsf{sd}) \otimes \mathbf{G} \in \mathbb{Z}_q^{(n+1) \times m(m(n+1)\lceil \log q \rceil + \lambda)}$$

where $\mathbf{R}_i \leftarrow \{0,1\}^{m \times m}$ for all $1 \le i \le (m(n+1)\lceil \log q \rceil + \lambda)$.
Set $\mathrm{L_T} = m(n+1)(m(n+1)\lceil \log q \rceil + \lambda)\lceil \log q \rceil$ so that $\mathsf{bits}(\mathbf{T}) \in \{0,1\}^{\mathrm{L_T}}$.

4. Computes circular encoding as follows.

$$\mathbf{e}_{\mathrm{D}} \leftarrow \mathcal{D}^{(\mathrm{L_T}+1)m}_{\mathbb{Z},\sigma'}, \;\; \mathbf{D} = \mathbf{t}^\intercal(\mathbf{A}_{\mathsf{path}} - (1, \mathsf{bits}(\mathbf{T})) \otimes \mathbf{G}) + \mathbf{e}_{\mathrm{D}}^\intercal.$$

5. Output $\mathsf{ct} = (\mathbf{c}_{\mathbf{B}}, \mathbf{c}_{\mathsf{msg}}, \mathbf{T}, \mathbf{D})$.

$\mathsf{Dec}(\mathsf{sk}, \mathbf{x}, \mathsf{ct}, C)$. The decryption algorithm does the following.

1. Parse $\mathsf{sk} = (\mathbf{K}, \mathbf{r}^\intercal)$ and $\mathsf{ct} = (\mathbf{A}_{\mathbf{t}}, \mathbf{c}_{\mathbf{B}}, \mathbf{c}_{\mathsf{msg}}, \mathbf{T}, \mathbf{D})$.
2. Compute $\mathsf{ct}_{\mathbf{B}}^\intercal \cdot \mathbf{K}$ and parse it as $(\mathbf{c}_0^\intercal \;\; \mathbf{c}_{\mathsf{att}}^\intercal \;\; \mathbf{c}_1^\intercal)$ where $\mathbf{c}_0 \in \mathbb{Z}_q$, $\mathbf{c}_{\mathsf{att}} \in \mathbb{Z}_q^{(\mathrm{L}+1)m}$, $\mathbf{c}_1 \in \mathbb{Z}_q^\ell$.
3. Compute $(\mathbf{S}'_{\mathbf{r}}, \mathbf{E}'_{\mathbf{r}}) \leftarrow \mathsf{ComputeAdvice}_{\mathbf{r}}(\bar{\mathbf{A}}_{\mathsf{fhe}}, \mathbf{A}_{\mathsf{circ}}, \mathbf{A}_{\mathsf{path}}, \mathbf{T}, \mathbf{D}, \mathbf{c}_1^\intercal)$.
4. Set $\mathbf{c}_{\mathsf{circ}}^\intercal := \mathbf{E}'_{\mathbf{r}}$ and compute $\mathbf{c}_{C,\mathbf{x}}^\intercal \leftarrow \mathsf{UEvalCX}(\mathbf{A}_{\mathsf{att}}, \mathbf{c}_{\mathsf{att}}^\intercal, \mathbf{A}_{\mathsf{circ}}, \mathbf{c}_{\mathsf{circ}}^\intercal, C, \mathbf{x}, \mathbf{S}'_{\mathbf{r}})$.
5. Output $\mathsf{ct}_{\mathsf{msg}}^\intercal \cdot \mathbf{r} - \mathbf{c}_0^\intercal - \mathbf{c}_{C,\mathbf{x}}^\intercal \cdot \mathbf{u}$

Parameters. We set our parameters as follows.

$$n = \mathrm{poly}(\lambda),\; q = 2^{14\lambda}\lambda^{\omega(1)},\; p = 2^{10\lambda},\; m = O(n \log q),\; B = 2^{7\lambda},$$

$$\sigma_{\mathbf{r}} = \mathrm{poly}(\lambda),\; \sigma_{\mathbf{s}} = \sigma' = 2^\lambda,\; \sigma_0 = 2^{7\lambda},\; \sigma_1 = 2^{2\lambda}\lambda^{\omega(1)},\; \sigma_{\mathsf{msg}} = 2^{5\lambda},\; \sigma_{\mathsf{att}} = 2^{5\lambda},$$

$$\tau = O(\sqrt{(n+1)(m+2)\log q}),\; \chi_0 = 2^{6\lambda}\lambda^{\omega(1)},\; \chi_1 = 2^{2\lambda}\lambda^{\omega(1)},\; \chi_{\mathsf{att}} = 2^{3\lambda}$$

where $\chi_0, \chi_1, \chi_{\mathsf{att}}$ appear only in the security proof.

Efficiency. Using the above set parameters, we have

$$|\mathsf{mpk}| = \mathrm{poly}(\lambda, L), \;\; |\mathsf{sk}| = \mathrm{poly}(\lambda, L), \;\; |\mathsf{ct}| = \mathrm{poly}(\lambda).$$

Correctness We show the correctness step by step below.

– First, we note that

$$\mathsf{ct}_{\mathbf{B}} \cdot \mathbf{K} = \left((\mathbf{s}_0^\intercal \mid \mathbf{s}^\intercal \mid \mathbf{t}^\intercal)\mathbf{B} + \mathbf{e}_{\mathbf{B}}^\intercal\right) \mathbf{B}_\tau^{-1} \begin{pmatrix} \mathbf{A}_0\mathbf{r} & & \\ & (\mathbf{A}_{\mathsf{att}} - (1, \mathbf{x}^\intercal) \otimes \mathbf{G}) \otimes \mathbf{r} & \\ & & \mathbf{A}_{\mathbf{r}} \end{pmatrix}$$

$$(\mathbf{c}_0^\intercal \;\; \mathbf{c}_{\mathsf{att}}^\intercal \;\; \mathbf{c}_1^\intercal) = (\mathbf{s}_0^\intercal(\mathbf{A}_0\mathbf{r}) + \mathbf{e}_0^\intercal \quad \mathbf{s}^\intercal((\mathbf{A}_{\mathsf{att}} - (1, \mathbf{x}^\intercal) \otimes \mathbf{G}) \otimes \mathbf{r}) + \mathbf{e}_{\mathsf{att}}^\intercal \quad \mathbf{t}^\intercal(\mathbf{A}_{\mathbf{r}}) + \mathbf{e}_1^\intercal)$$

where $\|\mathbf{e}_0\| \le \tau\|\mathbf{e}'_0\|, \|\mathbf{e}_{\mathsf{att}}\| \le \tau\|\mathbf{e}'_{\mathsf{att}}\|, \|\mathbf{e}_1\| \le \tau\|\mathbf{e}'_1\|$.

– Next, we have $\|\mathbf{s}_{\mathbf{r}}\| \le \sigma_{\mathbf{s}}\sigma_{\mathbf{r}} \cdot m \le 2^\lambda \cdot \mathrm{poly}(\lambda) \cdot m \le B$ and $q/p = 2^{3\lambda}\lambda^{\omega(1)}$.

– Correctness of $\mathsf{ComputeAdvice}_{\mathbf{r}}$. From our parameter setting, we have $\|\mathbf{e}_1\| \le \tau \cdot 2^{2\lambda}\lambda^{\omega(1)}\sqrt{\lambda}$, $\|\mathbf{e}_{\mathbf{D}}\|, \|\mathbf{e}'_{\mathsf{fhe}}\| \le 2^\lambda\sqrt{\lambda}$, where $\mathbf{e}_1, \mathbf{e}_{\mathbf{D}}, \mathbf{e}'_{\mathsf{fhe}}$ are the error terms present in the input to procedure $\mathsf{ComputeAdvice}_{\mathbf{r}}$. Next, we observe the following.

- The error term in Step 3 is $\mathbf{e}_{\mathsf{F}}^{\intercal} = (\mathbf{e}')^{\intercal}\mathbf{R}_{\mathsf{F}} + \mathbf{e}_{\mathsf{D}}^{\intercal}\mathbf{H}_{\mathsf{F},\mathbf{T}}$. We have $\|\mathbf{e}_{\mathsf{F}}\| \leq 2^{\lambda+\mathrm{poly}(\log\lambda)}\sqrt{\lambda}$.
- The error term in Step 4 is $\mathbf{e}^{\intercal} = \mathbf{e}_1^{\intercal} - \mathbf{e}_{\mathsf{F}}^{\intercal} \in \mathbb{Z}^{1\times\ell}$. We have $\|\mathbf{e}\| \leq 2^{3\lambda}$. This implies $\|\mathbf{e}_{\mathbf{S}}\|, \|\mathbf{e}_{\mathbf{E}}\| \leq 2^{3\lambda}$.
- The decryption error in $\bar{\mathbf{S}}_{\mathbf{r}}$ is $\mathbf{err}_{\mathsf{fhe}} = \mathbf{s}_{\mathbf{r}}^{\intercal}\mathbf{err}_1 - \mathsf{PRF}_1(\mathsf{sd},\mathbf{r})\mathsf{PRF}_2(\mathsf{sd},\mathbf{r})$.
  We have $\|\mathbf{err}_{\mathsf{fhe}}\| \leq O(m^2)\cdot\mathrm{poly}(\lambda)2^{5\lambda}\lambda^{\omega(1)} + m\cdot 2^{\lambda} \leq 2^{6\lambda}$, where we use the fact that the error introduced by $\mathsf{Rnd}(\cdot)$ is bounded by $q/2p$.
- The decryption error in $\mathbf{S}'_{\mathbf{r}}$ is $(\mathbf{err}'_{\mathsf{fhe}})^{\intercal} = \mathbf{s}_{\mathbf{r}}^{\intercal}(\mathbf{e}_{\mathbf{S}} + \mathbf{err}_2) - \mathsf{PRF}_1(\mathsf{sd},\mathbf{r})\mathsf{PRF}_2(\mathsf{sd},\mathbf{r})$.
  We have $\|\mathbf{err}'_{\mathsf{fhe}}\| \leq 2^{5\lambda} + O(m^2)\cdot\mathrm{poly}(\lambda)2^{5\lambda}\lambda^{\omega(1)} + m\cdot 2^{\lambda} \leq 2^{6\lambda}$, where we use the fact that the error introduced by $\mathsf{Rnd}(\cdot)$ is bounded by $q/2p$ and $\|\mathbf{s}_{\mathbf{r}}\|\cdot\|\mathbf{e}_{\mathbf{S}}\| \leq 2^{5\lambda}$.
- Next, we note that $\|\mathbf{e}_{\mathbf{S}}\| \leq 2^{3\lambda} \leq \frac{1}{2^{\lambda}}\cdot\frac{q}{p}$. This implies that $\mathsf{Rnd}(\mathbf{S}_{\mathbf{r}}) = \mathsf{Rnd}(\mathbf{S}_{\mathbf{r}} + \mathbf{e}_{\mathbf{S}}^{\intercal})$ with overwhelming probability.
- The error term in $\mathbf{E}'_{\mathbf{r}}$ is $\mathbf{e}_{\mathsf{circ}}^{\intercal} = \mathsf{PRF}_3(\mathsf{sd},\mathbf{r}) + \mathbf{e}_{\mathbf{E}}^{\intercal}$. We have $\|\mathbf{e}_{\mathsf{circ}}\| \leq 2^{\lambda} + 2^{3\lambda} \leq 2^{-\lambda}B$.

Using the above observations, we note that the procedure $\mathsf{ComputeAdvice}_{\mathbf{r}}$ outputs $\mathbf{S}'_{\mathbf{r}}$ and $\mathbf{E}'_{\mathbf{r}}$ where $\mathbf{E}'_{\mathbf{r}}$ encode $\mathbf{S}'_{\mathbf{r}}$ with all but negligible probability.

– Next, note that for $\mathbf{c}_{\mathsf{att}}^{\intercal} = \mathbf{s}^{\intercal}((\mathbf{A}_{\mathsf{att}} - (1,\mathbf{x}^{\intercal})\otimes\mathbf{G})\otimes\mathbf{r}) + \mathbf{e}_{\mathsf{att}}^{\intercal} = \mathbf{s}_{\mathbf{r}}^{\intercal}(\mathbf{A}_{\mathsf{att}} - (1,\mathbf{x}^{\intercal})\otimes\mathbf{G}) + \mathbf{e}_{\mathsf{att}}^{\intercal}$, where $\|\mathbf{s}_{\mathbf{r}}\|, \|\mathbf{e}_{\mathsf{att}}\| \leq B$. So, by the correctness of $\mathsf{UEvalCX}$, we have

$$\mathsf{ct}_{C,\mathbf{x}}^{\intercal} = \mathbf{s}_{\mathbf{r}}^{\intercal}(\mathbf{A}_C - C(\mathbf{x})\mathbf{G}) + \mathbf{e}_{C,\mathbf{x}}^{\intercal} = \mathbf{s}(\mathbf{I}_{n+1}\otimes\mathbf{r})(\mathbf{A}_C - C(\mathbf{x})\mathbf{G}) + \mathbf{e}_{C,\mathbf{x}}^{\intercal},$$

where, from Step 2 of procedure HLL Bootstrapping, it follows that $\|\mathbf{e}_{C,\mathbf{x}}\| \leq (m+2)^{O(\log n\log\log q)}\cdot m\cdot(\|\mathsf{PRF}_1(\mathsf{sd},\mathbf{r})^{\intercal}\| + \|(\mathbf{s}_{\mathbf{r}}^{\intercal}(\mathbf{e}_{\mathbf{S}} + \mathbf{err}_2))^{\intercal}\|) + \|\mathbf{e}_{\mathsf{circ}}\|\cdot 2^{O(\log^5\lambda)}$.
From our parameter setting and the error analysis above, $\|\mathbf{e}_{C,\mathbf{x}}\| \leq 2^{6\lambda} \leq 2^{-\lambda/2}B$.

– Next, for $C,\mathbf{x}$ such that $C(\mathbf{x}) = 0$, we have the following.

$$\begin{aligned}
&\mathsf{ct}_{\mathsf{msg}}^{\intercal}\cdot\mathbf{r} - \mathbf{c}_0^{\intercal} - \mathsf{ct}_{C,\mathbf{x}}^{\intercal}\cdot\mathbf{u}\\
&= (\mathbf{s}_0^{\intercal}\mathbf{A}_0 + \mathbf{s}^{\intercal}(\mathbf{A}_C\mathbf{u}\otimes\mathbf{I}_m) + \mu\cdot\mathbf{g}^{\intercal} + \mathbf{e}_{\mathsf{msg}}^{\intercal})\cdot\mathbf{r} - \mathbf{s}_0^{\intercal}(\mathbf{A}_0\mathbf{r}) - \mathbf{e}_0^{\intercal}\\
&- \mathbf{s}^{\intercal}(\mathbf{I}_{n+1}\otimes\mathbf{r})\mathbf{A}_C\mathbf{u} - \mathbf{e}_{C,\mathbf{x}}^{\intercal}\mathbf{u}\\
&= \mathbf{s}^{\intercal}(\mathbf{A}_C\mathbf{u}\otimes\mathbf{I}_m)\mathbf{r} + \mu\cdot\mathbf{g}^{\intercal}\mathbf{r} + \mathbf{e}_{\mathsf{msg}}^{\intercal}\mathbf{r} - \mathbf{e}_0^{\intercal} - \mathbf{s}^{\intercal}(\mathbf{A}_C\mathbf{u}\otimes\mathbf{I}_m)(\mathbf{1}\otimes\mathbf{r}) - \mathbf{e}_{C,\mathbf{x}}^{\intercal}\mathbf{u}\\
&\approx \mu\lfloor q/2\rceil + \mathbf{e}^{\intercal}.
\end{aligned}$$

where $\mathbf{e}^{\intercal} = \mathbf{e}_{\mathsf{msg}}^{\intercal}\mathbf{r} - \mathbf{e}_0^{\intercal} - \mathbf{e}_{C,\mathbf{x}}^{\intercal}\mathbf{u}$ and $\|\mathbf{e}\| \leq 2^{5\lambda}\sqrt{\lambda}\cdot m\cdot\mathrm{poly}(\lambda) - O(m)\cdot 2^{\lambda}\sqrt{\lambda} - 2^{6\lambda}\cdot\mathrm{poly}(\lambda) \leq 2^{7\lambda} < q/4$, which is within the bounds of rounding. Hence the decryption outputs μ with all but negligible probability.

4.2 Our Assumption

We detail our assumptions next. We introduce a circular version of the Tensor LWE assumption [Wee22] in similar spirit to the Evasive Circular assumption [HLL23].

