Deniable Attribute Based Encryption for Branching Programs from LWE

Daniel Apon * Xiong Fan † Feng-Hao Liu ‡

Abstract

Deniable encryption (Canetti et al. CRYPTO '97) is an intriguing primitive that provides a security guarantee against not only eavesdropping attacks as required by semantic security, but also stronger coercion attacks performed after the fact. The concept of deniability has later demonstrated useful and powerful in many other contexts, such as leakage resilience, adaptive security of protocols, and security against selective opening attacks. Despite its conceptual usefulness, our understanding of how to construct deniable primitives under standard assumptions is restricted.

In particular from standard lattice assumptions, i.e. Learning with Errors (LWE), we have only flexibly and non-negligible advantage deniable public-key encryption schemes, whereas with the much stronger assumption of indistinguishable obfuscation, we can obtain at least fully sender-deniable PKE and computation. How to achieve deniability for other more advanced encryption schemes under standard assumptions remains an interesting open question.

In this work, we construct a flexibly bi-deniable Attribute-Based Encryption (ABE) scheme for all polynomial-size Branching Programs from LWE. Our techniques involve new ways of manipulating Gaussian noise that may be of independent interest, and lead to a significantly sharper analysis of noise growth in Dual Regev type encryption schemes. We hope these ideas give insight into achieving deniability and related properties for further, advanced cryptographic systems from lattice assumptions.

1 Introduction

Deniable encryption, introduced by Canetti et al. [CDNO97] at CRYPTO 1997, is an intriguing primitive that allows Alice to privately communicate with Bob in a way that resists not only eavesdropping attacks as required by semantic security, but also stronger *coercion attacks* performed after the fact. An eavesdropper Eve stages a cocercion attack by additionally approaching Alice (or Bob, or both) *after* a ciphertext is transmitted and demanding to see all secret information: the plaintext, the random coins used by Alice for encryption, and any private keys held by Bob (or Alice) related to the ciphertext. In particular, Eve can use this information to "fully unroll" the *exact transcript* of some deterministic decryption procedure purportedly computed by Bob, as well as verify that the exact coins and decrypted plaintext in fact produce the coerced ciphertext. A secure deniable encryption scheme should maintain privacy of the sensitive data originally communicated between Alice and Bob under the coerced ciphertext (instead substituting a benign yet *convincing* plaintext in the view of Eve), even in the face of such a revealing attack and even if Alice and Bob may not interact during the coercion phase.

Historically, deniable encryption schemes have been challenging to construct. Under standard assumptions, Canetti et al. [CDNO97] constructed a sender-deniable[1] PKE where the distinguishing advantage

*University of Maryland, dapon@cs.umd.edu.

†Cornell University, xfan@cs.cornell.edu.

‡Florida Atlantic University, fenghao.liu@fau.edu.

[1]We differentiate between sender-, receiver-, and bi-deniable schemes. A bi-deniable scheme is both sender- and receiver-deniable.

between real and fake openings is an inverse polynomial depending on the public key size. But it was not until 2011 that O'Neill, Peikert, and Waters [OPW11] proposed the first constructions of bi-deniable PKE with *negligible* deniability distinguishing advantage: from simulatable PKE generically, as well as from Learning with Errors (LWE [Reg05]) directly.

Concurrently, Bendlin et al. [BNNO11] showed an inherent limitation: any non-interactive public-key encryption scheme may be receiver-deniable (resp. bi-deniable) only with *non-negligible* $\Omega(1/\mathsf{size}(\mathsf{pk}))$ distinguishing advantage in the deniability experiment. Indeed, O'Neill et al. [OPW11] bypass the impossibility result of [BNNO11] by working in the so-called *flexible*[2] model of deniability. In the flexible of deniability, private keys sk are distributed by a central key authority. In the event that Bob is coerced to reveal a key sk that decrypts chosen ciphertext ct^*, the key authority distributes a *faking key* fk to Bob, which Bob can use to generate a fake key sk^* (designed to behave identically to sk except on ciphertext ct^*). If this step is allowed, then O'Neill et al. demonstrate that for their constructions, Eve has at most negligible advantage in distinguishing whether Bob revealed an honest sk or fake sk^*.

A major breakthrough in deniable encryption arrived with the work of Sahai and Waters [SW14], who proposed the first sender-deniable PKE with negligible distinguishing advantage from indistinguishability obfuscation ($i\mathcal{O}$) for $\mathsf{P/poly}$ [GGH$^+$13]. The concept of deniability has been demonstrated useful in the contexts of leakage resilience [DLZ], adaptive security for protocols, and as well as deniable computation (or algorithms) [CGP, DKR, GP]. In addition to coercion resistance, a bi-deniable encryption scheme is a non-committing encryption scheme [CFGN96], as well as a scheme secure under selective opening (SOA) attacks [BHY09], which are of independent theoretical interest.

Very recently, De Caro, Iovino, and O'Neill [CIO16] gave various constructions of deniable *functional* encryption. First, they show a generic transformation of any IND-secure FE scheme for circuits into a flexibly receiver-deniable FE for circuits. Second, they give a direct construction of receiver-deniable FE for Boolean formulae from bilinear maps. Further, in the stronger *multi-distributional* model of deniable functional encryption – where there are special "deniable" set-up and encryption algorithms in addition to the plain ones, and where under coercion, it may non-interactively be made to seem as only the normal algorithms were used – De Caro et al. [CIO16] construct receiver-deniable FE for circuits under the additional (powerful) assumption of different-inputs obfuscation ($di\mathcal{O}$).

De Caro et al. [CIO16] also show (loosely speaking) that any receiver-deniable FE implies SIM-secure FE for the same functionality. Following [CIO16], we also emphasize that deniability for functional encryption is a **strictly stronger** property than SIM security, since fixed coerced ciphertexts must decrypt correctly and benignly *in the real world*. Finally, we mention that in concurrent work, Apon, Fan, and Liu, in an unpublished work [AFL15], construct flexibly bi-deniable inner product encryption from standard *lattice* assumptions. This work generalizes and thus subsumes the prior results of [AFL15].

Despite the apparent theoretical utility in understanding the extent to which cryptographic constructions are deniable, our current knowledge of constructing such schemes from standard lattice assumptions is still limited. From LWE, we have only flexible and non-negligible advantage deniable encryption schemes (or IPE from [AFL15]), whereas with the much more powerful assumption of indistinguishability obfuscation ($i\mathcal{O}$), we can obtain at least fully-secure sender-deniable PKE and computation [CGP, DKR, GP], or as mentioned above even a multi-distributional receiver-deniable FE for all circuits from the even stronger assumption of $di\mathcal{O}$.

[2]We borrow the name "flexible" from Boneh, Lewi, and Wu [BLW15] as the original term "multi-distributional" of O'Neill et al. [OPW11] is used to define a slightly different security property in the recent work by De Caro et al. [CIO16] than we achieve here.

1.1 Our Contributions

In this work, we further narrow this gap by investigating a richer primitive – attribute-based encryption (ABE) [GPSW06, BGG+14, GV15] – *without* the use of obfuscation as a black box primitive. We hope that the techniques developed in this work can further shed light on deniability for even richer schemes such as functional encryption [BSW11, GGH+13, BGG+14, GVW15] under standard assumptions.

- Our main contribution is the construction of a flexibly bi-deniable ABE for poly-sized branching programs (which can compute $\mathsf{NC1}$ via Barrington's theorem [Bar89]) from the standard Learning with Errors assumption [Reg05].