Assumption 4.1 (Circular Tensor LWE). Let $\mathbf{s} = (\mathbf{s}_1^{\intercal},\cdots\mathbf{s}_n^{\intercal},-\mathbf{1}_m)^{\intercal}$ where $\mathbf{s}_i \leftarrow \mathcal{D}_{\mathbb{Z},\sigma_{\mathbf{s}}}^m$, $\mathbf{r} = (\mathbf{r}_1,\ldots,\mathbf{r}_m)$, where $\mathbf{r}_i \leftarrow \mathcal{D}_{\mathbb{Z},\sigma_{\mathbf{r}}}$ such that $\sum_{i\in[m]}\mathbf{r}_i = 1$. For $i\in[Q]$, set $\mathbf{s}_{\mathbf{r}_i}^{\intercal} = \mathbf{s}^{\intercal}(\mathbf{I}_{n+1}\otimes\mathbf{r}_i) = (\bar{\mathbf{s}}_{\mathbf{r}_i}^{\intercal},-1) \in \mathbb{Z}^{n+1}$. Suppose

$$\mathbf{A}_{\mathsf{att}} = \begin{pmatrix}\bar{\mathbf{A}}_{\mathsf{att}}\\ \mathbf{a}_{\mathsf{att}}^{\intercal}\end{pmatrix} \leftarrow \mathbb{Z}_q^{(n+1)\times(\mathrm{L}+1)m},\ \bar{\mathbf{A}}_{\mathsf{fhe}} \leftarrow \mathbb{Z}_q^{n\times m},\ \mathbf{A}_{\mathsf{circ}} = \begin{pmatrix}\bar{\mathbf{A}}_{\mathsf{circ}}\\ \mathbf{a}_{\mathsf{circ}}^{\intercal}\end{pmatrix} \leftarrow \mathbb{Z}_q^{(n+1)\times m_{\mathsf{circ}}}$$

and for $i\in[Q]$, $\sigma_{\mathsf{att}},\sigma' << q$, suppose

$$\mathbf{e}_{\mathsf{att},i} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma_{\mathsf{att}}}^{(\mathrm{L}+1)m},\ \mathbf{e}_{\mathsf{fhe},i} \leftarrow [-\sigma'\ ,\ \sigma']^m\ ,\ \mathbf{e}_{\mathsf{circ},i} \leftarrow [-\sigma'\ ,\ \sigma']^{m_{\mathsf{circ}}}$$

$$\delta_{\mathsf{att},i} \leftarrow \mathbb{Z}_q^{(\mathrm{L}+1)m},\ \ \delta_{\mathsf{fhe},i} \leftarrow \mathbb{Z}_q^{m},\ \delta_{\mathsf{circ},i} \leftarrow \mathbb{Z}_q^{m_{\mathsf{circ}}},\ \Delta_i \leftarrow \mathbb{Z}_q^{(n+1)\times m_{\mathbf{S}}}$$

$$\mathbf{R}_i \leftarrow \{0,1\}^{m\times m_{\mathbf{S}}}, \mathbf{S}_{\mathbf{r}_i} = \begin{pmatrix} \bar{\mathbf{A}}_{\mathsf{fhe}} \\ \bar{\mathbf{s}}_{\mathbf{r}_i}^{\intercal}\bar{\mathbf{A}}_{\mathsf{fhe}} + \mathbf{e}_{\mathsf{fhe},i}^{\intercal} \end{pmatrix} \mathbf{R}_i - \mathsf{bits}(\mathbf{s}_{\mathbf{r}_i}) \otimes \mathbf{G} \in \mathbb{Z}_q^{(n+1)\times m_{\mathbf{S}}}$$

where the parameters are functions of λ.
For all $\mathbf{x}_1, \ldots, \mathbf{x}_Q \in \{0,1\}^L$, the *circular tensor* LWE *assumption* states that

$$\left(1^\lambda, \mathbf{A}_{\mathsf{att}}, \mathbf{A}_{\mathsf{circ}}, \bar{\mathbf{A}}_{\mathsf{fhe}}, \left\{ \begin{matrix} \bar{\mathbf{s}}_{\mathbf{r}_i}^{\intercal}(\bar{\mathbf{A}}_{\mathsf{att}} - (1, \mathbf{x}_i^{\intercal}) \otimes \bar{\mathbf{G}}) + \mathbf{e}_{\mathsf{att},i}^{\intercal}, & \bar{\mathbf{s}}_{\mathbf{r}_i}^{\intercal}\bar{\mathbf{A}}_{\mathsf{fhe}} + \mathbf{e}_{\mathsf{fhe},i}^{\intercal}, \\ \mathbf{S}_{\mathbf{r}_i}, & \bar{\mathbf{s}}_{\mathbf{r}_i}^{\intercal}(\bar{\mathbf{A}}_{\mathsf{circ}} - (1, \mathsf{bits}(\mathbf{S}_i)) \otimes \bar{\mathbf{G}}) + \mathbf{e}_{\mathsf{circ},i}^{\intercal}, \ \mathbf{r}_i \end{matrix} \right\}_{i\in[Q]}\right)_{\lambda\in\mathbb{N}}$$

$$\approx \left(1^\lambda,\ \mathbf{A}_{\mathsf{att}},\ \mathbf{A}_{\mathsf{circ}},\ \bar{\mathbf{A}}_{\mathsf{fhe}},\ \{\delta_{\mathsf{att},i}^{\intercal},\ \delta_{\mathsf{fhe},i}^{\intercal},\ \Delta_i,\ \delta_{\mathsf{circ},i}^{\intercal}\}_{i\in[Q]}\right)_{\lambda\in\mathbb{N}}$$

Remark 4.2. Here, we choose $\mathbf{e}_{\mathsf{fhe}}$ and $\mathbf{e}_{\mathsf{circ}}$ from uniform distribution over an interval rather than from usual Gaussian distribution. This change makes little difference to the assumption, since when the modulus q is super-polynomial, we can reduce the problem with Gaussian distribution to the one with uniform distribution with (superpolynomially) larger width and vice versa, due to the smudging (Lemma 2.20).

4.3 Security Proof

Theorem 4.3. Assuming evasive LWE (assumption 2.24), circular tensor LWE (assumption 4.1) and LWE (assumption 2.22), there exists a CP-ABE scheme for unbounded depth and unbounded size circuits which satisfies very selective security (Definition 2.15).

Proof. Suppose the ABE adversary $\mathcal{A}$ with randomness $\mathsf{coins}_{\mathcal{A}}$ queries C and $\mathbf{x}_1, \ldots, \mathbf{x}_Q$ such that $C(\mathbf{x}_1) = \cdots = C(\mathbf{x}_Q) = 1$. The view of the adversary when bit μ is encoded is:

$$\begin{pmatrix} \mathsf{mpk} = (\bar{\mathbf{A}}_{\mathsf{fhe}}, \mathbf{A}_0, \mathbf{A}_{\mathsf{path}}, \mathbf{A}_{\mathsf{att}}, \mathbf{A}_{\mathsf{circ}}, \mathbf{B}, \mathbf{u}), \\ \mathbf{c}_{\mathbf{B}}^{\intercal} = (\mathbf{s}_0^{\intercal} \mid \mathbf{s}^{\intercal} \mid \mathbf{t}^{\intercal})\mathbf{B} + \mathbf{e}_{\mathbf{B}}^{\intercal},\ \ \mathsf{ct}_{\mathsf{msg}}^{\intercal} = \mathbf{s}_0^{\intercal}\mathbf{A}_0 + \mathbf{s}^{\intercal}(\mathbf{A}_C\mathbf{u} \otimes \mathbf{I}_m) + \mu \cdot \mathbf{g}^{\intercal} + \mathbf{e}_{\mathsf{msg}}^{\intercal} \\ \mathbf{T} = \mathbf{A}_{\mathsf{t}}\mathbf{R} - \mathsf{bits}(\mathbf{s}, \mathsf{sd}) \otimes \mathbf{G},\ \mathbf{D} = \mathbf{t}^{\intercal}(\mathbf{A}_{\mathsf{path}} - (1, \mathsf{bits}(\mathbf{T})) \otimes \mathbf{G}) + \mathbf{e}_{\mathbf{D}}^{\intercal},\quad \{\mathbf{K}_i,\quad \mathbf{r}_i\}_{i\in[Q]} \end{pmatrix}$$

where for $i \in [Q]$, we have

$$\mathbf{K}_i = \mathbf{B}_\tau^{-1} \begin{pmatrix} \mathbf{A}_0\mathbf{r}_i & & \\ & (\mathbf{A}_{\mathsf{att}} - (1, \mathbf{x}^{\intercal}) \otimes \mathbf{G}) \otimes \mathbf{r}_i & \\ & & \mathbf{A}_{\mathbf{r}_i} \end{pmatrix}$$

Note that to prove $\mathsf{ct}_{\mathsf{msg}}^{\intercal} = \mathbf{s}_0^{\intercal}\mathbf{A}_0 + \mathbf{s}^{\intercal}(\mathbf{A}_C\mathbf{u} \otimes \mathbf{I}_m) + \mu \cdot \mathbf{g}^{\intercal} + \mathbf{e}_{\mathsf{msg}}^{\intercal}$ is indistinguishable from random, it suffices to show that $\mathbf{c}_{\mathsf{msg}}^{\intercal} = \mathbf{s}_0^{\intercal}\mathbf{A}_0 + \mathbf{s}^{\intercal}(\mathbf{A}_C\mathbf{u} \otimes \mathbf{I}_m) + \mathbf{e}_{\mathsf{msg}}^{\intercal}$. We prove the pseudorandomness of the above distribution by replacing $\mathsf{ct}_{\mathsf{msg}}^{\intercal}$ with $\mathbf{c}_{\mathsf{msg}}^{\intercal}$.
We invoke the Lemma 2.26 variant of evasive LWE hardness assumption for a matrix $\mathbf{B}$ with Gaussian parameter τ and

a sampler Samp that outputs $(\mathbf{S}, \mathbf{P}, \mathsf{aux} = (\mathsf{aux}_1, \mathsf{aux}_2))$ defined as follows.

$$\mathsf{aux}_1 = \left(\mathsf{ct}_{\mathsf{msg}} = (\mathbf{s}_0,\ \mathbf{s}) \begin{pmatrix} \mathbf{A}_0 \\ \mathbf{A}_C\mathbf{u} \otimes \mathbf{I}_m \end{pmatrix},\ \mathbf{T} = \begin{pmatrix} \bar{\mathbf{A}}_{\mathsf{fhe}} \\ \bar{\mathbf{t}}^\intercal \bar{\mathbf{A}}_{\mathsf{fhe}} + \bar{\mathbf{e}}_{\mathsf{fhe}}^\intercal \end{pmatrix} \mathbf{R} - \mathsf{bits}(\mathbf{s}, \mathsf{sd}) \otimes \mathbf{G}, \right.$$
$$\left. \mathbf{D} = \mathbf{t}^\intercal(\mathbf{A}_{\mathsf{path}} - (1, \mathsf{bits}(\mathbf{T})) \otimes \mathbf{G}) + \mathbf{e}_{\mathbf{D}}^\intercal \right)$$
$$\mathsf{aux}_2 = (\mathbf{x}_1, \ldots, \mathbf{x}_Q, C, \mathsf{coins}_{\mathcal{A}}, \mathbf{r}_1, \ldots, \mathbf{r}_Q, \bar{\mathbf{A}}_{\mathsf{fhe}}, \mathbf{A}_0, \mathbf{A}_{\mathsf{path}}, \mathbf{A}_{\mathsf{att}}, \mathbf{A}_{\mathsf{circ}}, \mathbf{u})$$
$$\mathbf{S} = (\mathbf{s}_0^\intercal \mid \mathbf{s}^\intercal \mid \mathbf{t}^\intercal)$$
$$\mathbf{P}_0 = (\mathbf{A}_0\mathbf{r}_1\|, \ldots, \|\mathbf{A}_0\mathbf{r}_Q)$$
$$\mathbf{P}_1 = ((\mathbf{A}_{\mathsf{att}} - (1, \mathbf{x}^\intercal) \otimes \mathbf{G}) \otimes \mathbf{r}_1\|, \ldots, \|(\mathbf{A}_{\mathsf{att}} - (1, \mathbf{x}^\intercal) \otimes \mathbf{G}) \otimes \mathbf{r}_Q)$$
$$\mathbf{P}_2 = (\mathbf{A}_{\mathbf{r}_1}\|, \ldots, \|\mathbf{A}_{\mathbf{r}_Q})$$
$$\mathbf{P} = \begin{pmatrix} \mathbf{P}_0 & & \\ & \mathbf{P}_1 & \\ & & \mathbf{P}_2 \end{pmatrix}$$

We set $\mathsf{aux}_0 = (\mathbf{x}_1, \ldots, \mathbf{x}_Q, C, \mathsf{coins}_{\mathcal{A}})$.
By applying evasive LWE, it suffices to show pseudorandomness of the following distribution, given aux_0,

$$\left(\begin{array}{c} \mathsf{mpk} = (\bar{\mathbf{A}}_{\mathsf{fhe}}, \mathbf{A}_0, \mathbf{A}_{\mathsf{path}}, \mathbf{A}_{\mathsf{att}}, \mathbf{A}_{\mathsf{circ}}, \mathbf{B}, \mathbf{u}) \\ \mathbf{c}_{\mathbf{B}}^\intercal = (\mathbf{s}_0^\intercal \mid \mathbf{s}^\intercal \mid \mathbf{t}^\intercal)\mathbf{B} + \mathbf{e}_{\mathbf{B}}^\intercal,\ \mathbf{c}_{\mathsf{msg}}^\intercal = \mathbf{s}_0^\intercal\mathbf{A}_0 + \mathbf{s}^\intercal(\mathbf{A}_C\mathbf{u} \otimes \mathbf{I}_m) + \mathbf{e}_{\mathsf{msg}}^\intercal, \\ \mathbf{T} = \mathbf{A}_{\mathbf{t}}\mathbf{R} - \mathsf{bits}(\mathbf{s}, \mathsf{sd}) \otimes \mathbf{G},\ \ \mathbf{D} = \mathbf{t}^\intercal(\mathbf{A}_{\mathsf{path}} - (1, \mathsf{bits}(\mathbf{T})) \otimes \mathbf{G}) + \mathbf{e}_{\mathbf{D}}^\intercal, \\ \{\mathbf{c}_{0,i}^\intercal = \mathbf{s}_0^\intercal(\mathbf{A}_0\mathbf{r}_i) + \mathbf{e}_{0,i}^\intercal\ ,\ \mathbf{c}_{\mathsf{att},i}^\intercal = \mathbf{s}_{\mathbf{r}_i}^\intercal(\mathbf{A}_{\mathsf{att}} - (1, \mathbf{x}_i^\intercal) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{att},i}^\intercal\ ,\ \mathbf{c}_{1,i}^\intercal = \mathbf{t}^\intercal\mathbf{A}_{\mathbf{r}_i} + \mathbf{e}_{1,i}^\intercal\ ,\ \mathbf{r}_i\}_{i \in [Q]} \end{array} \right) \tag{1}$$

where $\mathbf{e}_{0,i}^\intercal \leftarrow \mathcal{D}_{\mathbb{Z},\sigma_0}$, $\mathbf{e}_{\mathsf{att},i}^\intercal \leftarrow \mathcal{D}_{\mathbb{Z},\sigma_{\mathsf{att}}}^{(\mathrm{L}+1)m}$, $\mathbf{e}_{1,i}^\intercal \leftarrow \mathcal{D}_{\mathbb{Z},\sigma_1}^{\ell}$ for $i \in [Q]$.

Hyb_0. This is the distribution as specified in Equation (1).

Hyb_1. This hybrid is same as Hyb_0, except we compute $\mathbf{c}_{1,i}^\intercal$ as a function of $\mathbf{T}, \mathbf{D}, \mathbf{S}_{\mathbf{r}_i}, \mathbf{E}_{\mathbf{r}_i}$.

Specifically, we compute

$$\mathbf{c}_{1,i}^\intercal := \mathbf{D}\ \mathbf{H}_{\mathrm{F},\mathbf{T}} + (\mathbf{S}_{\mathbf{r}_i}, \mathbf{E}_{\mathbf{r}_i}) + \mathbf{e}_{1,i}^\intercal\ , \text{for}\ \ i \in [Q],$$

We claim that Hyb_0 and Hyb_1 are statistically indistinguishable. To see this, note that:

- $\mathbf{t}^\intercal(\mathbf{A}_{\mathbf{r}_i}) = \mathbf{D}\ \mathbf{H}_{\mathrm{F},\mathbf{T}} + (\mathbf{S}_{\mathbf{r}_i}, \mathbf{E}_{\mathbf{r}_i}) + \mathbf{e}_{\mathrm{F},i}^\intercal$, where $\|\mathbf{e}_{\mathrm{F},i}\| \leq 2^{\lambda + \mathrm{poly}(\log\lambda)}\sqrt{\lambda} \leq 2^{2\lambda}$, by our parameter setting.
- We have $\mathbf{e}_{1,i}^\intercal \approx_s \mathbf{e}_{\mathrm{F},i}^\intercal + \mathbf{e}_{1,i}^\intercal$ by noise flooding (Lemma 2.20) since $\lambda^{\omega(1)}\|\mathbf{e}_{\mathrm{F},i}\| \leq \lambda^{\omega(1)}2^{2\lambda} = \chi_1$.