Theorem 1.1. *Under the standard* LWE *assumption, there is a flexibly bi-deniable attribute-based encryption scheme for all poly-size branching programs.*

Recall that in an attribute-based encryption (ABE) scheme for a family of functions $\mathcal{F} : \mathcal{X} \to \mathcal{Y}$, every secret key sk_f is associated with a predicate $f \in \mathcal{F}$, and every ciphertext $\mathsf{ct}_{\boldsymbol{x}}$ is associated with an attribute $\boldsymbol{x} \in \mathcal{X}$. A ciphertext $\mathsf{ct}_{\boldsymbol{x}}$ can be decrypted by a given secret key sk_f to its payload message m only when $f(x) = 0 \in \mathcal{Y}$. Informally, the typical security notion for an ABE scheme is *collusion resistance*, which means no collection of keys can provide information on a ciphertext's message, if the individual keys are not authorized to decrypt the ciphertext in the first place. Intuitively, a bi-deniable ABE must provide both collusion and coercion resistance.

Other contributions of this work can be summarized as:

- A new form of the Extended Learning with Errors (eLWE) assumption [OPW11, AP12, BLP+13], which is convenient in the context of Dual Regev type ABE/FE schemes that apply the Leftover Hash Lemma [DRS04] in their security proofs.

- An explicit, tightened noise growth analysis for lattice-based ABE for branching programs. Prior work used the loose l_∞ norm to give a rough upper bound, which is technically insufficient to achieve deniability using our proof techniques. (We require matching upper *and lower* bounds on post-evaluation noise sizes.)

The eLWE assumption above is roughly the standard LWE assumption, but where the distinguisher also receives "hints" on the LWE sample's noise vector $\boldsymbol{e}$ in the form of inner products, i.e. distributions $\left\{\mathbf{A}, \boldsymbol{b} = \mathbf{A}^T\boldsymbol{s} + \boldsymbol{e}, \boldsymbol{z}, \langle \boldsymbol{z}, \boldsymbol{e}\rangle\right\}$ where (intuitively) $\boldsymbol{z}$ is a decryption key in the real system (which are denoted $\boldsymbol{r}$ elsewhere). Our contribution here is a new reduction from the standard LWE assumption to our correlated variant of extended-LWE, eLWE^+, where the adversary requests arbitrary correlations (expressed as a matrix $\mathbf{R}$) between the hints, in the case of a prime poly-size modulus with noise-less hints. We show this by extending the LWE to eLWE reduction of Alperin-Sheriff and Peikert [AP12] to our setting.

1.2 Our Approach

At a high level, our work begins with the ABE for branching programs of Gorbunov and Vinayagamurthy [GV15]. We will augment the basic ABE-BP = (Setup, Keygen, Enc, Dec) with an additional suite of algorithms ($\mathsf{DenSetup}$, DenEnc, $\mathsf{SendFake}$, $\mathsf{RecFake}$) to form our flexibly bi-deniable ABE-BP. Doing so requires careful attention to the setting of parameters, as we explain in the sequel.

We remark now that – due to reasons related to the delicateness of our parameter setting – the ABE scheme of [GV15] is *particularly suited* to being made bi-deniable, as compared to similar schemes such as the ABE for arithmetic circuits of Boneh et al. [BGG+14]. We will explain this in what follows as well.

Intuition for Our New Deniability Mechanism. As in the work of O'Neill et al. [OPW11], our approach to bi-deniability relies primarily on a curious property of Dual Regev type [GPV08] secret keys: by correctness of any such scheme, each key $\boldsymbol{r}$ is guaranteed to behave as intended for some $1-\mathsf{negl}(n)$ fraction of the possible random coins used to encrypt, but system parameters may be set so that each key is also guaranteed to be *faulty* (i.e. fail to decrypt) on some $\mathsf{negl}(n)$ fraction of the possible encryption randomness. More concretely, each secret key vector $\boldsymbol{r}$ in lattice-based schemes is sampled from an m-dimensional Gaussian distribution, as is the error term $\boldsymbol{e}$ (for LWE public key $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$). For every fixed $\boldsymbol{r}$, with overwhelming probability over the choice of $\boldsymbol{e}$, the vectors $\boldsymbol{r}, \boldsymbol{e} \in \mathbb{Z}_q^m$ will point in highly uncorrelated directions in m-space. However, if the vector $\boldsymbol{r}$ and $\boldsymbol{e}$ happen to point in similar directions, the error magnitude will be (loosely) *squared* during decryption.

Our scheme is based around the idea that a receiver, coerced on honest key-ciphertext pair $(\boldsymbol{r}, \mathsf{ct}^*)$, can use the key authority's *faking key* fk to learn the precise error vector $\boldsymbol{e}^*$ used to construct ct^*. Given $\boldsymbol{e}^*, \boldsymbol{r}$, and fk, the receiver re-samples a fresh secret key $\boldsymbol{r}^*$ that is functionally-equivalent to the honest key $\boldsymbol{r}$, except that $\boldsymbol{r}^*$ is strongly correlated with the vector $\boldsymbol{e}^*$ in ct^*. When the coercer then attempts to decrypt the challenge ciphertext ct^* using $\boldsymbol{r}^*$, the magnitude of decryption error will artificially grow and cause the decryption to output the value we want to deny to. Yet, when the coercer attempts to decrypt any other independently-sampled ciphertext ct, decryption will succeed with overwhelming probability under $\boldsymbol{r}^*$ if it would have under $\boldsymbol{r}$.

We emphasize that to properly show coercion resistance (when extending this intuition to the case of Dual Regev ABE instead of Dual Regev PKE), this behavior of $\boldsymbol{r}^*$ should hold *even when* ct *and* ct^* *embed the same attribute* $\boldsymbol{x}$. (Indeed, the majority of our effort is devoted to ensuring this simple geometric intuition allows a valid instantiation of the denying algorithms $(\mathsf{DenSetup}, \mathsf{DenEnc}, \mathsf{SendFake}, \mathsf{RecFake})$ without "damaging" the basic operation of $(\mathsf{Setup}, \mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec})$ in the underlying ABE scheme.)

Then, given the ability to "artificially blow-up" the decryption procedure of a specific key on a ciphertext-by-ciphertext basis, we can employ an idea originally due to Canetti et al. [CDNO97] of *translucent sets*, but generalized to the setting of ABE instead of PKE, to construct our new, flexibly bi-deniable ABE-BP scheme out of the framework provided by the "plain" SIM-secure ABE-BP scheme of [GV15].

Highlights of the Gorbunov-Vinayagamurthy Scheme. In the ABE for (width 5) branching programs of [GV15], bits a are "LWE-encoded" by the vector

$$\psi_{\mathbf{A},\boldsymbol{s},a} = \boldsymbol{s}^T(\mathbf{A} + a \cdot \mathbf{G}) + \boldsymbol{e} \in \mathbb{Z}_q^m$$

where $\mathbf{G}$ is the gadget matrix [MP12].

The ciphertext ct encrypting message μ under BP-input $\boldsymbol{x}$ is given by

$$\mathsf{ct} = (\psi_0, \psi^c, \{\psi_i\}_{i\in[\ell]}, \{\psi_{0,i}\}_{i\in[5]}, c),$$

and is composed of a Dual Regev ct-pair of vectors (ψ_0, c) encrypting the ciphertext's message μ, an encoding ψ^c representing the (freshly randomized) encoding of the constant 1, five encodings $\{\psi_{0,i}\}_{i\in[5]}$ representing a (freshly randomized) encoding of the initial state of a width-5, length-ℓ branching program BP, and ℓ encodings $\{\psi_i\}_{i\in[\ell]}$ – one for each step of the branching program's evaluation, storing a constant-sized *permutation matrix* associated with the i-th level of BP. Note that each "LWE encoding" ψ is performed under a distinct public key matrix $\mathbf{A}$, $\mathbf{A}^c$, $\{\mathbf{A}_i\}$, or $\{\mathbf{A}_{0,i}\}$ respectively.