Thus it suffices to show pseudorandomness of the following, given aux_0,

$$\left(\begin{array}{c} \mathsf{mpk} = (\bar{\mathbf{A}}_{\mathsf{fhe}}, \mathbf{A}_0, \mathbf{A}_{\mathsf{path}}, \mathbf{A}_{\mathsf{att}}, \mathbf{A}_{\mathsf{circ}}, \mathbf{B}, \mathbf{u}), \\ \mathbf{c}_{\mathbf{B}}^\intercal = (\mathbf{s}_0^\intercal \mid \mathbf{s}^\intercal \mid \mathbf{t}^\intercal)\mathbf{B} + \mathbf{e}_{\mathbf{B}}^\intercal,\ \mathbf{c}_{\mathsf{msg}}^\intercal = \mathbf{s}_0^\intercal\mathbf{A}_0 + \mathbf{s}^\intercal(\mathbf{A}_C\mathbf{u} \otimes \mathbf{I}_m) + \mathbf{e}_{\mathsf{msg}}^\intercal, \\ \mathbf{T} = \mathbf{A}_{\mathbf{t}}\mathbf{R} - \mathsf{bits}(\mathbf{s}, \mathsf{sd}) \otimes \mathbf{G},\ \ \mathbf{D} = \mathbf{t}^\intercal(\mathbf{A}_{\mathsf{path}} - (1, \mathsf{bits}(\mathbf{T})) \otimes \mathbf{G}) + \mathbf{e}_{\mathbf{D}}^\intercal, \\ \{\mathbf{c}_{0,i}^\intercal = \mathbf{s}_0^\intercal(\mathbf{A}_0\mathbf{r}_i) + \mathbf{e}_{0,i}^\intercal\ ,\ \mathbf{c}_{\mathsf{att},i}^\intercal = \mathbf{s}_{\mathbf{r}_i}^\intercal(\mathbf{A}_{\mathsf{att}} - (1, \mathbf{x}_i^\intercal) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{att},i}^\intercal\ ,\ \mathbf{S}_{\mathbf{r}_i} + \mathbf{e}_{\mathbf{S},i}^\intercal,\ \mathbf{E}_{\mathbf{r}_i} + \mathbf{e}_{\mathbf{E},i}^\intercal,\ \mathbf{r}_i\}_{i \in [Q]} \end{array} \right)$$

Hyb_2. This hybrid is same as Hyb_1, except we compute

$$\mathbf{c}_{0,i}^\intercal := \mathbf{c}_{\mathsf{msg}}^\intercal \cdot \mathbf{r}_i - \mathsf{UEvalCX}(\mathbf{A}_{\mathsf{att}}, \mathbf{c}_{\mathsf{att},i}^\intercal, \mathbf{A}_{\mathsf{circ}}, \mathbf{E}_{\mathbf{r}_i} + \mathbf{e}_{\mathbf{E},i}^\intercal, C, \mathbf{x}, \mathbf{S}_{\mathbf{r}_i}')\mathbf{u} - \mathbf{s}_{\mathbf{r}_i}^\intercal\mathbf{G}\mathbf{u} + \mathbf{e}_{0,i}$$

where $\mathbf{S}'_{\mathbf{r}_i} = \mathsf{Rnd}(\mathbf{S}_{\mathbf{r}_i} + \mathbf{e}^\intercal_{\mathbf{S},i})$. Note that since $\|\mathbf{e}_{\mathbf{S},i}\| \le \|\mathbf{e}_{1,i}\| \le 2^{2\lambda}\lambda^{\omega(1)} \le \frac{1}{2^\lambda} q/p$, Claim 3.2 implies that $\mathsf{Rnd}(\mathbf{S}_{\mathbf{r}_i} + \mathbf{e}^\intercal_{\mathbf{S},i}) = \mathsf{Rnd}(\mathbf{S}_{\mathbf{r}_i})$ with overwhelming probability, where $\mathsf{Rnd}(\mathbf{S}_{\mathbf{r}_i})$ is encoded in $\mathbf{E}_{\mathbf{r}_i}$.

We claim that Hyb_1 and Hyb_2 are statistically indistinguishable. To see this, note that:

- we have $\|\mathbf{s}_{\mathbf{r}_i}\| \le B$ and $\|\mathbf{e}_{\mathsf{att},i}\| \le 2^{5\lambda}\sqrt{\lambda} \le B$.
- Next, by the correctness of $\mathsf{UEvalCX}$, we have

$$\begin{aligned}
&\mathbf{c}^\intercal_{\mathsf{msg}} \cdot \mathbf{r}_i - \mathsf{UEvalCX}(\mathbf{A}_{\mathsf{att}}, \mathbf{c}^\intercal_{\mathsf{att},i}, \mathbf{A}_{\mathsf{circ}}, \mathbf{E}_{\mathbf{r}_i} + \mathbf{e}^\intercal_{\mathbf{E},i}, C, \mathbf{x}, \mathbf{S}_{\mathbf{r}_i} + \mathbf{e}^\intercal_{\mathbf{S},i})\mathbf{u} - \mathbf{s}^\intercal_{\mathbf{r}_i}\mathbf{G}\mathbf{u} \\
&= (\mathbf{s}_0^\intercal \mathbf{A}_0 + \mathbf{s}^\intercal(\mathbf{A}_C\mathbf{u} \otimes \mathbf{I}_m) + \mathbf{e}^\intercal_{\mathsf{msg}}) \cdot \mathbf{r}_i - \mathbf{s}(\mathbf{I}_{n+1} \otimes \mathbf{r})(\mathbf{A}_C - C(\mathbf{x})\mathbf{G})\mathbf{u} + \mathbf{e}^\intercal_{C,\mathbf{x}}\mathbf{u} - \mathbf{s}^\intercal_{\mathbf{r}_i}\mathbf{G}\mathbf{u} \\
&= (\mathbf{s}_0^\intercal \mathbf{A}_0 + \mathbf{s}^\intercal(\mathbf{A}_C\mathbf{u} \otimes \mathbf{I}_m) + \mathbf{e}^\intercal_{\mathsf{msg}}) \cdot \mathbf{r}_i - \mathbf{s}^\intercal(\mathbf{A}_C\mathbf{u} \otimes \mathbf{I}_m)(\mathbf{1} \otimes \mathbf{r}) + \mathbf{s}^\intercal_{\mathbf{r}_i}\mathbf{G}\mathbf{u} + \mathbf{e}^\intercal_{C,\mathbf{x}}\mathbf{u} - \mathbf{s}^\intercal_{\mathbf{r}_i}\mathbf{G}\mathbf{u} \\
&= \mathbf{s}_0^\intercal \mathbf{A}_0 \mathbf{r}_i + \mathbf{e}^\intercal_{\mathsf{msg}} \cdot \mathbf{r}_i + \mathbf{e}^\intercal_{C,\mathbf{x}}\mathbf{u}
\end{aligned}$$

  where $\|\mathbf{e}_{C,\mathbf{x}}\| \le (m+2)^{O(\log n \log\log q)} \cdot m \cdot (\|\mathsf{PRF}_1(\mathsf{sd}, \mathbf{r}_i)^\intercal\| + \|(\mathbf{s}^\intercal_{\mathbf{r}}(\mathbf{e}_{\mathbf{S}} + \mathbf{err}_2))^\intercal\|) + \|\mathbf{e}_{\mathsf{circ}}\| \cdot 2^{O(\log^5 \lambda)} \le 2^{6\lambda}$, by our parameter setting.
- We have $\mathbf{e}^\intercal_{0,i} \approx_s \mathbf{e}^\intercal_{\mathsf{msg}}\mathbf{r}_i + \mathbf{e}^\intercal_{C,\mathbf{x}}\mathbf{u} + \mathbf{e}^\intercal_{0,i}$ by noise flooding (Lemma 2.20) since $\|\mathbf{e}_{\mathsf{msg}}\|\|\mathbf{r}_i\| + \|\mathbf{e}_{C,\mathbf{x}}\|\|\mathbf{u}\| \le 2^{7\lambda}\lambda^{\omega(1)} = \chi_0$.

Thus it suffices to show pseudorandomness of the following, given aux_0,

$$\left(\begin{array}{c}
\mathsf{mpk} = (\bar{\mathbf{A}}_{\mathsf{fhe}}, \mathbf{A}_0, \mathbf{A}_{\mathsf{path}}, \mathbf{A}_{\mathsf{att}}, \mathbf{A}_{\mathsf{circ}}, \mathbf{B}, \mathbf{u}), \\
\mathbf{c}^\intercal_{\mathbf{B}} = (\mathbf{s}_0^\intercal \mid \mathbf{s}^\intercal \mid \mathbf{t}^\intercal)\mathbf{B} + \mathbf{e}^\intercal_{\mathbf{B}},\ \mathbf{c}^\intercal_{\mathsf{msg}} = \mathbf{s}_0^\intercal \mathbf{A}_0 + \mathbf{s}^\intercal(\mathbf{A}_C\mathbf{u} \otimes \mathbf{I}_m) + \mathbf{e}^\intercal_{\mathsf{msg}}, \\
\mathbf{T} = \mathbf{A}_{\mathbf{t}}\mathbf{R} - \mathsf{bits}(\mathbf{s}, \mathsf{sd}) \otimes \mathbf{G},\ \ \mathbf{D} = \mathbf{t}^\intercal(\mathbf{A}_{\mathsf{path}} - (1, \mathsf{bits}(\mathbf{T})) \otimes \mathbf{G}) + \mathbf{e}^\intercal_{\mathbf{D}}, \\
\{(\mathbf{c}^\intercal_{0,i})' = \mathbf{s}^\intercal_{\mathbf{r}_i}\mathbf{G}\mathbf{u} + \mathbf{e}_{0,i},\ \mathbf{c}^\intercal_{\mathsf{att},i} = \mathbf{s}^\intercal_{\mathbf{r}_i}(\mathbf{A}_{\mathsf{att}} - (1, \mathbf{x}_i^\intercal) \otimes \mathbf{G}) + \mathbf{e}^\intercal_{\mathsf{att},i},\ \mathbf{S}_{\mathbf{r}_i} + \mathbf{e}^\intercal_{\mathbf{S},i},\ \mathbf{E}_{\mathbf{r}_i} + \mathbf{e}^\intercal_{\mathbf{E},i},\ \mathbf{r}_i\}_{i \in [Q]}
\end{array}\right)$$

Hyb_3. This hybrid is same as Hyb_2, except we sample $\mathbf{c}_{\mathbf{B}} \leftarrow \mathbb{Z}_q^{(n+1)(m+2)w}, \mathbf{c}_{\mathsf{msg}} \leftarrow \mathbb{Z}_q^m$. Note that the LWE secret $\mathbf{s}_0$ does not appear anywhere else. We have $\mathsf{Hyb}_2 \approx_c \mathsf{Hyb}_3$, via the LWE assumption.

Thus it suffices to show pseudorandomness of the following distribution, given aux_0,

$$\left(\begin{array}{c}
\mathsf{mpk} = (\bar{\mathbf{A}}_{\mathsf{fhe}}, \mathbf{A}_0, \mathbf{A}_{\mathsf{path}}, \mathbf{A}_{\mathsf{att}}, \mathbf{A}_{\mathsf{circ}}, \mathbf{B}, \mathbf{u}),\ \mathbf{c}_{\mathbf{B}} \leftarrow \mathbb{Z}_q^{(n+1)(m+2)w},\ \mathbf{c}_{\mathsf{msg}} \leftarrow \mathbb{Z}_q^m, \\
\mathbf{T} = \mathbf{A}_{\mathbf{t}}\mathbf{R} - \mathsf{bits}(\mathbf{s}, \mathsf{sd}) \otimes \mathbf{G}, \mathbf{D} = \mathbf{t}^\intercal(\mathbf{A}_{\mathsf{path}} - (1, \mathsf{bits}(\mathbf{T})) \otimes \mathbf{G}) + \mathbf{e}^\intercal_{\mathbf{D}}, \\
\{(\mathbf{c}^\intercal_{0,i})' = \mathbf{s}^\intercal_{\mathbf{r}_i}\mathbf{G}\mathbf{u} + \mathbf{e}_{0,i},\ \mathbf{c}^\intercal_{\mathsf{att},i} = \mathbf{s}^\intercal_{\mathbf{r}_i}(\mathbf{A}_{\mathsf{att}} - (1, \mathbf{x}_i^\intercal) \otimes \mathbf{G}) + \mathbf{e}^\intercal_{\mathsf{att},i}\}_{i \in [Q]} \\
\{\mathbf{S}_{\mathbf{r}_i} = \mathsf{hct}_{\mathbf{s}_{\mathbf{r}_i}}(\mathsf{bits}(\mathbf{s}_{\mathbf{r}_i}); \mathsf{PRF}_2(\mathsf{sd}, \mathbf{r}_i)) + \mathbf{e}^\intercal_{\mathbf{S},i},\ \mathbf{E}_{\mathbf{r}_i} = \mathbf{s}^\intercal_{\mathbf{r}_i}(\mathbf{A}_{\mathsf{circ}} - (1, \mathsf{bits}(\mathbf{S}_{\mathbf{r}_i})) \otimes \mathbf{G}) + \mathsf{PRF}_3(\mathsf{sd}, \mathbf{r}_i) + \mathbf{e}^\intercal_{\mathbf{E},i},\ \mathbf{r}_i\}_{i \in [Q]}
\end{array}\right)$$

Hyb_4. This hybrid is same as Hyb_3, except we sample $\mathbf{A}_{\mathbf{t}} \leftarrow \mathbb{Z}_q^{(n+1)\times m},\ \mathbf{D} \leftarrow \mathbb{Z}_q^{1\times(\mathrm{L_T}+1)m}$. We have $\mathsf{Hyb}_3 \approx_c \mathsf{Hyb}_4$, via the LWE assumption. We show that if there exists an adversary $\mathcal{A}$ who can distinguish between the two hybrids with non-negligible advantage, then there is a reduction $\mathcal{B}$ that breaks LWE security with non-negligible advantage. The reduction is as follows.

1. The LWE challenger sends $\mathbf{A}_{\mathsf{lwe}} \in \mathbb{Z}_q^{n\times(\mathrm{L_T}+2)m}, \mathbf{b} \in \mathbb{Z}_q^{(\mathrm{L_T}+2)m}$ to $\mathcal{B}$.
2. $\mathcal{B}$ parses $\mathbf{A}_{\mathsf{lwe}} = (\bar{\mathbf{A}}'_{\mathsf{fhe}}\ \ \mathbf{A}'_{\mathsf{path}})$, where $\bar{\mathbf{A}}'_{\mathsf{fhe}} \in \mathbb{Z}_q^{n\times m}, \mathbf{A}'_{\mathsf{path}} \in \mathbb{Z}_q^{n\times(\mathrm{L_T}+1)m}$ and $\mathbf{b}^\intercal = [(\bar{\mathbf{b}}')^\intercal\ \ (\mathbf{b}'_1)^\intercal]$, where $\bar{\mathbf{b}}' \in \mathbb{Z}_q^m, \mathbf{b}'_1 \in \mathbb{Z}_q^{(\mathrm{L_T}+1)m}$ and does the following.

– Sets $\mathbf{A_t} = \begin{pmatrix} \bar{\mathbf{A}}'_{\mathsf{fhe}} \\ (\bar{\mathbf{b}}')^\intercal \end{pmatrix}$.

– Computes $\mathbf{T} = \mathbf{A_t R} - \mathsf{bits}(\mathbf{s}, \mathsf{sd}) \otimes \mathbf{G}$.

– Sets $\bar{\mathbf{A}}_{\mathsf{path}} = \mathbf{A}'_{\mathsf{path}} + (1, \mathsf{bits}(\mathbf{T})) \otimes \bar{\mathbf{G}}$ and $\mathbf{A}_{\mathsf{path}} = \begin{pmatrix} \bar{\mathbf{A}}_{\mathsf{path}} \\ \underline{\mathbf{a}}^\intercal_{\mathsf{path}} \end{pmatrix}$, where $\underline{\mathbf{a}}_{\mathsf{path}} \leftarrow \mathbb{Z}_q^{(\mathrm{L_T}+1)m}$.

– Sends $\mathbf{T},\ \mathbf{D} = (\mathbf{b}'_1)^\intercal - \left(\underline{\mathbf{a}}^\intercal_{\mathsf{path}} - (1, \mathsf{bits}(\mathbf{T})) \otimes \iota^\intercal_{n+1} \otimes \mathbf{g}\right)$ to $\mathcal{A}$.

3. The adversary outputs a bit β'. $\mathcal{B}$ forwards the bit β' to the LWE challenger.

We note that if the LWE challenger sent $\mathbf{b} = \mathbf{t}\mathbf{A}_{\mathsf{lwe}} + \mathbf{e}_{\mathsf{lwe}}$, then $\mathcal{B}$ simulated Hyb_3 with $\mathcal{A}$ else if LWE challenger sent random $\mathbf{b} \leftarrow \mathbb{Z}_q^{(\mathrm{L_T}+2)m}$ then $\mathcal{B}$ simulated Hyb_4 with $\mathcal{A}$.