The (key-homomorphic) evaluation procedure takes as input a ciphertext $\mathsf{ct} = (\psi_0, \psi^c, \{\psi_i\}, \{\psi_{0,i}\}, c)$ and the public key $\mathsf{pk} = (\mathbf{A}, \mathbf{A}^c, \{\mathbf{A}_i\}, \{\mathbf{A}_{0,i}\})$, as well as the cleartext branching program description BP and the BP-input $\boldsymbol{x}$. It produces the evaluated public key $\mathbf{V}_{\mathsf{BP}}$ and the evaluated encoding $\psi_{\mathsf{BP}(\boldsymbol{x})}$. Given a short secret key *vector* $\boldsymbol{r} \in \mathbb{Z}^{2m}$ matching (some public coset $\boldsymbol{u}$ of) the lattice generated by $[\mathbf{A}|\mathbf{V}_{\mathsf{BP}}] \in \mathbb{Z}^{n\times 2m}$, the encoding *vector* $\psi_{\mathsf{BP}(\boldsymbol{x})}$ (whose Dual Regev encoding-components (ψ_0, c) also match coset $\boldsymbol{u}$) can be decrypted to the message μ if and only if $\mathsf{BP}(\boldsymbol{x}) = \mathsf{accept} = \mathbf{0}$.

On the Necessity of Exact Noise Control. In order to push the intuition for our deniability mechanism through for an ABE of the above form, we must overcome a number of technical hurdles.

The major challenge is an implicit technical requirement to *very tightly control* the precise noise magnitude of evaluated ciphertexts. In previous functional (and homomorphic) encryption schemes from lattices, the emphasis is placed on upper bounding evaluated noise terms, to ensure that they do not grow too large and cause decryption to fail. Moreover, security (typically) holds for any ciphertext noise level at or above the starting ciphertexts' noises. In short, noise growth during evaluation is nearly always undesirable.

As with previous schemes, we too must upper bound the noise growth of evaluated ciphertexts in order to ensure basic correctness of our ABE. But unlike previous schemes, we must take the step of also (carefully) *lower bounding* the noise growth during the branching program evaluation (which technically motivates deviating from the l_∞ norm of prior analyses). This is due to the fact, highlighted above, that producing directional alignment between a key and error term can at most *square* the noise present during decryption. Since coercion resistance requires that it must always be possible to deny any ciphertext originally intended for any honest key, it must be that, with overwhelming probability, every honest key and every honest ciphertext produce evaluated error that is no less than the square root of the maximum noise threshold tolerated.

In a little more detail – as we will later demonstrate in Section 4 – in dimension m there is precisely an expected $\mathsf{poly}(m)$ *gap* in magnitude between the inner products of **(i)** two relatively *orthogonal* key/error vectors $\boldsymbol{r}, \boldsymbol{e}_{\mathsf{BP}(\boldsymbol{x})}$, and **(ii)** two highly *correlated* key/error vectors $\boldsymbol{r}^*, \boldsymbol{e}_{\mathsf{BP}(\boldsymbol{x})}$. The ability to deny is based around our ability to design $\boldsymbol{r}^*$ that are statistically indistinguishable from $\boldsymbol{r}$ in the attacker's view, but where $\boldsymbol{r}^*$ "punctures out" decryptions of ciphertexts with error vectors pointing in the *direction* of $\boldsymbol{e}_{\mathsf{BP}(\boldsymbol{x})}$ in m-space (error-vector directions are unique to each honest ct with overwhelming probability).

Crucially, this approach **generically forces** the use of a *polynomial-sized modulus* q in the scheme.[3] In particular, when error vectors $\boldsymbol{e}$ may (potentially) grow to be some superpolynomial magnitude in the dimension m of the public/secret keys, we totally lose any efficiently testable notion of "error vector orientation in m-space" for the purposes of Dual Regev type decryption.

Further, in order to "correctly trace and distinguish" different orientations throughout the computation of an arbitrary branching program BP, we are required to make careful use of *multi-dimensional Gaussian* distributions. These are sampled using covariance matrices $\mathbf{Q} \in \mathbb{Z}^{m \times m}$ that allow us to succinctly describe the underlying, geometric *randomized rotation* action on error vector *orientations* in m-space with each arithmetic operation of the BP evaluation in the overall ABE-BP scheme. (We use the geometrically-inspired term "rotation matrix" to describe our low-norm matrices $\mathbf{R}$ for this reason.)

An additional subtlety in our new noise analysis is that we require the individual multiplications of the ct evaluation procedure to have *independently sampled* error vectors in each operand-encoding – and thus be "independently oriented" – in order for the overall analysis to go through correctly. (While there could in principle be some way around this technical obstacle in the analysis, we were unable to find one.) This appears to a priori exclude a straightforward denying procedure for *all circuits* [BGG$^+$14], where a gate's input wires' preceding sub-circuits may have cross-wires between them. But it naturally permits denying *branching program computations*, where at the i-th time-step, an i-th *independently generated* ct-component is merged into an accumulated BP state, as with [GV15].

Finally, we mention that an inherent limitation in the techniques of Apon et al. [AFL15], used to construct (the weaker notion of) flexibly bi-deniable inner product encryption from LWE, is bypassed in the current work at the cost of supporting only BP computations of an a-priori bounded length ℓ. Namely, it was the case in [AFL15] that the *length* ℓ of the attribute vector $\boldsymbol{w}$ had to be "traded off" against the dimension

[3] One consequence of a poly-size modulus requirement is that the fully key-homomorphic scheme of Boneh et al. [BGG$^+$14], taken verbatim, can only be denied for up to $\mathsf{NC0}$ functions using our approach. Past this, attempts to produce fake keys in an identical manner to this work may be detected by a statistical test under coercion.

m of the public/secret keys. We suppress the details, other than to point out that this issue can be resolved by artificially boosting the magnitude of the low-norm matrices used to generate error terms in fresh ciphertexts from $\{-1, 1\}$ up to $\{-\Theta(m\ell), \Theta(m\ell)\}$-valued matrices. This, of course, requires knowing the length ℓ of the branching program up front. (Intuitively, this technical change as compared to [AFL15] allows for a sharp *inductive lower bound* on the *minimum* noise growth across all possible function-input pairs that might be evaluated in a given instance of our bi-deniable ABE-BP scheme.)

1.3 Future Directions

The next, most natural question is whether bi-deniable functional encryption can be built out of similar techniques (from only LWE), perhaps by leveraging our bi-deniable ABE for $\mathsf{NC1}$ computations as a building block. We briefly sketch one possible approach and the obstacles encountered. Recall that Goldwasser et al. [GKP$^+$13] show to transform the combination of **(i)** any ABE for a circuit family $\mathcal{C}$, **(ii)** fully homomorphic encryption, and **(iii)** a randomized encoding scheme (such as Yao's garbled circuits) into a 1-key (resp. *bounded collusion*) SIM-secure functional encryption scheme for $\mathcal{C}$.

If we instantiate the Goldwasser et al. transformation with our deniable ABE, we get a functional encryption scheme for $\mathsf{NC1}$. We can then boost functional encryption for shallow circuits to functional encryption for all circuits using the "trojan method" of Ananth et al. [ABSV15]. As it turns out, it is easy to directly prove *flexible receiver-deniability* of the final scheme, independently of but matching the generic results of De Caro et al. [CIO16] for receiver-deniable FE.

Unfortunately, we do not know how to prove (even, flexible) *sender-deniability* of this final scheme. Roughly speaking, the problem is that each ciphertext's attribute in such a scheme contains an FHE ciphertext $\mathsf{ct}_{\mathsf{FHE}}$ for its attribute, and this attribute leaks to the attacker (resp. cocercer) on decryptions that succeed. In particular, there is nothing stopping the coercer from demanding that the sender also provide randomness r_S that *opens the attribute's FHE ciphertext*.

We speculate that a possible way around this obstacle would be to use an *adaptively-secure* homomorphic encryption scheme for $\mathsf{NC1}$ computations. Note that adaptively-secure FHE is known to be impossible for circuits with $\omega(\log(n))$ depth due to a counting argument lower bound by Katz, Thiruvengadam, and Zhou [KTZ13], but this leaves open the possibility of an $\mathsf{NC1}$-homomorphic encryption scheme with the necessary properties to re-obtain (flexible) sender deniability for lattice-based FE. We leave this as an intriguing open problem for future work.

2 Preliminaries

Notations. Let PPT denote probabilistic polynomial time. We use bold uppercase letters to denote matrices, and bold lowercase letters to denote vectors, where vectors are by default column vectors throughout the paper. We let λ be the security parameter, $[n]$ denote the set $\{1, ..., n\}$, and $|\boldsymbol{t}|$ denote the number of bits in a string or vector $\boldsymbol{t}$. We denote the i-th bit value of a string $\boldsymbol{s}$ by $\boldsymbol{s}[i]$. We use $[\cdot|\cdot]$ to denote the concatenation of vectors or matrices, and $||\cdot||$ to denote the norm of vectors or matrices respectively. We use the ℓ_2 norm for all vectors unless explicitly stated otherwise.

We present necessary background knowledge of branching programs and lattices (such as the LWE assumption and lattice sampling algorithms) as follows.

2.1 Branching Program

We recall the definition of branching program in [BV14, GV15]. A width-w branching program BP of length L with input space $\{0,1\}^\ell$ and s states (represented by $[s]$) is a sequence of L tuples of form $(\mathsf{var}(t), \sigma_{t,0}, \sigma_{t,1})$, where

- $\sigma_{t,0}$ and $\sigma_{t,1}$ are injective functions from $[s]$ to itself.
- $\mathsf{var} : [L] \to [\ell]$ is a function that associates the t-th tuple $\sigma_{t,0}, \sigma_{t,1}$ with the input bit $x_{\mathsf{var}(t)}$.

The branching program BP on input $\boldsymbol{x} = (x_1, ..., x_\ell)$ computes its outputs as follows. At step t, we denote the state of the computation by $\eta_t \in [s]$. The initial state is set to be $\eta_0 = 1$. In general, η_t can be computed recursively as

$$\eta_t = \sigma_{t,x_{\mathsf{var}(t)}}(\eta_{t-1})$$

Finally, after L steps, the output of the computation $\mathsf{BP}(\boldsymbol{x}) = 1$ if $\eta_L = 1$ and 0 otherwise. As mentioned in [BV14], we represent states with bits rather than numbers to bound the noise growth. In particular, we represent the state $\eta_t \in [s]$ by a unit vector $\boldsymbol{v}_t \in \{0,1\}^s$. The idea is that $\boldsymbol{v}_t = 1$ if and only if $\sigma_{t,x_{\mathsf{var}(t)}}(\eta_{t-1}) = i$. Note that we can also write the above expression as $\boldsymbol{v}_t[i] = 1$ if and only if either:

- $\boldsymbol{v}_{t-1}[\sigma_{t,0}^{-1}(i)] = 1$ and $x_{\mathsf{var}(t)} = 0$.
- $\boldsymbol{v}_{t-1}[\sigma_{t,1}^{-1}(i)] = 1$ and $x_{\mathsf{var}(t)} = 1$.

This later form will be useful since we can rewrite the above conditions in the following formula. For $t \in [L]$ and $i \in [s]$,

$$\begin{aligned}\boldsymbol{v}_t[i] :&= \boldsymbol{v}[\sigma_{t,0}^{-1}(i)](1 - x_{\mathsf{var}(t)}) + \boldsymbol{v}_{t-1}[\sigma_{t,1}^{-1}(i)] \cdot x_{\mathsf{var}(t)} \\ &= \boldsymbol{v}_{t-1}[\gamma_{t,i,0}](1 - x_{\mathsf{var}(t)}) + \boldsymbol{v}_{t-1}[\gamma_{t,i,1}] \cdot x_{\mathsf{var}(t)}\end{aligned}$$

where we set $\gamma_{t,i,0} = \sigma_{t,i}^{-1}(i)$ and $\gamma_{t,i,1} = \sigma_{t,i}^{-1}(i)$, and $\gamma_{t,i,0}, \gamma_{t,i,1}$ can be publicly computed from the description of the branching program. Hence, $\{\mathsf{var}(t), \{\gamma_{t,i,0}, \gamma_{t,i,1}\}_{i\in[s]}\}$ is also a valid representation of a branching program BP.

For clarity of representation, we will deal with width-5 permutation branching program, which is shown to be equivalent to the circuit class $\mathcal{NC}^1$ [Bar86]. Hence, we have $s = w = 5$, and the functions σ_0, σ_1 are permutations on $[5]$.

2.2 Lattice Background

A full-rank m-dimensional integer lattice $\Lambda \subset \mathbb{Z}^m$ is a discrete additive subgroup whose linear span is $\mathbb{R}^m$. The basis of Λ is a linearly independent set of vectors whose linear combinations are exactly Λ. Every integer lattice is generated as the $\mathbb{Z}$-linear combination of linearly independent vectors $\mathbf{B} = \{\boldsymbol{b}_1, ..., \boldsymbol{b}_m\} \subset \mathbb{Z}^m$. For a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, we define the "$q$-ary" integer lattices:

$$\Lambda_q^\perp = \{\boldsymbol{e} \in \mathbb{Z}^m | \boldsymbol{e}^T\mathbf{A} = 0 \bmod q\}, \qquad \Lambda_q^{\mathbf{u}} = \{\boldsymbol{e} \in \mathbb{Z}^m | \boldsymbol{e}^T\mathbf{A} = \boldsymbol{u} \bmod q\}$$

It is obvious that $\Lambda_q^{\boldsymbol{u}}$ is a coset of $\Lambda_q^\perp$.

Let Λ be a discrete subset of $\mathbb{Z}^m$. For any vector $\boldsymbol{c} \in \mathbb{R}^m$, and any positive parameter $\sigma \in \mathbb{R}$, let $\rho_{\sigma,\boldsymbol{c}}(\boldsymbol{x}) = \exp(-\pi||\boldsymbol{x} - \boldsymbol{c}||^2/\sigma^2)$ be the Gaussian function on $\mathbb{R}^m$ with center $\boldsymbol{c}$ and parameter σ. Next, we set $\rho_{\sigma,\boldsymbol{c}}(\Lambda) = \sum_{\boldsymbol{x}\in\Lambda} \rho_{\sigma,\boldsymbol{c}}(\boldsymbol{x})$ be the discrete integral of $\rho_{\sigma,\boldsymbol{x}}$ over Λ and $\mathcal{D}_{\Lambda,\sigma,\boldsymbol{c}}(\boldsymbol{y}) := \frac{\rho_{\sigma,\boldsymbol{c}}(\boldsymbol{y})}{\rho_{\sigma,\boldsymbol{c}}(\Lambda)}$. We abbreviate σ in the $\rho_{\sigma,\boldsymbol{c}}$ and $\mathcal{D}_{\sigma,\boldsymbol{c}}$ when $\sigma = 0$.

More frequently, we will use the generalized multi-dimensional (or m-variate) Discrete Gaussian distribution $\mathcal{D}_{\mathbb{Z}_1^m,\mathbf{Q}}$, which denotes sampling a $\mathbb{Z}_1$-valued m-vector with *covariance matrix* $\mathbf{Q} \in \mathbb{Z}_1^{m\times m}$. In order to sample from the distribution $\mathcal{D}_{\mathbb{Z}_1^m,\mathbf{Q}}$, proceed as follows:

- Sample $\boldsymbol{t}' = (t'_1, ..., t'_m) \in \mathbb{R}^m$ independently as $t'_i \leftarrow \mathcal{D}_1$ for $i \in [m]$.

- Find the Cholesky decomposition $\mathbf{Q} = \mathbf{L}\mathbf{L}^T$.

- Output the vector $\boldsymbol{t} := \mathbf{L}\boldsymbol{t}'$ as the sample $\boldsymbol{t} \leftarrow \mathcal{D}_{\mathbb{Z}_1^m, \mathbf{Q}}$.