To see the latter case, we note that if $\mathbf{b} \leftarrow \mathbb{Z}_q^{(\mathrm{L_T}+2)m}$ then it implies $\bar{\mathbf{b}}' \leftarrow \mathbb{Z}_q^m$ and $\mathbf{b}'_1 \leftarrow \mathbb{Z}_q^{(\mathrm{L_T}+1)m}$. Uniformity of $\bar{\mathbf{b}}' \leftarrow \mathbb{Z}_q^m$ implies $\mathbf{A_t} \leftarrow \mathbb{Z}_q^{(n+1)\times m}$. Randomness of $\mathbf{b}'_1$ implies randomness of $\mathbf{D} = (\mathbf{b}'_1)^\intercal - \left(\underline{\mathbf{a}}^\intercal_{\mathsf{path}} - (1, \mathsf{bits}(\mathbf{T})) \otimes \iota^\intercal_{n+1} \otimes \mathbf{g}\right)$.

Thus it suffices to show pseudorandomness of the following distribution, given aux_0,

$$\left(\begin{array}{c} \mathsf{mpk} = (\bar{\mathbf{A}}_{\mathsf{fhe}}, \mathbf{A}_0, \mathbf{A}_{\mathsf{path}}, \mathbf{A}_{\mathsf{att}}, \mathbf{A}_{\mathsf{circ}}, \mathbf{B}, \mathbf{u}),\ \mathbf{c}_{\mathbf{B}} \leftarrow \mathbb{Z}_q^{(n+1)(m+2)w},\ \mathbf{c}_{\mathsf{msg}} \leftarrow \mathbb{Z}_q^m, \\ \mathbf{T} = \mathbf{A_t R} - \mathsf{bits}(\mathbf{s}, \mathsf{sd}) \otimes \mathbf{G},\ \ \mathbf{D} \leftarrow \mathbb{Z}_q^{1\times(\mathrm{L_T}+1)m}, \\ \{(\mathbf{c}^\intercal_{0,i})' = \mathbf{s}^\intercal_{\mathbf{r}_i}\mathbf{G}\mathbf{u} + \mathbf{e}_{0,i},\ \mathbf{c}^\intercal_{\mathsf{att},i} = \mathbf{s}^\intercal_{\mathbf{r}_i}(\mathbf{A}_{\mathsf{att}} - (1, \mathbf{x}_i^\intercal) \otimes \mathbf{G}) + \mathbf{e}^\intercal_{\mathsf{att},i}\}_{i\in[Q]} \\ \{\mathbf{S}_{\mathbf{r}_i} = \mathsf{hct}_{\mathbf{s}_{\mathbf{r}_i}}(\mathsf{bits}(\mathbf{s}_{\mathbf{r}_i}); \mathsf{PRF}_2(\mathsf{sd}, \mathbf{r}_i)) + \mathbf{e}^\intercal_{\mathbf{S},i},\ \mathbf{E}_{\mathbf{r}_i} = \mathbf{s}^\intercal_{\mathbf{r}_i}(\mathbf{A}_{\mathsf{circ}} - (1, \mathsf{bits}(\mathbf{S}_{\mathbf{r}_i})) \otimes \mathbf{G}) + \mathsf{PRF}_3(\mathsf{sd}, \mathbf{r}_i) + \mathbf{e}^\intercal_{\mathbf{E},i},\ \mathbf{r}_i\}_{i\in[Q]} \end{array}\right)$$

Hyb_5. This hybrid is same as Hyb_4, except we sample $\mathbf{T} \leftarrow \mathbb{Z}_q^{(n+1)\times m_{\mathrm{T}}}$. We have $\mathsf{Hyb}_4 \approx_s \mathsf{Hyb}_5$ using leftover hash lemma. By leftover hash lemma (Lemma 2.21) we have that the statistical distance between $\mathbf{A_t R}$ and a uniform matrix $U \leftarrow \mathbb{Z}_q^{(n+1)\times m(\lambda+\mathrm{L})}$ is negligible. This implies that the statistical distance between $\mathbf{A_t R} - \mathsf{bits}(\mathbf{s}, \mathsf{sd}) \otimes \mathbf{G}$ and $\mathbf{T} \leftarrow \mathbb{Z}_q^{(n+1)\times m_{\mathrm{T}}}$ is negligible. Thus it suffices to show pseudorandomness of the following distribution, given aux_0,

$$\left(\begin{array}{c} \mathsf{mpk} = (\bar{\mathbf{A}}_{\mathsf{fhe}}, \mathbf{A}_0, \mathbf{A}_{\mathsf{path}}, \mathbf{A}_{\mathsf{att}}, \mathbf{A}_{\mathsf{circ}}, \mathbf{B}, \mathbf{u}),\ \mathbf{c}_{\mathbf{B}} \leftarrow \mathbb{Z}_q^{(n+1)(m+2)w},\ \mathbf{c}_{\mathsf{msg}} \leftarrow \mathbb{Z}_q^m, \\ \mathbf{T} \leftarrow \mathbb{Z}_q^{(n+1)\times m_{\mathrm{T}}},\ \ \mathbf{D} \leftarrow \mathbb{Z}_q^{1\times(\mathrm{L_T}+1)m}, \\ \{(\mathbf{c}^\intercal_{0,i})' = \mathbf{s}^\intercal_{\mathbf{r}_i}\mathbf{G}\mathbf{u} + \mathbf{e}_{0,i},\ \mathbf{c}^\intercal_{\mathsf{att},i} = \mathbf{s}^\intercal_{\mathbf{r}_i}(\mathbf{A}_{\mathsf{att}} - (1, \mathbf{x}_i^\intercal) \otimes \mathbf{G}) + \mathbf{e}^\intercal_{\mathsf{att},i}\}_{i\in[Q]} \\ \{\mathbf{S}_{\mathbf{r}_i} = \mathsf{hct}_{\mathbf{s}_{\mathbf{r}_i}}(\mathsf{bits}(\mathbf{s}_{\mathbf{r}_i}); \mathsf{PRF}_2(\mathsf{sd}, \mathbf{r}_i)) + \mathbf{e}^\intercal_{\mathbf{S},i},\ \mathbf{E}_{\mathbf{r}_i} = \mathbf{s}^\intercal_{\mathbf{r}_i}(\mathbf{A}_{\mathsf{circ}} - (1, \mathsf{bits}(\mathbf{S}_{\mathbf{r}_i})) \otimes \mathbf{G}) + \mathsf{PRF}_3(\mathsf{sd}, \mathbf{r}_i) + \mathbf{e}^\intercal_{\mathbf{E},i},\ \mathbf{r}_i\}_{i\in[Q]} \end{array}\right)$$

Hyb_6: In this hybrid, we change all the PRF values computed using sd to random.

We claim that an adversary who can distinguish between Hyb_5 and Hyb_6 can be used to break PRF security[8]. Intuitively, this is because, the PRF seed sd is now no longer used in the distribution.

In more detail, the reduction queries the $\mathsf{PRF} = (\mathsf{PRF}_1, \mathsf{PRF}_2, \mathsf{PRF}_3)$ challenger with inputs $\mathbf{r}_i$ for $i \in [Q]$ and obtains real or random outputs $\mathbf{y}^\intercal_{1,i} \in [-\sigma', \sigma']^m$, $\mathbf{y}_{2,i} \in \{0,1\}^{m^2 n\lceil \log q\rceil}$, $\mathbf{y}^\intercal_{3,i} \in [-\sigma', \sigma']^{(\mathrm{L_S}+1)m}$. It embeds these into the construction of $\underline{\mathbf{a}}_{\mathbf{r}_i}$,$\mathbf{S}_{\mathbf{r}_i}$, $\mathbf{E}_{\mathbf{r}_i}$ as follows:

– Computes $\underline{\mathbf{a}}^\intercal_{\mathbf{r}_i} = \bar{\mathbf{s}}^\intercal_{\mathbf{r}_i}\bar{\mathbf{A}}_{\mathsf{fhe}} + \mathbf{y}_{1,i}$.

[8]We assume that a single PRF challenger has both PRF_1 and PRF_2.

– Computes $\mathbf{S}_{\mathbf{r}_i} = \begin{pmatrix} \bar{\mathbf{A}}_{\mathsf{fhe}} \\ \underline{\mathbf{a}}_{\mathbf{r}}^{\mathsf{T}} \end{pmatrix} \mathbf{y}_{2,i} - \mathsf{bits}(\mathbf{s}_{\mathbf{r}}) \otimes \mathbf{G}$, and $\bar{\mathbf{S}}_{\mathbf{r}_i} = \mathsf{Rnd}(\mathbf{S}_{\mathbf{r}_i})$.

– Computes $\mathbf{E}_{\mathbf{r}_i} = \mathbf{s}_{\mathbf{r}_i}^{\mathsf{T}}(\mathbf{A}_{\mathsf{circ}} - (1, \mathsf{bits}(\bar{\mathbf{S}}_{\mathbf{r}_i})) \otimes \mathbf{G}) + \mathbf{y}_{3,i} + \mathbf{e}_{\mathbf{E},i}^{\mathsf{T}}$.

Thus it suffices to show pseudorandomness of the following distribution, given aux_0,

$$\left(\begin{array}{c} \mathsf{mpk} = (\bar{\mathbf{A}}_{\mathsf{fhe}}, \mathbf{A}_0, \mathbf{A}_{\mathsf{path}}, \mathbf{A}_{\mathsf{att}}, \mathbf{A}_{\mathsf{circ}}, \mathbf{B}, \mathbf{u}),\ \mathbf{c}_{\mathbf{B}} \leftarrow \mathbb{Z}_q^{(n+1)(m+2)w},\ \mathbf{c}_{\mathsf{msg}} \leftarrow \mathbb{Z}_q^m, \\ \mathbf{T} \leftarrow \mathbb{Z}_q^{(n+1)\times m_{\mathrm{T}}},\ \ \mathbf{D} \leftarrow \mathbb{Z}_q^{1\times(\mathrm{L}_{\mathrm{T}}+1)m}, \\ \{(\mathbf{c}_{0,i}^{\mathsf{T}})' = \mathbf{s}_{\mathbf{r}_i}^{\mathsf{T}}\mathbf{G}\mathbf{u} + \mathbf{e}_{0,i},\ \mathbf{c}_{\mathsf{att},i}^{\mathsf{T}} = \mathbf{s}_{\mathbf{r}_i}^{\mathsf{T}}(\mathbf{A}_{\mathsf{att}} - (1, \mathbf{x}_i^{\mathsf{T}}) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{att},i}^{\mathsf{T}}\}_{i\in[Q]} \\ \{\mathbf{S}_{\mathbf{r}_i} = \mathsf{hct}_{\mathbf{s}_{\mathbf{r}_i}}(\mathsf{bits}(\mathbf{s}_{\mathbf{r}_i})) + \mathbf{e}_{\mathbf{S},i}^{\mathsf{T}},\ \mathbf{E}_{\mathbf{r}_i} = \mathbf{s}_{\mathbf{r}_i}^{\mathsf{T}}(\mathbf{A}_{\mathsf{circ}} - (1, \mathsf{bits}(\mathbf{S}_{\mathbf{r}_i})) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{circ},i}^{\mathsf{T}} + \mathbf{e}_{\mathbf{E},i}^{\mathsf{T}},\ \mathbf{r}_i\}_{i\in[Q]} \end{array} \right)$$

where $\mathbf{e}_{\mathsf{circ},i} \leftarrow [-\sigma'\ ,\ \sigma']^{(\mathrm{L}_{\mathrm{S}}+1)m}$.

Hyb_7: In this hybrid, we apply circular tensor LWE (Assumption 4.1). We claim $\mathsf{Hyb}_6 \approx_c \mathsf{Hyb}_7$. To see this, first we observe the following using $\mathbf{s}_{\mathbf{r}_i}^{\mathsf{T}} = (\bar{\mathbf{s}}_{\mathbf{r}_i}^{\mathsf{T}}, -1)$

– $\mathbf{E}_{\mathbf{r}_i} = \mathbf{s}_{\mathbf{r}_i}^{\mathsf{T}}(\mathbf{A}_{\mathsf{circ}} - (1, \mathsf{bits}(\mathbf{S}_{\mathbf{r}_i})) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{circ},i}^{\mathsf{T}} + \mathbf{e}_{\mathbf{E},i}^{\mathsf{T}}$
$= \left(\bar{\mathbf{s}}_{\mathbf{r}_i}^{\mathsf{T}}(\bar{\mathbf{A}}_{\mathsf{circ}} - (1, \mathsf{bits}(\mathbf{S}_{\mathbf{r}_i})) \otimes \bar{\mathbf{G}}) + \mathbf{e}_{\mathsf{circ},i}^{\mathsf{T}}\right) - \left(\mathbf{a}_{\mathsf{circ}}^{\mathsf{T}} - (1, \mathsf{bits}(\mathbf{S}_{\mathbf{r}_i})) \otimes \iota_{n+1}^{\mathsf{T}} \otimes \mathbf{g}\right) + \mathbf{e}_{\mathbf{E},i}^{\mathsf{T}}$.

– $\mathbf{c}_{\mathsf{att},i}^{\mathsf{T}} = \mathbf{s}_{\mathbf{r}_i}^{\mathsf{T}}(\mathbf{A}_{\mathsf{att}} - (1, \mathbf{x}_i^{\mathsf{T}}) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{att},i}^{\mathsf{T}}$
$= \left(\bar{\mathbf{s}}_{\mathbf{r}_i}^{\mathsf{T}}(\bar{\mathbf{A}}_{\mathsf{att}} - (1, \mathbf{x}_i^{\mathsf{T}}) \otimes \bar{\mathbf{G}}) + \mathbf{e}_{\mathsf{att},i}^{\mathsf{T}}\right) - \left(\mathbf{a}_{\mathsf{att}}^{\mathsf{T}} - (1, \mathbf{x}^{\mathsf{T}}) \otimes \iota_{n+1}^{\mathsf{T}} \otimes \mathbf{g}\right)$.

Next we note that

– We can change $\mathbf{S}_{\mathbf{r}}$ to random if $\mathsf{hct}(\mathbf{A}_{\mathbf{s}_{\mathbf{r}_i}}, \mathsf{bits}(\mathbf{s}_{\mathbf{r}_i}))$ is indistinguishable from random.

– We can change $\mathbf{E}_{\mathbf{r}}$ to random if $\bar{\mathbf{s}}_{\mathbf{r}_i}^{\mathsf{T}}(\bar{\mathbf{A}}_{\mathsf{circ}} - (1, \mathsf{bits}(\mathbf{S}_i)) \otimes \bar{\mathbf{G}}) + \mathbf{e}_{\mathsf{circ},i}^{\mathsf{T}}$ and $\mathbf{S}_{\mathbf{r}_i}$ are indistinguishable from random random.

– We can change $\mathbf{c}_{\mathsf{att},i}^{\mathsf{T}}$ to random if $\bar{\mathbf{s}}_{\mathbf{r}_i}^{\mathsf{T}}(\bar{\mathbf{A}}_{\mathsf{att}} - (1, \mathbf{x}_i^{\mathsf{T}}) \otimes \bar{\mathbf{G}}) + \mathbf{e}_{\mathsf{att},i}^{\mathsf{T}}$ is indistinguishable from random.

Invoking circular tensor LWE assumption with respect to secret $\mathbf{s}_{\mathbf{r}_i}, \mathbf{r}_i$, we achieve randomness of the latter terms in the above.

Thus it suffices to show pseudorandomness of the following distribution, given aux_0,

$$\left(\begin{array}{c} \mathsf{mpk} = (\bar{\mathbf{A}}_{\mathsf{fhe}}, \mathbf{A}_0, \mathbf{A}_{\mathsf{path}}, \mathbf{A}_{\mathsf{att}}, \mathbf{A}_{\mathsf{circ}}, \mathbf{B}, \mathbf{u}),\ \mathbf{c}_{\mathbf{B}} \leftarrow \mathbb{Z}_q^{(n+1)(m+2)w},\ \mathbf{c}_{\mathsf{msg}} \leftarrow \mathbb{Z}_q^m, \\ \mathbf{T} \leftarrow \mathbb{Z}_q^{(n+1)\times m_{\mathrm{T}}},\ \ \mathbf{D} \leftarrow \mathbb{Z}_q^{1\times(\mathrm{L}_{\mathrm{T}}+1)m}, \\ \{(\mathbf{c}_{0,i}^{\mathsf{T}})' \leftarrow \mathbb{Z}_q,\ \mathbf{c}_{\mathsf{att},i}^{\mathsf{T}} \leftarrow \mathbb{Z}_q^{(\mathrm{L}+1)m},\ \mathbf{S}_{\mathbf{r}_i} \leftarrow \mathbb{Z}_q^{(n+1)\times m_{\mathrm{S}}},\ \mathbf{E}_{\mathbf{r}_i} \leftarrow \mathbb{Z}_q^{1\times(\mathrm{L}_{\mathrm{S}}+1)m},\ \mathbf{r}_i\}_{i\in[Q]} \end{array} \right)$$

which completes the proof.

□

Theorem 4.4. Under the LWE assumption, evasive LWE assumption (Assumption 2.24) and circular tensor assumption (Assumption 4.1), there exists a very selectively secure CP-ABE scheme supporting unbounded depth circuits $\{C : \{0,1\}^L \to \{0,1\}\}$ and one bit message with efficiency

$$|\mathsf{mpk}| = \mathrm{poly}(\lambda, L), \quad |\mathsf{sk}| = \mathrm{poly}(\lambda, L), \quad |\mathsf{ct}| = \mathrm{poly}(\lambda).$$

5 Generic compiler: ABE for Turing Machines and NL

For conciseness, we first provide a generic construction of ABE that combines many instance of ABE together in Section 5.1. This compiler is adapted from [AMVY21], who gave it for FE in the bounded key setting. We then construct ABE for TM in Section 5.2 and ABE for NL in Section 5.3 by instantiating the generic construction.

5.1 Generalized Bundling of Functionality

Consider an ABE scheme $\mathsf{ABE} = (\mathsf{ABE.Setup}, \mathsf{ABE.KeyGen}, \mathsf{ABE.Enc}, \mathsf{ABE.Dec})$ for a parameter $\mathsf{prm} = 1^i$, a relation $R_i : \mathcal{X}_i \times \mathcal{Y}_i \to \{0,1\} \cup \{\bot\}$ for all $i \in \mathbb{N}$ and a message space $\mathcal{M}$. Using such ABE, we will construct a new ABE with ciphertext attribute space A, key attribute space B and message space $\mathcal{M}$. We assume that there exist efficiently computable maps $\mathcal{S} : \mathbb{N} \to 2^{\mathbb{N}}$ and $\mathcal{T} : \mathbb{N} \to 2^{\mathbb{N}}$ such that $\max\mathcal{S}(n)$ and $\max\mathcal{T}(n)$ can be bounded by some fixed polynomial in n. We also assume that there exist maps f with domain A and g with domain B such that

$$f(x) \in \prod_{i \in \mathcal{S}(|x|)} \mathcal{X}_i \quad \text{and} \quad g(y) \in \prod_{i \in \mathcal{T}(|y|)} \mathcal{Y}_i,$$

where $|x|$ and $|y|$ are the lengths of x and y as binary strings. Namely, f and g are maps such that

$$f : \mathrm{A} \ni x \mapsto \{f(x)_i \in \mathcal{X}_i\}_{i \in \mathcal{S}(|x|)}, \quad g : \mathrm{B} \ni y \mapsto \{g(y)_i \in \mathcal{Y}_i\}_{i \in \mathcal{T}(|y|)}.$$

Here, we require that the length of $|f(x)|_i$ and $|g(x)|_i$ can be computed from the length of $|x|$ alone and they do not depend on the actual value of x. In this setting, we can construct an ABE scheme $\mathsf{Bd\text{-}ABE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ for a two input function $R^{\mathsf{bndl}} : \mathrm{A} \times \mathrm{B} \to \{0,1\}^*$ defined in the following

$$R^{\mathsf{bndl}}(x,y) = \begin{cases} 0, & \text{if } R_i(f(x)_i, g(y)_i) = 0 \text{ for all } i \in \mathcal{S}(|x|) \cap \mathcal{T}(|y|), \\ 1, & \text{if } R_i(f(x)_i, g(y)_i) = 1 \text{ for all } i \in \mathcal{S}(|x|) \cap \mathcal{T}(|y|). \end{cases} \tag{2}$$

where $f(x)_i \in \mathcal{X}_i$ and $g(y)_i \in \mathcal{Y}_i$ are the i-th entries of $f(x)$ and $g(x)$, respectively.

Ingredients. We now describe the underlying building blocks used to obtain our ABE construction:

1. An ABE scheme $\mathsf{ABE} = (\mathsf{ABE.Setup}, \mathsf{ABE.KeyGen}, \mathsf{ABE.Enc}, \mathsf{ABE.Dec})$ for a parameter $\mathsf{prm} = 1^i$, a relation $R_i : \mathcal{X}_i \times \mathcal{Y}_i \to \{0,1\} \cup \{\bot\}$ for $i \in \mathbb{N}$ and a message space $\mathcal{M}$.
2. A garbled circuit scheme $\mathsf{GC} = (\mathsf{GC.Garble}, \mathsf{GC.Eval})$. We assume that a label is represented by a binary string and denote its length by $L(\lambda, |C|)$, where C is the circuit being garbled. We can instantiate it by Yao's garbled circuit [Yao82], which can be based on any one-way function.
3. An IBE scheme $\mathsf{IBE} = (\mathsf{IBE.Setup}, \mathsf{IBE.Enc}, \mathsf{IBE.KeyGen}, \mathsf{IBE.Dec})$ with IND-CPA security whose identity space and message space are $\{0,1\}^*$. We assume that the key generation algorithm is deterministic. This is without loss of generality, since we can use PRF to derandomize the key generation algorithm. We can instantiate IBE from various standard assumptions including LWE [ABB10a, CHKP10], CDH, and Factoring [DG17].

Construction. Here we provide the description of the construction of $\mathsf{Bd\text{-}ABE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ for R^{bndl} above.

$\mathsf{Setup}(1^\lambda) \to (\mathsf{mpk}, \mathsf{msk})$. On input the security parameter λ, do the following.

1. Run $(\mathsf{IBE.mpk}, \mathsf{IBE.msk}) \leftarrow \mathsf{IBE.Setup}(1^\lambda)$.
2. Output the master key pair as $(\mathsf{mpk}, \mathsf{msk}) := (\mathsf{IBE.mpk}, \mathsf{IBE.msk})$.

$\mathsf{KeyGen}(\mathsf{msk}, y) \to \mathsf{sk}_y$. On input master secret key $\mathsf{msk} = \mathsf{IBE.msk}$, a key attribute $y \in \mathrm{B}$, do the following.

1. Compute $\mathcal{T}(|y|) \subseteq \mathbb{N}$, where $|y|$ is the length of y as a binary string.
2. Run $(\mathsf{ABE.mpk}_i, \mathsf{ABE.msk}_i) \leftarrow \mathsf{ABE.Setup}(1^\lambda, 1^i)$ for $i \in \mathcal{T}(|y|)$.
3. Compute $g(y) = \{g(y)_i \in \mathcal{Y}_i\}_{i \in \mathcal{T}(|y|)}$.
4. For $i \in \mathcal{T}(|y|)$, compute
$$\mathsf{ABE.sk}_i \leftarrow \mathsf{ABE.KeyGen}(\mathsf{ABE.msk}_i, g(y)_i).$$
5. Let $\ell_i := |\mathsf{ABE.mpk}_i|$. For all $i \in \mathcal{T}(|y|)$ and $j \in \ell_i$, generate a secret key as
$$\mathsf{IBE.sk}_{i,j} \leftarrow \mathsf{IBE.KeyGen}(\mathsf{IBE.msk}, (i, j, \mathsf{ABE.mpk}_{i,j}))$$
where $\mathsf{ABE.mpk}_{i,j}$ is the j-th bit of $\mathsf{ABE.mpk}_i \in \{0,1\}$ as a binary string.
6. Output
$$\mathsf{sk}_y = \left(\mathcal{T}(|y|),\ \left\{\mathsf{ABE.sk}_i, g(y)_i, \left\{\mathsf{IBE.sk}_{i,j}\right\}_{j \in [\ell_i]}\right\}_{i \in \mathcal{T}(|y|)}\right). \tag{3}$$

$\mathsf{Enc}(\mathsf{mpk}, x, \mu) \to \mathsf{ct}_x$. On input the encryption key $\mathsf{mpk} = \mathsf{IBE.mpk}$, a ciphertext attribute $x \in \mathrm{A}$, and a message $\mu \in \mathcal{M}$, do the following.

1. Compute $\mathcal{S}(|x|) \subset \mathbb{N}$, where $|x|$ is the length of x as a binary string.
2. Compute $f(x) = \{f(x)_i\}_{i \in \mathcal{S}(|x|)}$.
3. Do the following for all $i \in \mathcal{S}(|x|)$.
   (a) Compute the length ℓ_i of $\mathsf{ABE.mpk}_i$. This is done without knowing $\mathsf{ABE.mpk}_i$.
   (b) Sample a randomness r_i for the encryption algorithm $\mathsf{ABE.Enc}(\mathsf{ABE.mpk}_i, f(x)_i, \mu; r_i)$. This is done without knowing $\mathsf{ABE.mpk}_i$.
   (c) Define a circuit
   $$\mathsf{E}_i(\cdot) := \mathsf{ABE.Enc}(\cdot, f(x)_i, \mu; r_i)$$
   that takes as input a string $\mathsf{str} \in \{0,1\}^{\ell_i}$ and outputs $\mathsf{ABE.Enc}(\mathsf{str}, f(x)_i, \mu_i; r_i)$, where str is interpreted as a master public key of the ABE.
   (d) Generate a garbled circuit
   $$\left\{\mathsf{lab}_{i,j,b}\right\}_{j \in [\ell_i], b \in \{0,1\}} \leftarrow \mathsf{GC.Garble}(1^\lambda, \mathsf{E}_i).$$
   (e) For all $j \in [\ell_i]$ and $b \in \{0,1\}$, compute
   $$\mathsf{IBE.ct}_{i,j,b} \leftarrow \mathsf{IBE.Enc}\left(\mathsf{IBE.mpk}, (i, j, b), \mathsf{lab}_{i,j,b}\right).$$
4. Output
$$\mathsf{ct}_x = \left(\mathcal{S}(|x|), \{f(x)_i\}_{i \in \mathcal{S}(|x|)},\ \left\{\mathsf{IBE.ct}_{i,j,b}\right\}_{i \in \mathcal{S}(|x|), j \in [\ell_i], b \in \{0,1\}}\right). \tag{4}$$

$\mathsf{Dec}(\mathsf{sk}_y, y, \mathsf{ct}_x, x) \to \mu' \setminus \perp$. On input a secret key sk_y, key attribute y, a ciphertext ct_x, and ciphertext attribute x, do the following.

1. Parse the secret key sk_y as Eq. (3) and the ciphertext ct_x as Eq. (4).
2. For all $i \in \mathcal{S}(|x|) \cap \mathcal{T}(|y|)$ do the following.
   (a) Retrieve $\mathsf{ABE.mpk}_i$ from the $\mathsf{ABE.sk}_i$.
   (b) For all $j \in [\ell_i]$ compute $\mathsf{lab}'_{i,j} := \mathsf{IBE.Dec}(\mathsf{IBE.sk}_{i,j,\mathsf{ABE.mpk}_{i,j}}, \mathsf{IBE.ct}_{i,j,\mathsf{ABE.mpk}_{i,j}})$.
   (c) Compute $c_i := \mathsf{GC.Eval}(\{\mathsf{lab}'_{i,j}\}_{j \in [\ell_i]})$.
   (d) Compute $z_i := \mathsf{ABE.Dec}(\mathsf{ABE.sk}_i, g(y)_i, c_i, f(x)_i)$.
3. Output $\{z_i\}_{i \in \mathcal{S}(|x|) \cap \mathcal{T}(|y|)}$.

Correctness. We now show that the above construction is correct via the following theorem.

Theorem 5.1. Suppose ABE, GC and IBE schemes are correct. Then the Bd-ABE scheme is correct.

Proof. We observe that $\mathsf{lab}'_{i,j} = \mathsf{lab}_{i,j,\mathsf{ABE.mpk}_{i,j}}$ holds for all $\mathsf{lab}'_{i,j}$ recovered in Step 2b of the decryption algorithm by the correctness of IBE, since the ciphertext and secret key are both generated with respect to the identity $(i, j, \mathsf{ABE.mpk}_{i,j})$. Then, by the correctness of GC, we have $c_i = \mathsf{ABE.Enc}(\mathsf{ABE.mpk}_i, f(x)_i, \mu; r_i)$ for all $i \in \mathcal{S}(|x|) \cap \mathcal{T}(|y|)$ recovered in Step 2c of the decryption algorithm. Finally, if $R^{\mathsf{bndl}}(x, y) = 0$, then by the definition of R^{bndl}, we have $R_i(f(x)_i, g(y)_i) = 0$ for all $i \in \mathcal{S}(|x|) \cap \mathcal{T}(|y|)$ and thus by the correctness of ABE, we get $z_i = \mu$ for all z_i recovered in Step 2d of the decryption algorithm as desired. □

Security. We prove the security of the Bd-ABE scheme via the following theorem.

Theorem 5.2. Suppose GC is a secure garbled circuit scheme and IBE is IND-CPA secure identity-based encryption scheme. Then, assuming ABE is Sel-IND secure (Definition 2.14), so is Bd-ABE. Furthermore, if ABE is VerSel-IND secure (Definition 2.15), so is Bd-ABE[9].

Proof. Here we prove the statement for Sel-IND security. The proof proceeds via a sequence of hybrid games between the challenger and a PPT adversary $\mathcal{A}$.

Hyb_0. This is the real world with $\beta = 0$, i.e., the challenge ciphertext is computed using the message μ_0. We write the complete game here to set up the notations and easy reference in later hybrids.

1. $\mathcal{A}$ outputs the challenge ciphertext attribute $x \in \mathrm{A}$.
2. The challenger generates $(\mathsf{IBE.mpk}, \mathsf{IBE.msk}) \leftarrow \mathsf{IBE.Setup}(1^\lambda)$, sets $\mathsf{mpk} = \mathsf{IBE.mpk}$ and sends it to $\mathcal{A}$.
3. **Key Queries:** The adversary can make key queries, before and after challenge query, in an arbitrary order. For each key query $y \in \mathrm{B}$, the challenger does the following.
   - It computes the mapping $\mathcal{T}(|y|)$ and generates $(\mathsf{ABE.mpk}_i, \mathsf{ABE.msk}_i) \leftarrow \mathsf{ABE.Setup}(1^\lambda, 1^i)$ for $i \in \mathcal{T}(|y|)$.
   - It computes the mapping $g(y)$ and generates $\mathsf{ABE.sk}_i \leftarrow \mathsf{ABE.KeyGen}(\mathsf{ABE.msk}_i, g(y)_i)$ for all $i \in \mathcal{T}(|y|)$.
   - It sets $\ell_i := |\mathsf{ABE.mpk}_i|$ and computes $\mathsf{IBE.sk}_{i,j} \leftarrow \mathsf{IBE.KeyGen}(\mathsf{IBE.msk}, (i, j, \mathsf{ABE.mpk}_{i,j}))$ for all $i \in \mathcal{T}(|y|)$ and $j \in \ell_i$.
   - It returns $\mathsf{sk}_y = (\mathcal{T}(|y|), g(y), \{\mathsf{ABE.sk}_i\}_{i \in \mathcal{T}(|y|)}, \{\mathsf{IBE.sk}_{i,j}\}_{i \in \mathcal{T}(|y|), j \in \ell_i})$.
4. **Challenge Query:** $\mathcal{A}$ outputs a pair of equal length messages $(\mu_0, \mu_1) \in \mathcal{M}^2$. The challenger does the following.
   - It computes $\mathcal{S}(|x|)$ and $f(x) = \{f(x)_i\}_{i \in \mathcal{S}(|x|)}$.
   - For all $i \in \mathcal{S}(|x|)$,
     - It samples the randomness r_i and defines the circuit $\mathsf{E}_i(\cdot) := \mathsf{ABE.Enc}(\cdot, f(x)_i, \mu_0; r_i)$ as in the construction.
     - It generates $\{\mathsf{lab}_{i,j,b}\}_{j \in [\ell_i], b \in \{0,1\}} \leftarrow \mathsf{GC.Garble}(1^\lambda, \mathsf{E}_i)$.
     - It computes $\mathsf{IBE.ct}_{i,j,b} \leftarrow \mathsf{IBE.Enc}(\mathsf{IBE.mpk}, (i, j, b), \mathsf{lab}_{i,j,b})$.
   - It returns $\mathsf{ct}_x = (\mathcal{S}(|x|), f(x), \{\mathsf{IBE.ct}_{i,j,b}\}_{i \in \mathcal{S}(|x|), j \in [\ell_i], b \in \{0,1\}})$.
5. $\mathcal{A}$ outputs a guess bit β'.

Hyb_1. In this hybrid, we change the way challenge query is answered. In particular, the challenger computes $\mathsf{IBE.ct}_{i,j,b} \leftarrow \mathsf{IBE.Enc}(\mathsf{IBE.mpk}, (i, j, b), \mathsf{lab}_{i,j,\mathsf{ABE.mpk}_{i,j}})$, for all $i \in \mathcal{S}(|x|), j \in [\ell_i]$, and $b \in \{0, 1\}$, where $\mathsf{ABE.mpk}_{i,j}$ is the j-th bit of $\mathsf{ABE.mpk}_i$.
Note that we encrypt the same label for both $b = 0$ and $b = 1$.

[9] In the latter case, the adversary can see the public key before declaring all the key queries.

Hyb_2. In this hybrid, we further change the way challenge query is answered. In particular, we change the way $\mathsf{lab}_{i,j,\mathsf{ABE.mpk}_{i,j}}$ is generated. The challenger does the following for all $i \in \mathcal{S}(|x|)$,

- It computes $\mathsf{ABE.ct}_i \leftarrow \mathsf{ABE.Enc}(\mathsf{ABE.mpk}_i, f(x)_i, \mu_0)$.
- It generates $\{\mathsf{lab}_{i,j}\}_{j\in[\ell_i]} \leftarrow \mathsf{GC.Sim}(1^\lambda, 1^{\ell_i}, 1^{|E_i(\cdot)|}, \mathsf{ABE.ct}_i)$.
- It sets $\mathsf{lab}_{i,j,\mathsf{ABE.mpk}_{i,j}} = \{\mathsf{lab}_{i,j}\}$ for all $j \in [\ell_i]$.