Recall that the Cholesky decomposition takes as input any positive-definite matrix $\mathbf{Q} \in \mathbb{R}^{m \times m}$ and outputs a lower triangular matrix $\mathbf{L}$ so that $\mathbf{Q} = \mathbf{L}\mathbf{L}^T$. Further, we recall the fact that the sum of two m-variate Gaussians with means μ_1, μ_2 and variances $\mathbf{Q}_1, \mathbf{Q}_2$ is an m-variate Gaussian with mean $\mu_1 + \mu_2$ and variance $\mathbf{Q}_1 + \mathbf{Q}_2$.

Next we show a useful lemma that we need for our construction.

Lemma 2.1. *Let $\mathbf{I}_{m \times m}$ be the m-by-m identity matrix, $\mathbf{R} \in \mathbb{R}^{m \times m}$, and $\mathbf{Q} \stackrel{\text{def}}{=} a^2\mathbf{I}_{m \times m} - b^2\mathbf{R}^T\mathbf{R}$ for positive numbers a, b such that $a > b||\mathbf{R}^T||$. Then $\mathbf{Q}$ is positive definite.*

Proof. To show that $\mathbf{Q}$ is positive definite, we need to show that for any column vector $\boldsymbol{x}$ of dimension m, we have $\boldsymbol{x}^T \cdot \mathbf{Q} \cdot \boldsymbol{x} > 0$. We prove this by unfolding the matrix $\mathbf{Q}$:

$$\begin{aligned} \boldsymbol{x}^T \cdot \mathbf{Q} \cdot \boldsymbol{x} &= \boldsymbol{x}^T \cdot (a^2\mathbf{I}_{m \times m} - b^2\mathbf{R}^T\mathbf{R}) \cdot \boldsymbol{x} \\ &= a^2\boldsymbol{x}^T\mathbf{I}_{m \times m}\boldsymbol{x} - b^2\boldsymbol{x}^T\mathbf{R}^T\mathbf{R}\boldsymbol{x} \\ &= a^2||\boldsymbol{x}^T||^2 - b^2||\boldsymbol{x}^T\mathbf{R}^T||^2 \\ &> b^2||\boldsymbol{x}^T||^2 \cdot ||\mathbf{R}^T||^2 - b^2||\boldsymbol{x}^T\mathbf{R}^T||^2. \end{aligned}$$

Since $||\boldsymbol{x}^T|| \cdot ||\mathbf{R}^T|| \geq ||\boldsymbol{x}^T\mathbf{R}^T||$, we can conclude $\boldsymbol{x}^T \cdot \mathbf{Q} \cdot \boldsymbol{x} > 0$. □

Definition 2.2 (Matrix norm)**.** *Let S^m denote the set of vectors in $\mathbb{R}^{m+1}$ whose length is 1. Then the norm of a matrix $\mathbf{R} \in \mathbb{R}^{m \times m}$ is defined to be $\sup_{\boldsymbol{x} \in S^m} ||\boldsymbol{x}^T\mathbf{R}||$.*

We note that throughout the paper, for vector-matrix multiplication, we always multiply the vector on the left-hand side of the matrix. Then we have the following lemma, which bounds the norm for some specified distributions.

Lemma 2.3 ([ABB10])**.** *Regarding the norm defined above, we have the following bounds:*

- *Let $\mathbf{R} \in \{-1, 1\}^{m \times m}$ be sampled uniformly at random, then we have $\mathbf{Pr}[||\mathbf{R}|| > 12\sqrt{2m}] < e^{-2m}$.*
- *Let $\mathbf{R}$ be sampled from $\mathcal{D}_{\mathbb{Z}^{m \times m}, \sigma}$, then we have $\mathbf{Pr}[||\mathbf{R}|| > \sigma\sqrt{m}] < e^{-2m}$.*

Randomness extraction. We will use the following lemma to argue the indistinghishability of two different distributions, which is a generalization of the leftover hash lemma proposed by Dodis et al. [DRS04].

Lemma 2.4 ([ABB10])**.** *Suppose that $m > (n+1)\log q + w(\log n)$. Let $\mathbf{R} \in \{-1, 1\}^{m \times k}$ be chosen uniformly at random for some polynomial $k = k(n)$. Let $\mathbf{A}, \mathbf{B}$ be matrix chosen randomly from $\mathbb{Z}_q^{n \times m}, \mathbb{Z}_q^{n \times k}$ respectively. Then, for all vectors $\boldsymbol{w} \in \mathbb{Z}^m$, the two following distributions are statistically close:*

$$(\mathbf{A}, \mathbf{A}\mathbf{R}, \boldsymbol{w}^T\mathbf{R}) \approx (\mathbf{A}, \mathbf{B}, \boldsymbol{w}^T\mathbf{R})$$

Learning With Errors. The LWE problem was introduced by Regev [Reg05], who showed that solving it *on the average* is as hard as (quantumly) solving several standard lattice problems *in the worst case*.

Definition 2.5 (LWE)**.** *For an integer $q = q(n) \geq 2$, and an error distribution $\chi = \chi(n)$ over $\mathbb{Z}_q$, the learning with errors problem $\mathsf{LWE}_{n,m,q,\chi}$ is to distinguish between the following pairs of distributions:*

$$\{\mathbf{A}, \boldsymbol{b} = \mathbf{A}^T\boldsymbol{s} + \boldsymbol{e}\} \text{ and } \{\mathbf{A}, \boldsymbol{u}\}$$

where $\mathbf{A} \stackrel{\$}{\leftarrow} \mathbb{Z}_q^{n \times m}$, $\boldsymbol{s} \stackrel{\$}{\leftarrow} \mathbb{Z}_q^n$, $\boldsymbol{u} \stackrel{\$}{\leftarrow} \mathbb{Z}_q^m$, and $\boldsymbol{e} \stackrel{\$}{\leftarrow} \chi^m$.

Trapdoors and sampling algorithms. We will use the following algorithms to sample short vectors from specified lattices.

Lemma 2.6 ([GPV08]). *Let q, n, m be positive integers with $q \geq 2$ and sufficiently large $m = \Omega(n \log q)$. There exists a* PPT *algorithm* $\mathsf{TrapGen}(q, n, m)$ *that with overwhelming probability outputs a pair* $(\mathbf{A} \in \mathbb{Z}_q^{n \times m}, \mathbf{T_A} \in \mathbb{Z}^{m \times m})$ *such that* $\mathbf{A}$ *is statistically close to uniform in* $\mathbb{Z}_q^{n \times m}$ *and* $\mathbf{T_A}$ *is a basis for* $\Lambda_q^{\perp}(\mathbf{A})$ *satisfying* $||\mathbf{T_A}|| \leq O(n \log q)$.

Lemma 2.7 ([GPV08, CHKP10, ABB10]). *Let $q > 2, m > n$ and $s > ||\mathbf{T_A}|| \cdot w(\sqrt{\log m + m_1})$. There are two algorithms as follows:*

- *There is an efficient algorithm* $\mathsf{SampleLeft}(\mathbf{A}, \mathbf{B}, \mathbf{T_A}\boldsymbol{u}, s)$*: It takes in* $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$*, a short basis* $\mathbf{T_A}$ *for lattice* $\Lambda_q^{\perp}(\mathbf{A})$*, a matrix* $\mathbf{B} \in \mathbb{Z}_q^{n \times m_1}$*, a vector* $\boldsymbol{u} \in \mathbb{Z}_q^n$ *and a Gaussian parameter* s*, then outputs a vector* $\mathbf{r} \in \mathbb{Z}_q^{m+m_1}$ *such that* $\boldsymbol{r} \in \Lambda_q^{\boldsymbol{u}}(\mathbf{F})$*, where* $\mathbf{F} := (\mathbf{A}|\mathbf{B})$*, and is statistical close to* $D_{\Lambda_q^{\boldsymbol{u}}(\mathbf{F}),s}$.