Hyb_3. In this hybrid, we further change the way challenge query is answered. In particular, the challenger computes $\mathsf{ABE.ct}_i \leftarrow \mathsf{ABE.Enc}(\mathsf{ABE.mpk}_i, f(x)_i, \mu_1)$ for all $i \in \mathcal{S}(|x|)$.

The following hybrids are unwinding of the preceding hybrids.

Hyb_4. In this hybrid, we change the way challenge query is answered. In particular, we change the way $\mathsf{lab}_{i,j,\mathsf{ABE.mpk}_{i,j}}$ is generated. The challenger does the following for all $i \in \mathcal{S}(|x|)$.

- It samples an encryption randomness r_i and define $\mathsf{E}_i(\cdot) = \mathsf{ABE.Enc}(\cdot, f(x)_i, \mu_1; r_i)$.
- It generates $\{\mathsf{lab}_{i,j,b}\}_{j\in[\ell_i],b\in\{0,1\}} \leftarrow \mathsf{GC.Garble}(1^\lambda, \mathsf{E}_i)$.
- It sets $\mathsf{lab}_{i,j,\mathsf{ABE.mpk}_{i,j}} = \{\mathsf{lab}_{i,j,b}\}$ for all $j \in [\ell_i]$, where $b = \mathsf{ABE.mpk}_{i,j}$.

Hyb_5. In this hybrid, we further change the way challenge query is answered. In particular, the challenger computes $\mathsf{IBE.ct}_{i,j,b} \leftarrow \mathsf{IBE.Enc}(\mathsf{IBE.mpk}, (i,j,b), \mathsf{lab}_{i,j,b})$, for all $i \in \mathcal{S}(|x|), j \in [\ell_i]$, and $b \in \{0,1\}$.
This is the real world with $\beta = 1$.

Indistinguishability of hybrids We now show that the consecutive hybrids are indistinguishable.

Claim 5.3. Assume that IBE is IND-CPA secure. Then $\mathsf{Hyb}_0 \approx_c \mathsf{Hyb}_1$.

Proof. We show that if $\mathcal{A}$ can distinguish between Hyb_0 and Hyb_1 with non-negligible advantage ϵ, then there exists a PPT adversary $\mathcal{B}$ against the IND-CPA security of IBE scheme with advantage ϵ. The reduction is as follows.

1. The IBE challenger generates $(\mathsf{IBE.mpk}, \mathsf{IBE.msk}) \leftarrow \mathsf{IBE.Setup}(1^\lambda)$ and samples a bit $\hat{\beta} \leftarrow \{0,1\}$. It sends $\mathsf{IBE.mpk}$ to $\mathcal{B}$.
2. $\mathcal{B}$ invokes $\mathcal{A}$. $\mathcal{A}$ outputs the challenge ciphertext attribute $x \in$ A. $\mathcal{B}$ sets $\mathsf{mpk} = \mathsf{IBE.mpk}$ and forwards it to $\mathcal{A}$.
3. **Key Queries:** For a key query $y \in$ B, $\mathcal{B}$ does the following.
   - It computes the mapping $\mathcal{T}(|y|)$ and generates $(\mathsf{ABE.mpk}_i, \mathsf{ABE.msk}_i) \leftarrow \mathsf{ABE.Setup}(1^\lambda, 1^i)$ for $i \in \mathcal{T}(|y|)$.
   - It computes the mapping $g(y)$ and generates $\mathsf{ABE.sk}_i \leftarrow \mathsf{ABE.KeyGen}(\mathsf{ABE.msk}_i, g(y)_i)$ for all $i \in \mathcal{T}(|y|)$.
   - It sets $\ell_i := |\mathsf{ABE.mpk}_i|$ and sends secret key query for identity $(i, j, \mathsf{ABE.mpk}_{i,j})$ for all $i \in \mathcal{T}(|y|)$ and $j \in \ell_i$ to the IBE challenger. The challenger returns $\{\mathsf{IBE.sk}_{i,j}\}_{i\in\mathcal{T}(|y|),j\in\ell_i}$.
   - It returns $\mathsf{sk}_y = (\mathcal{T}(|y|), g(y), \{\mathsf{ABE.sk}_i\}_{i\in\mathcal{T}(|y|)}, \{\mathsf{IBE.sk}_{i,j}\}_{i\in\mathcal{T}(|y|),j\in\ell_i})$ to $\mathcal{A}$.
4. **Challenge Query:** $\mathcal{A}$ outputs a pair of equal length messages (μ_0, μ_1). $\mathcal{B}$ does the following.
   - It sets $\mu = \mu_0$ and computes $\{\mathsf{lab}_{i,j,b}\}_{j\in[\ell_i],b\in\{0,1\}}$, for all $i \in \mathcal{S}(|x|)$, as in the honest encryption algorithm.
   - It computes $\{\mathsf{IBE.ct}_{i,j,b}\}_{i\in\mathcal{S}(|x|),j\in[\ell_i],b\in\{0,1\}}$ as follows.
     - If $b = \mathsf{ABE.mpk}_{i,j}$, it honestly encrypts $\mathsf{lab}_{i,j,b}$ to obtain $\mathsf{IBE.ct}_{i,j,b}$.

– If $b = 1 - \mathsf{ABE.mpk}_{i,j}$, $\mathcal{B}$ submits identity (i,j,b) and messages $(\mathsf{lab}_{i,j,b}, \mathsf{lab}_{i,j,1-b})$ to the IBE challenger. The challenger encrypts

$$\mathsf{IBE.ct}_{i,j,b} \leftarrow \begin{cases} \mathsf{IBE.Enc}\left(\mathsf{IBE.mpk}, (i,j,b), \mathsf{lab}_{i,j,b}\right), & \text{if } \hat{\beta} = 0 \\ \mathsf{IBE.Enc}\left(\mathsf{IBE.mpk}, (i,j,b), \mathsf{lab}_{i,j,1-b}\right) & \text{if } \hat{\beta} = 1 \end{cases}$$

and returns $\mathsf{IBE.ct}_{i,j,b}$ to $\mathcal{B}$.
Note that the same label $\mathsf{lab}_{i,j,\mathsf{ABE.mpk}_{i,j}}$ is encrypted for identities $(i,j,0)$ and $(i,j,1)$ if $\beta = 1$.

- It returns $\mathsf{ct}_x = (\mathcal{S}(|x|), f(x), \{\mathsf{IBE.ct}_{i,j,b}\}_{i\in\mathcal{S}(|x|), j\in[\ell_i], b\in\{0,1\}})$ to $\mathcal{A}$.

5. $\mathcal{A}$ outputs a guess bit β'. $\mathcal{B}$ forwards β' to the IBE challenger.

We observe that if the IBE challenger samples $\hat{\beta} = 0$, then $\mathcal{B}$ simulated Hyb_0, else Hyb_1 with $\mathcal{A}$. Hence, advantage of $\mathcal{B} = |\Pr(\beta' = 1|\hat{\beta} = 0) - \Pr(\beta' = 1|\hat{\beta} = 1)| = |\Pr(\beta' = 1|\mathsf{Hyb}_0) - \Pr(\beta' = 1|\mathsf{Hyb}_1)| = \epsilon$ (by assumption).

Admissibility of $\mathcal{B}$. Observe that $\mathcal{B}$ submits identities of the form $(i,j,\mathsf{ABE.mpk}_{i,j})$ for secret key queries and identities of the form $(i,j,1-\mathsf{ABE.mpk}_{i,j})$ for encryption queries. So, $\mathcal{B}$ does not make a secret key query for an identity that is also submitted to the IBE challenger for an encryption query. This establishes the admissibility of $\mathcal{B}$. □

Claim 5.4. Assume that GC is secure. Then $\mathsf{Hyb}_1 \approx_c \mathsf{Hyb}_2$.

Proof. We show that if $\mathcal{A}$ can distinguish between Hyb_1 and Hyb_2 with non-negligible advantage ϵ, then there exists a PPT adversary $\mathcal{B}$ against the security of GC scheme with advantage ϵ. The reduction is as follows.

1. The GC challenger samples a bit $\hat{\beta} \leftarrow \{0,1\}$ and starts the game with $\mathcal{B}$.
2. $\mathcal{B}$ invokes $\mathcal{A}$. $\mathcal{A}$ outputs the challenge ciphertext attribute $x \in \mathrm{A}$.
3. $\mathcal{B}$ generates $(\mathsf{IBE.mpk}, \mathsf{IBE.msk}) \leftarrow \mathsf{IBE.Setup}(1^\lambda)$, sets $\mathsf{mpk} = \mathsf{IBE.mpk}$ and forwards it to $\mathcal{A}$.
4. **Key Queries:** For a key query $y \in \mathrm{B}$, $\mathcal{B}$ computes $\mathsf{sk}_y = (\mathcal{T}(|y|), g(y), \{\mathsf{ABE.sk}_i\}_{i\in\mathcal{T}(|y|)}, \{\mathsf{IBE.sk}_{i,j}\}_{i\in\mathcal{T}(|y|), j\in\ell_i})$ as in Hyb_0.
5. **Challenge Query:** $\mathcal{A}$ outputs a pair of equal length messages (μ_0, μ_1). $\mathcal{B}$ does the following for all $i \in \mathcal{S}(|x|)$.
   - It samples r_i and defines a circuit $\mathsf{E}_i(\cdot) := \mathsf{ABE.Enc}(\cdot, f(x)_i, \mu_0; r_i)$.
   - It submits the circuit $\mathsf{E}_i(\cdot)$ and $\mathsf{ABE.mpk}_i$ to the GC challenger. The challenger does the following.
     – If $\hat{\beta} = 0$, it computes $\{\mathsf{lab}'_{i,j,b}\}_{j\in[\ell_i], b\in\{0,1\}} \leftarrow \mathsf{GC.Garble}(1^\lambda, \mathsf{E}_i)$ and sets $\mathsf{lab}_{i,j,\mathsf{ABE.mpk}_{i,j}} = \mathsf{lab}'_{i,j,\mathsf{ABE.mpk}_{i,j}}$, where $\mathsf{ABE.mpk}_{i,j}$ is the j-th bit of $\mathsf{ABE.mpk}_i$.
     – If $\hat{\beta} = 1$, it computes $\{\mathsf{lab}'_{i,j}\}_{j\in[\ell_i]} \leftarrow \mathsf{GC.Sim}(1^\lambda, 1^{\ell_i}, 1^{|E_i(\cdot)|}, E_i(\mathsf{ABE.mpk}_i))$ and sets $\mathsf{lab}_{i,j,\mathsf{ABE.mpk}_{i,j}} = \mathsf{lab}'_{i,j}$.
     – It returns $\mathsf{lab}_{i,j,\mathsf{ABE.mpk}_{i,j}}$ to $\mathcal{B}$.
   - It computes $\mathsf{IBE.ct}_{i,j,b} \leftarrow \mathsf{IBE.Enc}(\mathsf{IBE.mpk}, (i,j,b), \mathsf{lab}_{i,j,\mathsf{ABE.mpk}_{i,j}})$, for all $i \in \mathcal{S}(|x|), j \in [\ell_i]$, and $b \in \{0,1\}$.
   - It returns $\mathsf{ct}_x = (\mathcal{S}(|x|), f(x), \{\mathsf{IBE.ct}_{i,j,b}\}_{i\in\mathcal{S}(|x|), j\in[\ell_i], b\in\{0,1\}})$ to $\mathcal{A}$.
6. $\mathcal{A}$ outputs a guess bit β'. $\mathcal{B}$ forwards β' to the GC challenger.

We observe that if the GC challenger samples $\hat{\beta} = 0$, then $\mathcal{B}$ simulated the distribution Hyb_1 , else Hyb_2 with $\mathcal{A}$. Hence, advantage of $\mathcal{B} = |\Pr(\beta' = 1|\hat{\beta} = 0) - \Pr(\beta' = 1|\hat{\beta} = 1)| = |\Pr(\beta' = 1|\mathsf{Hyb}_1) - \Pr(\beta' = 1|\mathsf{Hyb}_2)| = \epsilon$ (by assumption). □

Claim 5.5. Assume that ABE is Sel-IND secure. Then $\mathsf{Hyb}_2 \approx_c \mathsf{Hyb}_3$.

Proof. We show that if $\mathcal{A}$ can distinguish between Hyb_2 and Hyb_3 with non-negligible advantage ϵ, then there exists a PPT adversary $\mathcal{B}$ against the security of ABE scheme with advantage ϵ. In this game, the ABE challenger is the challenger corresponding to all the i-th instance of ABE such that $i \in \mathcal{S}(|x|)$. For $i \notin \mathcal{S}(|x|)$, the reduction $\mathcal{B}$ itself generates such i-the instance ABE. The reduction is as follows.

1. The ABE challenger samples a bit $\hat{\beta} \leftarrow \{0,1\}$ and starts the game with $\mathcal{B}$.
2. $\mathcal{B}$ invokes $\mathcal{A}$. $\mathcal{A}$ outputs the challenge ciphertext attribute $x \in \mathrm{A}$.
3. $\mathcal{B}$ generates $(\mathsf{IBE.mpk}, \mathsf{IBE.msk}) \leftarrow \mathsf{IBE.Setup}(1^\lambda)$, sets $\mathsf{mpk} = \mathsf{IBE.mpk}$ and forwards it to $\mathcal{A}$. $\mathcal{B}$ also computes $\mathcal{S}(|x|)$ and $f(x) = \{f(x)_i\}_{i \in \mathcal{S}(|x|)}$.
4. $\mathcal{B}$ sends the challenge ciphertext attributes $f(x)_i$ for the i-th instance of ABE, where $i \in \mathcal{S}(|x|)$, to the ABE challenger. The ABE challenger generates $(\mathsf{ABE.mpk}_i, \mathsf{ABE.msk}_i) \leftarrow \mathsf{ABE.Setup}(1^\lambda, 1^i)$ for $i \in \mathcal{S}(|x|)$ and returns $\{\mathsf{ABE.mpk}_i\}_{i \in \mathcal{S}(|x|)}$ to $\mathcal{B}$.
5. **Key Queries:** For a key query $y \in \mathrm{B}$, $\mathcal{B}$ does the following.
   - It computes the mapping $\mathcal{T}(|y|)$ and for all $i \in \mathcal{T}(|y|) \setminus \mathcal{S}(|x|)$, it generates $(\mathsf{ABE.mpk}_i, \mathsf{ABE.msk}_i) \leftarrow \mathsf{ABE.Setup}(1^\lambda, 1^i)$.
   - It computes the mapping $g(y) = \{g(y)_i\}_{i \in \mathcal{T}(|y|)}$. For all $i \in \mathcal{T}(|y|) \cap \mathcal{S}(|x|)$, it sends a key query $g(y)_i$ for the i-th instance of ABE to its challenger and gets back $\mathsf{ABE.sk}_i$. For $i \in \mathcal{T}(|y|) \setminus \mathcal{S}(|x|)$, $\mathcal{B}$ computes $\mathsf{ABE.sk}_i$ itself.
   - It sets $\ell_i := |\mathsf{ABE.mpk}_i|$ and computes $\mathsf{IBE.sk}_{i,j} \leftarrow \mathsf{IBE.KeyGen}(\mathsf{IBE.msk}, (i, j, \mathsf{ABE.mpk}_{i,j}))$ for all $i \in \mathcal{T}(|y|)$ and $j \in \ell_i$.
   - It returns $\mathsf{sk}_y = (\mathcal{T}(|y|), g(y), \{\mathsf{ABE.sk}_i\}_{i \in \mathcal{T}(|y|)}, \{\mathsf{IBE.sk}_{i,j}\}_{i \in \mathcal{T}(|y|), j \in \ell_i})$. to $\mathcal{A}$.
6. **Challenge Query:** $\mathcal{A}$ outputs a pair of equal length messages (μ_0, μ_1). $\mathcal{B}$ does the following.
   - For all $i \in \mathcal{S}(|x|)$,
     - It sends challenge ciphertext query μ_0, μ_1 for the i-th instance of ABE to the challenger. The challenger computes and returns $\mathsf{ABE.ct}_i \leftarrow \mathsf{ABE.Enc}(\mathsf{ABE.mpk}_i, f(x)_i, \mu_{\hat{\beta}})$.
     - It generates $\{\mathsf{lab}_{i,j}\}_{j \in [\ell_i]} \leftarrow \mathsf{GC.Sim}(1^\lambda, 1^{\ell_i}, 1^{|E_i(\cdot)|}, \mathsf{ABE.ct}_i)$ and computes $\mathsf{IBE.ct}_{i,j,b} \leftarrow \mathsf{IBE.Enc}(\mathsf{IBE.mpk}, (i, j, b), \mathsf{lab}_{i,j})$, for all $j \in [\ell_i]$, and $b \in \{0,1\}$.
   - It returns $\mathsf{ct}_x = (\mathcal{S}(|x|), f(x), \{\mathsf{IBE.ct}_{i,j,b}\}_{i \in \mathcal{S}(|x|), j \in [\ell_i], b \in \{0,1\}})$ to $\mathcal{A}$.
7. $\mathcal{A}$ outputs a guess bit β'. $\mathcal{B}$ forwards β' to the ABE challenger.

We observe that if the ABE challenger samples $\hat{\beta} = 0$, then $\mathcal{B}$ simulated the distribution Hyb_2 , else Hyb_3 with $\mathcal{A}$. Hence, advantage of $\mathcal{B} = |\Pr(\beta' = 1|\hat{\beta} = 0) - \Pr(\beta' = 1|\hat{\beta} = 1)| = |\Pr(\beta' = 1|\mathsf{Hyb}_1) - \Pr(\beta' = 1|\mathsf{Hyb}_2)| = \epsilon$.

Admissibility of $\mathcal{B}$. Observe that $\mathcal{B}$ issues ciphertext queries for the challenge attribute of type $\{f(x)_i\}_{i \in \mathcal{S}(|x|)}$ and key queries of the form $\{g(y)_i\}_{i \in \mathcal{T}(|y|) \cap \mathcal{S}(|x|)}$. Also, by the admissibility of $\mathcal{A}$, it can only issue key queries $y \in \mathrm{B}$ such that $R^{\mathsf{bndl}}(x, y) = 1$ for the challenge ciphertext attribute $x \in \mathrm{A}$. Then by the definition of R^{bndl}, we have $R_i(f(x)_i, g(y)_i) = 1$ for all $i \in \mathcal{S}(|x|) \cap \mathcal{T}(|y|)$. Thus all the key queries made by $\mathcal{B}$ satisfies $R_i(f(x)_i, g(y)_i) = 1$. This establishes the admissibility of $\mathcal{B}$. □

The rest of the hybrids, Hyb_4 and Hyb_5, are simply unwinding the previous hybrids and their proofs of indistinguishability are same as their corresponding counterparts in the first set of hybrids and hence, omitted. □

Circuit $U_{i,x,t}$

Hardwired constants: x, t.
On input $M = (Q, \delta, F) \in \{0,1\}^i$, proceed as follows:

1. Parse the input $M \in \{0,1\}^i$ as a description of a Turing machine.
2. Run M on input x for t steps.
3. Output 1 if the state is in F (accept) and 0 otherwise.

Figure 2: : Circuit $U_{i,x,t}$.

5.2 ABE for Turing Machines

Here, we provide the construction of ABE for Turing machines. Namely, a ciphertext is associated with $(x, 1^t)$ and a secret key is for a Turing machine M, and the decryption results to 1 if the machine accepts the input within t steps and 0 otherwise. To construct such a scheme, we start with constructing two schemes with partial functionality and then combine them. The one scheme takes care of the case where $|(x, 1^t)| > |M|$, while the other takes care of the case where $|(x, 1^t)| \leq |M|$. The idea for the construction is very similar to FE for TM in [AMVY21].

5.2.1 The Case of $|(x, 1^t)| > |M|$

We first show that by applying the conversion in Section 5.1 to the CP-ABE scheme for unbounded depth circuits in Section 4, we can obtain an ABE scheme for Turing machines for the case where $|(x, 1^t)| > |M|$. Formally, we construct an ABE for $R^{>} : \mathrm{A} \times \mathrm{B} \to \{0,1\}$, where $\mathrm{A} = \{0,1\}^*$, B is the set of all Turing machines, and

$$R^{>}((x, 1^t), M) = \begin{cases} 1 & (\text{if } M \text{ accepts } x \text{ in } t \text{ steps}) \wedge \left(|(x, 1^t)| > |M|\right) \\ 0 & \text{otherwise.} \end{cases}.$$

To apply the conversion, we use various instances of the unbounded depth and size cpABE for $\mathsf{prm} = 1^i$, $R_i : \mathcal{X}_i \times \mathcal{Y}_i \to \{0,1\} \cup \{\bot\}$ where $\mathcal{X}_i$ is the set of circuits with input length i, and output length 1, $\mathcal{Y}_i = \{0,1\}^i$, and $R_i(C, x) = C(x)$. We then set $\mathcal{S}$, $\mathcal{T}$, f, and g as

$$\mathcal{S}(i) = \{1, 2, \ldots, i-1\}, \quad \mathcal{T}(i) = \{i\}, \quad f(x, 1^t) = \{U_{i,x,t}(\cdot)\}_{i \in [|(x,1^t)|-1]}, \quad g(M) = M$$

where $U_{i,x,t}(\cdot)$ is defined as Figure 2. The circuit (in particular, Step 2 of the computation) is padded so that the circuit size only depends on $|(x, 1^t)|$. Note that $U_{i,x,t}$ is in $\mathcal{X}_i$ even for x and t with unbounded length, since $\mathcal{X}_i$ contains circuits of unbounded size. Then, by inspection, we can observe that for $\mathcal{X}_i, \mathcal{Y}_i, \mathcal{S}, \mathcal{T}, f, g, \mathrm{A}, \mathrm{B}$ defined as above, R^{bndl} defined as Equation (2) is equivalent to $R^{>}$ except for the case of $|(x, 1^t)| \leq |M|$. In this case, we have $\mathcal{S}(|x|) \cap \mathcal{T}(|y|) = \emptyset$. To handle this, we add the extra step to the decryption algorithm of the former where it outputs $\bot$ if $\mathcal{S}(|x|) \cap \mathcal{T}(|y|) = \emptyset$. Since the original scheme is $\mathsf{VerSel\text{-}IND}$ secure, so is the resulting scheme by Theorem 5.2.

5.2.2 The Case of $|(x, 1^t)| \leq |M|$

We next show that by applying the conversion in Section 5.1 to the KP-ABE scheme from [BGG+14], we can obtain an ABE scheme for TM for the case where $|(x, 1^t)| \leq |M|$. Formally, we construct an ABE for $R^{\leq} : \mathrm{A} \times \mathrm{B} \to \{0,1\} \cup \{\bot\}$, where $\mathrm{A} = \{0,1\}^*$, B is the set of all non-deterministic Turing machines, and

$$R^{\leq}((x, 1^t), M) = \begin{cases} 1 & (\text{if } M \text{ accepts } x \text{ within } t \text{ steps}) \wedge \left(|(x, 1^t)| \leq |M|\right) \\ 0 & \text{otherwise.} \end{cases}.$$

Circuit $U_{i,M}$

Hardwired constants: Description of a Turing Machine M.
On input $y \in \{0,1\}^i$, proceed as follows:

1. Parse the input $y \in \{0,1\}^i$ as $(x, 1^t)$.
2. Otherwise, run M on input x for t steps.
3. Output 1 if the state is in F (accept) and 0 otherwise.

Figure 3: : Circuit $U_{i,M}$.

To apply the conversion, we use various instances of kpABE from [BGG+14] for $\mathsf{prm} = 1^i$, $R_i : \mathcal{X}_i \times \mathcal{Y}_i \to \{0,1\}$ where $\mathcal{X}_i = \{0,1\}^i$ and $\mathcal{Y}_i$ is the set of circuits with input length i, depth $i \cdot \lambda$, and output length 1, and $R_i(x, C) = C(x)$. We then set $\mathcal{S}$, $\mathcal{T}$, f, and g as

$$\mathcal{S}(i) = i, \quad \mathcal{T}(i) = \{1, 2, \ldots, i\}, \quad f(x, 1^t) = (x, 1^t), \quad g(M) = \{U_{i,M}(\cdot)\}_{i \in [M]},$$

where $U_{i,M}(\cdot)$ is defined as Figure 3. Here, we check that $U_{i,M}(\cdot)$ is in $\mathcal{Y}_i$. Recall that even though $\mathcal{Y}_i$ supports circuits with unbounded size, it has a bound on the depth of the circuit. We therefore argue that the depth of $U_{i,M}(\cdot)$ does not exceed $i\lambda$, even for unbounded size $|M|$. We evaluate the depth of Step 2 of the circuit, since this is the only non-trivial step. By Lemma 2.11, this step can be implemented by a circuit with depth

$$t \cdot \mathrm{poly}(\log |x|, \log t, \log |M|) \leq i \cdot \mathrm{poly}(\log \lambda) \leq i \cdot \lambda$$

Then, by inspection, we can observe that for $\mathcal{X}_i, \mathcal{Y}_i, \mathcal{S}, \mathcal{T}, f, g, \mathrm{A}, \mathrm{B}$ defined as above, R^{bndl} defined as Equation (2) is equivalent to $R^{\leq}$ except for the case of $|(x, 1^t)| > |M|$. In this case, we have $\mathcal{S}(|x|) \cap \mathcal{T}(|y|) = \emptyset$. To handle this, we add the extra step to the decryption algorithm of the former where it outputs $\perp$ if $\mathcal{S}(|x|) \cap \mathcal{T}(|y|) = \emptyset$. Since the original scheme is Sel-IND secure, so is the resulting scheme by Theorem 5.2.

5.2.3 Putting the Pieces Together

Here, we combine the two schemes we considered so far to obtain the full-fledged scheme. We set $\mathrm{A} = \{0,1\}^*$ and B to be the set of all Turing machines. We also set $R_1 = R^{>}$, $R_2 = R^{\leq}$. $\mathcal{X}_i = \mathrm{A}$, $\mathcal{Y}_i = \mathrm{B}$ for $i = 1, 2$. We have already constructed schemes for R_1 and R_2 and now combine them. To do so, we set $\mathcal{S}$, $\mathcal{T}$, f, and g as

$$\mathcal{S}(i) = \{1, 2\}, \quad \mathcal{T}(i) = \{1, 2\}, \quad f(x, 1^t) = \{(x, 1^t), (x, 1^t)\}, \quad g(M) = \{M, M\}$$

We observe that for $\mathcal{X}_i, \mathcal{Y}_i, \mathcal{S}, \mathcal{T}, f, g, \mathrm{A}, \mathrm{B}$ defined as above, R^{bndl} defined as Equation (2) is

$$R^{\mathsf{bndl}}((x, 1^t), M) = \begin{cases} (1,0) & (\text{if } M \text{ accepts } x \text{ within } t \text{ steps}) \wedge \left(|(x, 1^t)| > |M|\right) \\ (0,1) & (\text{if } M \text{ accepts } x \text{ within } t \text{ steps}) \wedge \left(|(x, 1^t)| \leq |M|\right) \\ (0,0) & \text{otherwise.} \end{cases}.$$

To obtain the ABE scheme for TM, we add an extra step for the decryption algorithm of the ABE scheme for R^{bndl} obtained above, where we output the message recovered if the decryption succeeds for either left or right slot and $\perp$ otherwise. To sum up, we have the following theorem

Theorem 5.6. Assume a VerSel-IND secure cpABE scheme that supports circuits with unbounded depth and a Sel-IND secure kpABE scheme that supports bounded depth circuits. Then there exists a ABE for Turing machines, presented in Section 5.2.3, satisfying VerSel-IND security.

Instantiating the VerSel-IND secure cpABE scheme from Section 4 and Sel-IND secure kpABE scheme from [BGG+14], we get the following corollary.

Corollary 5.7. Under the LWE assumption, evasive LWE assumption (Assumption 2.24) and circular tensor LWE assumption (Assumption 4.1), there exists a very selectively secure ABE for TM with

$$|\mathsf{mpk}| = \mathrm{poly}(\lambda), \quad |\mathsf{sk}| = \mathrm{poly}(|M|, \lambda), \quad |\mathsf{ct}| = \mathrm{poly}(\lambda, |x|, t)$$

where the Turing machine M runs on input x for time step t.

5.3 ABE for NL

Here, we provide the construction of ABE for NL. Namely, a ciphertext encrypts a string $(x, 1^t, 1^{2^s})$, where x is a string, t is the time bound, s is the space bound for the computation and a secret key is associated with a non-deterministic Turing machine M. The decryption is possible if M accepts x within t steps and the space used for the computation does not exceed s. The idea for the construction is very similar to FE for NL in [AMVY21]. We consider two schemes that complement each other and then combine them.

Transition Matrix. Before describing the schemes, we need some preparations. Similarly to [LL20b], we represent the computation of $M(x)$ as a multiplication of matrices. To do so, let us enumerate all the possible internal configurations that may appear when we run $M = (Q, \delta, F)$ on input x with the space for the computation being bounded by s. As an internal configuration, we have $|x|$ and s choices for the input and work tape pointers, respectively. We also have $|Q|$ choices for the possible state, and 2^s possible choices for the contents of the work tape. Therefore, we have

$$N := s2^s \cdot |x| \cdot |Q|$$

possible internal configurations. We associate each $i \in [N]$ with such configuration and represent the configuration by a vector $\mathbf{e}_i$, which is a unit vector whose entries are all 0 except for the i-th entry that is set to be 1. We also define the matrix $\mathsf{Mat}(M, x, s)$ as

$$\mathsf{Mat}(M, x, s)_{i,j} := \begin{cases} 1 & \text{if the configuration } i \text{ can reach } j \text{ in one step by } \delta \\ 0 & \text{otherwise} \end{cases}$$

where $\mathsf{Mat}(M, x, s)_{i,j}$ is the (i, j)-th entry of the matrix.
Furthermore, we consider a special matrix multiplication over $\{0,1\}$, where the multiplication $\mathbf{A} \cdot \mathbf{B} \in \{0,1\}^{N_1 \times N_3}$ of two matrices $\mathbf{A} \in \{0,1\}^{N_1 \times N_2}$ and $\mathbf{B} \in \{0,1\}^{N_2 \times N_3}$ is defined as

$$(\mathbf{A} \cdot \mathbf{B})_{i,j} = \bigvee_{k \in [N_2]} \left(\mathbf{B}_{i,k} \wedge \mathbf{C}_{k,j} \right)$$

where we denote the (i, j)-th entry of a matrix $\mathbf{D}$ by $\mathbf{D}_{i,j}$ above. The multiplication can be defined for any size of matrices and in particular, also defined for computations involving vectors. We then define $\mathbf{e}_{\mathsf{stt}}$ as the vector corresponding to the initial state of the computation and $\mathbf{u}_{\mathsf{acc}}$ as

$$\mathbf{u}_{\mathsf{acc}} = \sum_{i \in [N] \,:\, i \text{ encodes accepting state}} \mathbf{e}_i.$$

Then, we observe that

$$\mathbf{e}_{\mathsf{stt}}^\top \cdot \left(\mathsf{Mat}(M, x, s)_{i,j} \right)^t \cdot \mathbf{u}_{\mathsf{acc}} = \begin{cases} 1 & \text{if } M \text{ accepts } x \text{ within } t \text{ steps and space } s \\ 0 & \text{otherwise} \end{cases}$$

holds from the property of the transition matrix, where we use the special multiplication above.

Circuit $U_{i,x,t,s}$

Hardwired constants: x, t, s.
On input $M \in \{0,1\}^i$, proceed as follows:

1. Parse the input $M = (Q, \delta, F)$ as a description of a Turing machine.
2. Compute $\mathsf{Mat}(M, x, s)_{j,k}$ for all $j, k \in [N]$ in parallel, where $N = s2^s \cdot |x| \cdot |Q|$.
3. Compute $\mathbf{A} := \mathsf{Mat}(M, x, s)^t$.
4. Compute and output $\mathbf{e}_{\mathsf{stt}}^\top \cdot \mathbf{A} \cdot \mathbf{u}_{\mathsf{acc}}$.

Figure 4: Circuit $U_{i,x,t,s}$.

5.3.1 The Case of $|(x, 1^t, 1^{2^s})| > |M|$

We first show that by applying the conversion in Section 5.1 to the CP-ABE scheme for circuits in [Wee22], we can obtain an ABE scheme for NL for the case where $|(x, 1^t, 1^{2^s})| > |M|$. Formally, we construct an ABE for $R^> : \mathrm{A} \times \mathrm{B} \to \{0,1\}$, where $\mathrm{A} = \{0,1\}^*$, B is the set of all non-deterministic Turing machines, and

$$R^>((x, 1^t, 1^{2^s}), M) = \begin{cases} 1 & (\text{if } M \text{ accepts } x \text{ within } t \text{ steps and space } s) \wedge \left(|(x, 1^t, 1^{2^s})| > |M|\right) \\ 0 & \text{otherwise.} \end{cases}.$$

To apply the conversion, we use various instances of cpABE from [Wee22] for $\mathsf{prm} = 1^i$, $R_i : \mathcal{X}_i \times \mathcal{Y}_i \to \{0,1\} \cup \{\bot\}$ where $\mathcal{X}_i$ is the set of circuits with input length i, depth λ, and output length 1, $\mathcal{Y}_i = \{0,1\}^i$, and $R_i(C, x) = C(x)$. We then set $\mathcal{S}$, $\mathcal{T}$, f, and g as

$$\mathcal{S}(i) = \{1, 2, \ldots, i-1\}, \quad \mathcal{T}(i) = \{i\}, \quad f(x, 1^t, 1^{2^s}) = \{U_{i,x,t,s}(\cdot)\}_{i \in [|(x,1^t,1^{2^s})|-1]}, \quad g(M) = M$$

where $U_{i,x,t,s}(\cdot)$ is defined as Figure 4. The circuit may be padded so that it does not leak more information about the hardwired constants (x, t, s) beyond $|(x, 1^t, 1^{2^s})|$.