- *There is an efficient algorithm* $\mathsf{SampleRight}(\mathbf{A}, \mathbf{B}, \mathbf{R}, \mathbf{T_B}, \boldsymbol{u}, s)$*: It takes in* $\mathbf{A} \in \mathbb{Z}_q^{n \times m}, \mathbf{R} \in \mathbb{Z}_q^{m \times n}$*, a matrix* $\mathbf{B} \in \mathbb{Z}_q^{n \times n}$*, a short basis* $\mathbf{T_B}$ *for lattice* $\Lambda_q^{\perp}(\mathbf{B})$*, a vector* $\boldsymbol{u} \in \mathbb{Z}_q^n$ *and a Gaussian parameter* s*, then outputs a vector* $\boldsymbol{r} \in \mathbb{Z}_q^{m+n}$ *such that* $\boldsymbol{r} \in \Lambda_q^{\mathbf{u}}(\mathbf{F})$*, where* $\mathbf{F} := (\mathbf{A}|\mathbf{AR}+\mathbf{B})$*, and is statistical close to* $D_{\Lambda_q^{\boldsymbol{u}}(\mathbf{F}),s}$.

- *There is a deterministic polynomial-time algorithm* $\mathsf{ExtBasis}(\mathbf{A}, \mathbf{T_A}, \mathbf{A}')$ *that takes in an arbitrary* $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$*, whose columns generate the entire group* $\mathbf{Z}_q^n$*, an arbitrary basis* $\mathbf{T_A} \in \mathbb{Z}^{m \times m}$ *of* $\Lambda^{\perp}(\mathbf{A})$*, then outputs a basis* $\mathbf{T}'$ *of* $\Lambda^{\perp}(\mathbf{A}|\mathbf{A}')$*, such that* $||\mathbf{T}|| = ||\mathbf{T_A}||$*. Moreover, the same holds even for any given permutation of columns of* $\mathbf{A}'$.

- *There is a deterministic polynomial time algorithm* $\mathsf{Invert}(\mathbf{A}, \mathbf{T_A}, \boldsymbol{b})$ *that, given any* $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ *with its trapdoor* $\mathbf{T_A} \in \mathbb{Z}_q^{m \times m}$ *such that* $||\mathbf{T}|| \cdot w(\sqrt{\log n}) \leq 1/\beta$ *for some* $\beta > 0$*, and* $\boldsymbol{b} = \boldsymbol{s}^T\mathbf{A} + \boldsymbol{x}$ *for arbitrary* $\boldsymbol{s} \in \mathbb{Z}_q^n$ *and random* $\boldsymbol{x} \leftarrow D_{\beta}^m$*, outputs* $\boldsymbol{x}$ *with overwhelming probability.*

- *There is a* PPT *algorithm* $\mathsf{SampleD}(\mathbf{T}, \boldsymbol{c}, s)$ *that, given arbitrary* $\boldsymbol{c} \in \mathbb{R}^m$ *and* $r \geq ||\widetilde{T}|| \cdot w(\log n)$*, generates a sample from* $\mathcal{D}_{\Lambda+c,r}$ *(up to* $\mathsf{negl}(n)$ *statistical distance).*

3 New Definitions and Tools

In this section, we first describe our new notion of flexibly bi-deniable ABE, which is a natural generalization of the flexibly bi-deniable PKE of [OPW11]. Then we define the notion of a flexibly attribute-based bi-translucent set (AB-BTS), which generalizes the idea of bi-translucent set (BTS) in the work [OPW11]. Using a similar argument as in the work [OPW11], we can show that an AB-BTS suffices to construct bi-deniable ABE. In the last part of this section, we define a new assumption called Extended LWE Plus, and show its hardness by giving a reduction from the standard LWE problem.

3.1 Flexibly Bi-Deniable ABE: Syntax and Deniability Definition

A flexibly bi-deniable key-policy attribute based encryption for a class of Boolean circuits $\mathcal{C} : \{0,1\}^{\ell} \to \{0,1\}$ consists a tuple of PPT algorithms $\Pi = (\mathsf{Setup}, \mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{DenSetup}, \mathsf{DenEnc}, \mathsf{SendFake}, \mathsf{RecFake})$. We describe them in detail as follows:

$\mathsf{Setup}(1^{\lambda})$: On input the security parameter λ, the setup algorithm outputs public parameters pp and master secret key msk.

$\mathsf{Keygen}(\mathsf{msk}, f)$: On input the master secret key msk and a function $f \in \mathcal{C}$, it outputs a secret key sk_f.

$\mathsf{Enc}(\mathsf{pp}, \boldsymbol{x}, \mu; r_S)$: On input the public parameter pp, an attribute/message pair $(\boldsymbol{x}, \mu)$ and randomness r_S, it outputs a ciphertext $c_{\boldsymbol{x}}$.

$\mathsf{Dec}(\mathsf{sk}_f, c_{\boldsymbol{x}})$: On input the secret key sk_f and a ciphertext $c_{\boldsymbol{x}}$, it outputs the corresponding plaintext μ if $f(\boldsymbol{x}) = 0$; otherwise, it outputs $\perp$.

$\mathsf{DenSetup}(1^\lambda)$: On input the security parameter λ, the deniable setup algorithm outputs pubic parameters pp, master secret key msk and faking key fk.

$\mathsf{DenEnc}(\mathsf{pp}, \boldsymbol{x}, \mu; r_S)$: On input the public parameter pp, an attribute/message pair $(\boldsymbol{x}, \mu)$ and randomness r_S, it outputs a ciphertext $c_{\boldsymbol{x}}$.

$\mathsf{SendFake}(\mathsf{pp}, r_S, \mu, \mu')$: On input public parameters pp, original random coins r_S, message μ of DenEnc and desired message μ', it outputs a faked random coin r'_S.

$\mathsf{RecFake}(\mathsf{pp}, \mathsf{fk}, c_{\boldsymbol{x}}, f, \mu')$: On input public parameters pp, faking key fk, a ciphertext $c_{\boldsymbol{x}}$, a function $f \in \mathcal{C}$, and desired message μ', the receiver faking algorithm outputs a faked secret key sk'_f.

Correctness. We say the flexibly bi-deniable ABE scheme described above is correct, if for any $(\mathsf{msk}, \mathsf{pp}) \leftarrow \mathsf{S}(1^\lambda)$, where $\mathsf{S} \in \{\mathsf{Setup}, \mathsf{DenSetup}\}$, any message μ, function $f \in \mathcal{C}$, and any attribute vector $\boldsymbol{x}$ where $f(\boldsymbol{x}) = 0$, we have $\mathsf{Dec}(\mathsf{sk}_f, c_{\boldsymbol{x}}) = \mu$, where $\mathsf{sk}_f \leftarrow \mathsf{Keygen}(\mathsf{msk}, f)$ and $c_{\boldsymbol{x}} \leftarrow \mathsf{E}(\mathsf{pp}, \boldsymbol{x}, \mu; r_S)$ where $\mathsf{E} \in (\mathsf{Enc}, \mathsf{DenEnc})$.

Bi-deniability definition. Let μ, μ' be two arbitrary messages, not necessarily different. We propose the bi-deniability definition by describing real experiment $\mathbf{Expt}^{\mathsf{Real}}_{\mathcal{A},\mu,\mu'}(1^\lambda)$ and faking experiment $\mathbf{Expt}^{\mathsf{Fake}}_{\mathcal{A},\mu,\mu'}(1^\lambda)$ regarding adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ below:

1. $(\boldsymbol{x}^*, \mathsf{state}_1) \leftarrow \mathcal{A}_1(\lambda)$
2. $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$
3. $c'_{\boldsymbol{x}^*} \leftarrow \mathsf{Enc}(\mathsf{pp}, \boldsymbol{x}^*, \mu; r_S)$
4. $(f^*, \mathsf{state}_2) \leftarrow \mathcal{A}_2^{\mathsf{KG}(\mathsf{msk}, \boldsymbol{x}^*, \cdot)}(\mathsf{pp}, \mathsf{state}_1, c_{\boldsymbol{x}^*})$
5. $\mathsf{sk}_{f^*} \leftarrow \mathsf{Keygen}(\mathsf{msk}, f^*)$
6. $b \leftarrow \mathcal{A}_3^{\mathsf{KG}(\mathsf{msk}, \boldsymbol{x}^*, \cdot)}(\mathsf{sk}_{f^*}, c, \mathsf{state}_2, r_S)$
7. Output $b \in \{0, 1\}$

(a) $\mathbf{Expt}^{\mathsf{Real}}_{\mathcal{A}}(1^\lambda)$

1. $(\boldsymbol{x}^*, \mathsf{state}_1) \leftarrow \mathcal{A}_1(\lambda)$
2. $(\mathsf{pp}, \mathsf{msk}, \mathsf{fk}) \leftarrow \mathsf{DenSetup}(1^\lambda)$
3. $c'_{\boldsymbol{x}^*} \leftarrow \mathsf{DenEnc}(\mathsf{pp}, \boldsymbol{x}^*, \mu'; r_S)$
4. $(f^*, \mathsf{state}_2) \leftarrow \mathcal{A}_2^{\mathsf{KG}(\mathsf{msk}, \boldsymbol{x}^*, \cdot)}(\mathsf{pp}, \mathsf{state}_1, c'_{\boldsymbol{x}^*})$
5. $r'_S \leftarrow \mathsf{SendFake}(\mathsf{pp}, \mu, \mu', r_S)$
6. $\mathsf{sk}_{f^*} \leftarrow \mathsf{RecFake}(\mathsf{pp}, \mathsf{fk}, c'_{\boldsymbol{x}}, \boldsymbol{v}^*, \mu')$
7. $b \leftarrow \mathcal{A}_3^{\mathsf{KG}(\mathsf{msk}, \boldsymbol{x}^*, \cdot)}(\mathsf{sk}_{f^*}, c, \mathsf{state}_2, r'_S)$
8. Output $b \in \{0, 1\}$

(b) $\mathbf{Expt}^{\mathsf{Fake}}_{\mathcal{A}}(1^\lambda)$

Figure 1: Security experiments for bi-deniable ABE

where $\mathsf{KG}(\mathsf{msk}, \boldsymbol{w}^*, \cdot)$ returns a secret key $\mathsf{sk}_{\boldsymbol{v}} \leftarrow \mathsf{Keygen}(\mathsf{msk}, \boldsymbol{v})$ if $\langle \boldsymbol{v}, \boldsymbol{w}^* \rangle \neq 0$ and $\perp$ otherwise.

Definition 3.1 (Flexibly Bi-Deniable ABE)**.** *An ABE scheme Π is bi-deniable if for any two messages μ, μ', any probabilistic polynomial-time adversaries $\mathcal{A}$ where $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$, there is a negligible function* $\mathsf{negl}(\lambda)$ *such that*

$$\mathbf{Adv}^{\Pi}_{\mathcal{A},\mu,\mu'}(1^\lambda) = |\mathbf{Pr}[\mathbf{Expt}^{\mathsf{Real}}_{\mathcal{A},\mu,\mu'}(1^\lambda) = 1] - \mathbf{Pr}[\mathbf{Expt}^{\mathsf{Fake}}_{\mathcal{A},\mu,\mu'}(1^\lambda) = 1]| \leq \mathsf{negl}(\lambda)$$

3.2 Attribute Based Bitranslucent Set Scheme

In this section, we define the notion of a *Attribute Based Bitranslucent Set* (AB-BTS), which is an extension of bitranslucent sets (BTS) as defined by O'Neill et al. in [OPW11]. Our new notion permits a more fine-grained degree of access control, where pseudorandom samples and secret keys are associated with attributes $\boldsymbol{x}$, and the testing algorithm can successfully distinguish a pseudorandom sample from a truly random one if and only if the attribute of the sample is accepted under a given secret key's policy f – i.e. when $f(\boldsymbol{x}) = \mathbf{0}$. This concept is reminiscent of *attribute-based encryption* (ABE), and in fact, we will show in the sequel how to construct a flexibly bi-deniable ABE from an AB-BTS. This is analogous to the construction of a flexibly bi-deniable PKE from O'Neill et al.'s BTS. We present the formal definition below.

Let $\mathcal{F}$ be some family of functions. An attribute based bitranslucent set (AB-BTS) scheme for $\mathcal{F}$ consists of the following algorithms:

$\mathsf{Setup}(1^\lambda)$: On input the security parameter, the normal setup algorithm outputs a public parameter pp and master secret key msk.

$\mathsf{DenSetup}(1^\lambda)$: On input the security parameter, the deniable setup algorithm outputs a public parameter pp, master secret key msk and faking key fk.

$\mathsf{Keygen}(\mathsf{msk}, f)$: On input the master secret key msk and a function $f \in \mathcal{F}$, the key generation algorithm outputs a secret key sk_f.

P- and U-samplers $\mathsf{SampleP}(\mathsf{pp}, \boldsymbol{x}; r_S)$ and $\mathsf{SampleU}(\mathsf{pp}, \boldsymbol{x}; r_S)$ output some $\boldsymbol{c}$.

$\mathsf{TestP}(\mathsf{sk}_f, \boldsymbol{c}_{\boldsymbol{x}})$: On input a secret key sk_f and a ciphertext $\boldsymbol{c}_{\boldsymbol{x}}$, the P-tester algorithm outputs 1 (accepts) or 0 (rejects).

$\mathsf{FakeSCoins}(\mathsf{pp}, r_S)$: On input a public parameters pp and randomness r_S, the sender-faker algorithm outputs randomness r_S^*.

$\mathsf{FakeRCoins}(\mathsf{pp}, \mathsf{fk}, \boldsymbol{c}_{\boldsymbol{x}}, f)$: On input a public parameters pp, the faking key fk, a ciphertext $\boldsymbol{c}_{\boldsymbol{x}}$ and a function $f \in \mathcal{F}$, the receiver-faker algorithm outputs a faked secret key sk'_f.

Definition 3.2 (AB-BTS)**.** *We say a scheme* $\Pi = (\mathsf{Setup}, \mathsf{DenSetup}, \mathsf{Keygen}, \mathsf{SampleP}, \mathsf{SampleU}, \mathsf{TestP}, \mathsf{FakeSCoins}, \mathsf{FakeRCoins})$ *is an AB-BTS scheme for a function family* $\mathcal{F}$ *if it satisfies:*

1. *(Correctness.) The following experiments accept or respectively reject with overwhelming probability over the randomness.*
   - *Let* $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$, $f \in \mathcal{F}$, $\mathsf{sk}_f \leftarrow \mathsf{Keygen}(\mathsf{msk}, f)$. *If* $f(\boldsymbol{x}) = 0$ *and* $\boldsymbol{c}_{\boldsymbol{x}} \leftarrow \mathsf{SampleP}(\mathsf{pp}, \boldsymbol{x}; r_S)$, *then* $\mathsf{TestP}(\mathsf{sk}_f, \boldsymbol{c}_{\boldsymbol{x}}) = 1$*; otherwise,* $\mathsf{TestP}(\mathsf{sk}_f, \boldsymbol{c}_{\boldsymbol{x}}) = 0$.
   - *Let* $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$, $f \in \mathcal{F}$, $\mathsf{sk}_f \leftarrow \mathsf{Keygen}(\mathsf{msk}, f)$, $\boldsymbol{c} \leftarrow \mathsf{SampleU}(\mathsf{pp}; r_S)$. *Then* $\mathsf{TestP}(\mathsf{sk}_f, \boldsymbol{c}) = 0$.
2. *(Indistinguishable public parameters.) The public parameters* pp *generated by the two setup algorithms* $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ *and* $(\mathsf{pp}, \mathsf{msk}, \mathsf{fk}) \leftarrow \mathsf{DenSetup}(1^\lambda)$ *should be indistinguishable.*
3. *(Selective bi-deniability.) Let* $\mathcal{F}$ *be a family of functions. We define the following two experiments: the real experiment* $\mathbf{Expt}^{\mathsf{Real}}_{\mathcal{A},\mathcal{F}}(1^\lambda)$ *and the faking experiment* $\mathbf{Expt}^{\mathsf{Fake}}_{\mathcal{A},\mathcal{F}}(1^\lambda)$ *regarding an adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ *below:*
   where $\mathsf{KG}(\mathsf{msk}, \boldsymbol{x}^*, \cdot)$ *returns a secret key* $\mathsf{sk}_f \leftarrow \mathsf{Keygen}(\mathsf{msk}, f)$ *if* $f \in \mathcal{F}$ *and* $f(\boldsymbol{x}^*) \neq 0$*; it returns* $\perp$ *otherwise. We also require that* $f^* \in \mathcal{F}$.