We now show that f is a valid map. Namely, $U_{i,x,t,s}$ is in $\mathcal{X}_i$ even for x and t with unbounded length. Recall that $\mathcal{X}_i$ has a bound on the depth of the circuits it supports, even though it does not have such a bound on the size. We argue that the depth of $U_{i,x,t,s}$ is bounded by λ. To do so, we evaluate the depth of each computation step of $U_{i,x,t,s}$. First, Step 1 of the computation is trivial. Step 2 can be implemented by a circuit of depth $\mathrm{poly}(\log |x|, \log s, \log |M|)$ by Lemma 2.12. Step 3 can be executed by $O(\log t)$ multiplication of matrices of size $N \times N$, which can be implemented with depth $O(\log N)$. Therefore, this step can be computed with depth $O(\log N \log t)$. Step 4 can also be implemented with depth $O(\log N)$. Therefore, the entire computation can be implemented with depth

$$\mathrm{poly}(\log s, \log |x|, \log t, \log N, \log |M|) \leq \mathrm{poly}(\log \lambda) \leq \lambda$$

as desired. Then, by inspection, we can observe that for $\mathcal{X}_i, \mathcal{Y}_i, \mathcal{S}, \mathcal{T}, f, g, \mathrm{A}, \mathrm{B}$ defined as above, R^{bndl} defined as Equation (2) is equivalent to $R^>$ except for the case of $|(x, 1^t, 1^{2^s})| \leq |M|$. In this case, we have $\mathcal{S}(|x|) \cap \mathcal{T}(|y|) = \emptyset$. To handle this, we add the extra step to the decryption algorithm of the former where it outputs $\bot$ if $\mathcal{S}(|x|) \cap \mathcal{T}(|y|) = \emptyset$. This gives us the construction of ABE for $R^>$. Since the original scheme is $\mathsf{VerSel\text{-}IND}$ secure, so is the resulting scheme by Theorem 5.2.

5.3.2 The Case of $|(x, 1^t, 1^{2^s})| \leq |M|$

We next show that by applying the conversion in Section 5.1 to the KP-ABE scheme from [BGG+14], we can obtain an ABE scheme for NL for the case where $|(x, 1^t, 1^{2^s})| \leq |M|$. Formally, we construct an ABE for

Circuit $U_{i,M}$

Hardwired constants: Description of a Turing Machine M.
On input $y \in \{0,1\}^i$, proceed as follows:

1. Parse the input $y \in \{0,1\}^i$ as $(x, 1^t, 1^{2^s})$.
2. Compute $\mathsf{Mat}(M,x,s)_{j,k}$ for all $j,k \in [N]$ in parallel, where $N = s2^s \cdot |x| \cdot |Q|$.
3. Compute $\mathbf{A} := \mathsf{Mat}(M,x,s)^t$.
4. Compute and output $\mathbf{e}_{\mathsf{stt}}^\top \cdot \mathbf{A} \cdot \mathbf{u}_{\mathsf{acc}}$.

Figure 5: Circuit $U_{i,M}$.

$R^{\leq} : \mathrm{A} \times \mathrm{B} \to \{0,1\} \cup \{\bot\}$, where $\mathrm{A} = \{0,1\}^*$, B is the set of all non-deterministic Turing machines, and

$$R^{\leq}((x,1^t,1^{2^s}),M) = \begin{cases} 1 & (\text{if } M \text{ accepts } x \text{ within } t \text{ steps and space } s) \wedge \left(|(x,1^t,1^{2^s})| \leq |M|\right) \\ 0 & \text{otherwise.} \end{cases}.$$

To apply the conversion, we use various instances of kpABE from [BGG+14] for $\mathsf{prm} = 1^i$, $R_i : \mathcal{X}_i \times \mathcal{Y}_i \to \{0,1\}$ where $\mathcal{X}_i = \{0,1\}^i$ and $\mathcal{Y}_i$ is the set of circuits with input length i, depth λ, and output length 1, and $R_i(x,C) = C(x)$. We then set $\mathcal{S}$, $\mathcal{T}$, f, and g as

$$\mathcal{S}(i) = i, \quad \mathcal{T}(i) = \{1,2,\ldots,i\}, \quad f(x,1^t,1^{2^s}) = (x,1^t,1^{2^s}), \quad g(M) = \{U_{i,M}(\cdot)\}_{i\in[M]},$$

where $U_{i,M}(\cdot)$ is defined as Figure 5.

We now show that g is a valid map. Namely, $U_{i,M}$ is in $\mathcal{Y}_i$ even for M with unbounded size. In particular, we have to show that the depth of the circuit is bounded by λ. This can be shown by the same argument as for $U_{i,x,t,s}$, since both circuits compute $\mathbf{e}_{\mathsf{stt}}^\top \cdot \mathsf{Mat}(M,x,s)^t \cdot \mathbf{u}_{\mathsf{acc}}$ from (x,t,s,M) in exactly the same way. Then, by inspection, we can observe that for $\mathcal{X}_i, \mathcal{Y}_i, \mathcal{S}, \mathcal{T}, f, g, \mathrm{A}, \mathrm{B}$ defined as above, R^{bndl} defined as Equation (2) is equivalent to $R^{\leq}$ except for the case of $|(x,1^t,1^{2^s})| > |M|$. However, this case can be handled by modifying the decryption algorithm as in Section 5.3.1. This gives us the construction of ABE for $R^{\leq}$. Since the original scheme is $\mathsf{Sel\text{-}IND}$ secure, so is the resulting scheme by Theorem 5.2.

5.3.3 Putting the Pieces Together

Here, we combine the two schemes we considered so far to obtain the full-fledged scheme. We set $\mathrm{A} = \{0,1\}^*$ and B to be the set of all Turing machines. We also set $R_1 = R^{>}$, $R_2 = R^{\leq}$. $\mathcal{X}_i = \mathrm{A}$, $\mathcal{Y}_i = \mathrm{B}$ for $i = 1,2$. We have already constructed schemes for R_1 and R_2 and now combine them. To do so, we set $\mathcal{S}$, $\mathcal{T}$, f, and g as

$$\mathcal{S}(i) = \{1,2\}, \quad \mathcal{T}(i) = \{1,2\}, \quad f(x,1^t) = \{(x,1^t,1^{2^s}),(x,1^t,1^{2^s})\}, \quad g(M) = \{M,M\}$$

We observe that for $\mathcal{X}_i, \mathcal{Y}_i, \mathcal{S}, \mathcal{T}, f, g, \mathrm{A}, \mathrm{B}$ defined as above, R^{bndl} defined as Equation (2) is

$$R^{\mathsf{bndl}}((x,1^t,1^{2^s}),M) = \begin{cases} (1,0) & (\text{if } M \text{ accepts } x \text{ within } t \text{ steps and space } s) \wedge \left(|(x,1^t,1^{2^s})| > |M|\right) \\ (0,1) & (\text{if } M \text{ accepts } x \text{ within } t \text{ steps and space } s) \wedge \left(|(x,1^t,1^{2^s})| \leq |M|\right) \\ (0,0) & \text{otherwise.} \end{cases}.$$

To obtain the ABE scheme for NL, we add an extra step for the decryption algorithm of the ABE scheme for R^{bndl} obtained above, where we output the message recovered if the decryption succeeds for either left or right slot and $\bot$ otherwise. To sum up, we have the following theorem:

Theorem 5.8. Assume a VerSel-IND secure cpABE and Sel-IND secure kpABE scheme that supports circuits with bounded depth. Then the ABE scheme for NL presented in Section 5.3.3 satisfies VerSel-IND security.

Instantiating the VerSel-IND secure cpABE scheme from [Wee22] and Sel-IND secure kpABE scheme from [BGG+14], we get the following corollary.

Corollary 5.9. Under the evasive LWE assumption (Assumption 2.24) and tensor LWE assumption (Assumption 4.1), there exists a very selectively secure ABE for NL with

$$|\mathsf{mpk}| = \mathrm{poly}(\lambda), \quad |\mathsf{sk}| = \mathrm{poly}(|M|, \lambda), \quad |\mathsf{ct}| = \mathrm{poly}(\lambda, |x|, t, 2^s)$$

where the machine M runs on input x for time step t and takes space s.

Acknowledgement. The first author is supported by the Cybersecurity Center of Excellence at IIT Madras and the DST Swarnajayanti fellowship. The third author was partly supported by JST AIP Acceleration Research JPMJCR22U5, JST CREST Grant Number JPMJCR22M1, and JSPS KAKENHI Grant Number 19H01109.

References

[ABB10a] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 553–572. Springer, Heidelberg, May / June 2010. (Cited on page 14, 33.)

[ABB10b] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 98–115. Springer, Heidelberg, August 2010. (Cited on page 14.)

[AM18] Shweta Agrawal and Monosij Maitra. FE and iO for Turing machines from minimal assumptions. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 473–512. Springer, Heidelberg, November 2018. (Cited on page 3.)

[AMVY21] Shweta Agrawal, Monosij Maitra, Narasimha Sai Vempati, and Shota Yamada. Functional encryption for Turing machines with dynamic bounded collusion from LWE. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part IV*, volume 12828 of *LNCS*, pages 239–269, Virtual Event, August 2021. Springer, Heidelberg. (Cited on page 4, 33, 39, 41.)

[AMY19a] Shweta Agrawal, Monosij Maitra, and Shota Yamada. Attribute based encryption (and more) for nondeterministic finite automata from LWE. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 765–797. Springer, Heidelberg, August 2019. (Cited on page 3, 4.)

[AMY19b] Shweta Agrawal, Monosij Maitra, and Shota Yamada. Attribute based encryption for deterministic finite automata from DLIN. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part II*, volume 11892 of *LNCS*, pages 91–117. Springer, Heidelberg, December 2019. (Cited on page 3.)

[ARYY23] Shweta Agrawal, Mélissa Rossi, Anshu Yadav, and Shota Yamada. Constant input attribute based (and predicate) encryption from evasive and tensor LWE. In *CRYPTO 2023, Part IV*, LNCS, pages 532–564. Springer, Heidelberg, August 2023. (Cited on page 4, 15, 16.)

[AS16] Prabhanjan Vijendra Ananth and Amit Sahai. Functional encryption for Turing machines. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part I*, volume 9562 of *LNCS*, pages 125–153. Springer, Heidelberg, January 2016. (Cited on page 3.)

[AY20a] Shweta Agrawal and Shota Yamada. CP-ABE for circuits (and more) in the symmetric key setting. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part I*, volume 12550 of *LNCS*, pages 117–148. Springer, Heidelberg, November 2020. (Cited on page 3.)

[AY20b] Shweta Agrawal and Shota Yamada. Optimal broadcast encryption from pairings and LWE. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 13–43. Springer, Heidelberg, May 2020. (Cited on page 5.)

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, Heidelberg, August 2001. (Cited on page 9.)

[BGG+14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 533–556. Springer, Heidelberg, May 2014. (Cited on page 3, 4, 5, 7, 13, 18, 39, 40, 41, 42, 43, 44.)

[BGV14] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):1–36, 2014. (Cited on page 6, 7.)

[BHR12a] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Adaptively secure garbling with applications to one-time programs and secure outsourcing. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 134–153. Springer, Heidelberg, December 2012. (Cited on page 9.)

[BHR12b] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *ACM CCS 2012*, pages 784–796. ACM Press, October 2012. (Cited on page 8.)

[BLP+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 575–584. ACM Press, June 2013. (Cited on page 14, 15.)

[BTVW17] Zvika Brakerski, Rotem Tsabary, Vinod Vaikuntanathan, and Hoeteck Wee. Private constrained PRFs (and more) from LWE. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 264–302. Springer, Heidelberg, November 2017. (Cited on page 6, 18.)

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *52nd FOCS*, pages 97–106. IEEE Computer Society Press, October 2011. (Cited on page 6, 7.)

[BV22] Zvika Brakerski and Vinod Vaikuntanathan. Lattice-inspired broadcast encryption and succinct ciphertext-policy abe. In *13th Innovations in Theoretical Computer Science Conference (ITCS 2022)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022. (Cited on page 5.)

[CHKP10] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 523–552. Springer, Heidelberg, May / June 2010. (Cited on page 14, 33.)

[CM15] Michael Clear and Ciaran McGoldrick. Multi-identity and multi-key leveled FHE from learning with errors. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 630–656. Springer, Heidelberg, August 2015. (Cited on page 7.)

[Coc01] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *IMA Int. Conf.*, 2001. (Cited on page 9.)

[DG17] Nico Döttling and Sanjam Garg. Identity-based encryption from the Diffie-Hellman assumption. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 537–569. Springer, Heidelberg, August 2017. (Cited on page 33.)

[GKP+13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. How to run Turing machines on encrypted data. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 536–553. Springer, Heidelberg, August 2013. (Cited on page 3.)

[GKW16] Rishab Goyal, Venkata Koppula, and Brent Waters. Semi-adaptive security and bundling functionalities made generic and easy. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 361–388. Springer, Heidelberg, October / November 2016. (Cited on page 8.)

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 89–98. ACM Press, October / November 2006. Available as Cryptology ePrint Archive Report 2006/309. (Cited on page 3.)

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008. (Cited on page 14.)

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 75–92. Springer, Heidelberg, August 2013. (Cited on page 6, 17.)

[GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 162–179. Springer, Heidelberg, August 2012. (Cited on page 9.)

[GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In *STOC*, 2013. (Cited on page 3, 4.)

[GW20] Junqing Gong and Hoeteck Wee. Adaptively secure abe for dfa from k-lin and more. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 278–308. Springer, 2020. (Cited on page 3.)

[GWW19] Junqing Gong, Brent Waters, and Hoeteck Wee. Abe for dfa from k-lin. In *Advances in Cryptology–CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part II 39*, pages 732–764. Springer, 2019. (Cited on page 3.)

[HLL23] Yao-Ching Hsieh, Huijia Lin, and Ji Luo. Attribute-based encryption for circuits of unbounded depth from lattices. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 415–434. IEEE, 2023. (Cited on page 3, 4, 6, 15, 17, 18, 19, 20, 21, 27.)

[KNTY19] Fuyuki Kitagawa, Ryo Nishimaki, Keisuke Tanaka, and Takashi Yamakawa. Adaptively secure and succinct functional encryption: Improving security and efficiency, simultaneously. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 521–551. Springer, Heidelberg, August 2019. (Cited on page 3.)

[KT18] Fuyuki Kitagawa and Keisuke Tanaka. Key dependent message security and receiver selective opening security for identity-based encryption. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 32–61. Springer, Heidelberg, March 2018. (Cited on page 9.)

[LL20a] Huijia Lin and Ji Luo. Compact adaptively secure ABE from k-Lin: Beyond NC^1 and towards NL. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, volume 12107 of *LNCS*, pages 247–277. Springer, Heidelberg, May 2020. (Cited on page 3, 10.)

[LL20b] Huijia Lin and Ji Luo. Succinct and adaptively secure ABE for ABP from k-Lin. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part III*, volume 12493 of *LNCS*, pages 437–466. Springer, Heidelberg, December 2020. (Cited on page 41.)

[LP09] Yehuda Lindell and Benny Pinkas. A proof of security of Yao's protocol for two-party computation. *Journal of Cryptology*, 22(2):161–188, April 2009. (Cited on page 9.)

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 700–718. Springer, Heidelberg, April 2012. (Cited on page 14.)

[MW16] Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key FHE. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 735–763. Springer, Heidelberg, May 2016. (Cited on page 7.)

[PF79] Nicholas Pippenger and Michael J Fischer. Relations among complexity measures. *Journal of the ACM (JACM)*, 26(2), 1979. (Cited on page 11.)

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM*, 56(6):34:1–34:40, 2009. (Cited on page 15.)

[SS10] Amit Sahai and Hakan Seyalioglu. Worry-free encryption: functional encryption with public keys. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM CCS 2010*, pages 463–472. ACM Press, October 2010. (Cited on page 5.)

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, 2005. (Cited on page 3.)

[Tsa22] Rotem Tsabary. Candidate witness encryption from lattice techniques. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part I*, volume 13507 of *LNCS*, pages 535–559. Springer, Heidelberg, August 2022. (Cited on page 3.)

[VWW22] Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. Witness encryption and null-IO from evasive LWE. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part I*, volume 13791 of *LNCS*, pages 195–221. Springer, Heidelberg, December 2022. (Cited on page 4.)

[Wat12] Brent Waters. Functional encryption for regular languages. In *Annual Cryptology Conference*, pages 218–235. Springer, 2012. (Cited on page 3.)

[Wee22] Hoeteck Wee. Optimal broadcast encryption and CP-ABE from evasive lattice assumptions. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 217–241. Springer, Heidelberg, May / June 2022. (Cited on page 3, 4, 5, 6, 7, 13, 14, 15, 17, 27, 42, 44.)

[WWW22] Brent Waters, Hoeteck Wee, and David J. Wu. Multi-authority ABE from lattices without random oracles. In Eike Kiltz and Vinod Vaikuntanathan, editors, *TCC 2022, Part I*, volume 13747 of *LNCS*, pages 651–679. Springer, Heidelberg, November 2022. (Cited on page 15.)

[Yao82] Andrew Chi-Chih Yao. Protocols for secure computations extended abstract. In *23rd FOCS*, volume 28, 1982. (Cited on page 8, 33.)