(a) $(f^*, \boldsymbol{x}^*, \mathsf{state}_1) \leftarrow \mathcal{A}_1(\lambda)$
(b) $(\mathsf{pp}, \mathsf{msk}, \mathsf{fk}) \leftarrow \mathsf{DenSetup}(1^\lambda)$
(c) $\boldsymbol{c} \leftarrow \mathsf{SampleU}(\mathsf{pp}; r_S)$
(d) $\mathsf{state}_2 \leftarrow \mathcal{A}_2^{\mathsf{KG}(\mathsf{msk}, \boldsymbol{x}^*, \cdot)}(\mathsf{pp}, \mathsf{state}_1, \boldsymbol{c})$
(e) $\mathsf{sk}_{f^*} \leftarrow \mathsf{Keygen}(\mathsf{msk}, f^*)$
(f) $b \leftarrow \mathcal{A}_3^{\mathsf{KG}(\mathsf{msk}, \boldsymbol{x}^*, \cdot)}(\mathsf{sk}_{f^*}, c, \mathsf{state}_2, r_S)$
(g) Output $b \in \{0, 1\}$

(a) $\mathbf{Expt}_{\mathcal{A}}^{\mathsf{Real}}(1^\lambda)$

(a) $(f^*, \boldsymbol{x}^*, \mathsf{state}_1) \leftarrow \mathcal{A}_1(\lambda)$
(b) $(\mathsf{pp}, \mathsf{msk}, \mathsf{fk}) \leftarrow \mathsf{DenSetup}(1^\lambda)$
(c) $\boldsymbol{c} \leftarrow \mathsf{SampleP}(\mathsf{pp}, \boldsymbol{x}^*; r_S)$
(d) $\mathsf{state}_2 \leftarrow \mathcal{A}_2^{\mathsf{KG}(\mathsf{msk}, \boldsymbol{x}^*, \cdot)}(\mathsf{pp}, \mathsf{state}_1, \boldsymbol{c})$
(e) $r'_S \leftarrow \mathsf{FakeSCoins}(\mathsf{pp}, r_S)$
(f) $\mathsf{sk}_{f^*} \leftarrow \mathsf{FakeRCoins}(\mathsf{pp}, \mathsf{fk}, \boldsymbol{c}, f^*)$
(g) $b \leftarrow \mathcal{A}_3^{\mathsf{KG}(\mathsf{msk}, \boldsymbol{x}^*, \cdot)}(\mathsf{sk}_{f^*}, c, \mathsf{state}_2, r'_S)$
(h) Output $b \in \{0, 1\}$

(b) $\mathbf{Expt}_{\mathcal{A}}^{\mathsf{Fake}}(1^\lambda)$

Figure 2: Security experiments for AB-BTS

We say the scheme is selectively bi-deniable for $\mathcal{F}$, if for any probabilistic polynomial-time adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$, there is a negligible function $\mathsf{negl}(\lambda)$ such that

$$\mathbf{Adv}_{\mathcal{A}}^{\Pi}(1^\lambda) = |\mathbf{Pr}[\mathbf{Expt}_{\mathcal{A},\mathcal{F}}^{\mathsf{Real}}(1^\lambda) = 1] - \mathbf{Pr}[\mathbf{Expt}_{\mathcal{A},\mathcal{F}}^{\mathsf{Fake}}(1^\lambda) = 1]| \leq \mathsf{negl}(\lambda)$$

Remark 3.3. *Correctness for the faking algorithms is implied by the bi-deniability property. In particular, with overwhelming probability over the overall randomness, the following holds: let $(\mathsf{pp}, \mathsf{msk}, \mathsf{fk}) \leftarrow \mathsf{DenSetup}(1^\lambda)$, $f \in \mathcal{F}$, $\mathsf{sk}_f \leftarrow \mathsf{Keygen}(\mathsf{msk}, f)$, x be a string and $\boldsymbol{c}_{\boldsymbol{x}} \leftarrow \mathsf{SampleP}(\mathsf{pp}, x; r_S)$, then*

- $\mathsf{SampleU}(\mathsf{pp}; \mathsf{FakeSCoins}(\mathsf{pp}, r_S)) = \boldsymbol{c}_{\boldsymbol{x}}$,
- $\mathsf{TestP}(\mathsf{FakeRCoins}(\mathsf{pp}, \mathsf{fk}, \boldsymbol{c}_{\boldsymbol{x}}, f), \boldsymbol{c}_{\boldsymbol{x}}) = 0$
- *For any other x', let $\boldsymbol{c}' \leftarrow \mathsf{SampleP}(\mathsf{pp}, x'; r'_S)$, then (with overwhelming probability) we have*

$$\mathsf{TestP}\left(\mathsf{FakeRCoins}(\mathsf{pp}, \mathsf{fk}, \boldsymbol{c}_{\boldsymbol{x}}, f), \boldsymbol{c}'\right) = \mathsf{TestP}(\mathsf{sk}_f, c').$$

It is not hard to see that if one of these does not hold, then one can easily distinguish the real experiment from the faking experiment.

Remark 3.4. *Canetti et al. [CDNO97] gave a simple encoding technique to construct a sender-deniable encryption scheme from a translucent set. O'Neill, Peikert, and Waters [OPW11] used a similar method to construct a flexibly bi-deniable encryption from a bi-translucent set scheme. Here we further observe that the same method as well allows us to construct a flexibly bi-deniable ABE scheme from bi-deniable AB-BTS. We present the construction in Section 4.4.*

3.3 Extended LWE and Our New Variant

O'Neill et al. [OPW11] introduced the Extended LWE problem, which allows a "hint" on the error vector $\boldsymbol{x}$ to leak in form of a noisy inner product. They observe a trivial "blurring" argument shows that LWE reduces to eLWE when the hint-noise βq is superpolynomially larger than the magnitude of samples from χ, and also allows for unboundedly many *independent* hint vectors $\langle \boldsymbol{z}, \boldsymbol{x}_i \rangle$ while retaining LWE-hardness.

Definition 3.5 (Extended LWE)**.** *For an integer $q = q(n) \geq 2$, and an error distribution $\chi = \chi(n)$ over $\mathbb{Z}_q$, the extended learning with errors problem $\mathsf{eLWE}_{n,m,q,\chi,\beta}$ is to distinguish between the following pairs of distributions:*

$$\{\mathbf{A}, \boldsymbol{b} = \mathbf{A}^T \boldsymbol{s} + \boldsymbol{e}, \boldsymbol{z}, \langle \boldsymbol{z}, \boldsymbol{b} - \boldsymbol{e} \rangle + e'\} \text{ and } \{\mathbf{A}, \boldsymbol{u}, \boldsymbol{z}, \langle \boldsymbol{z}, \boldsymbol{u} - \boldsymbol{x} \rangle + e'\}$$

where $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$, $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_q^n$, $\boldsymbol{u} \xleftarrow{\$} \mathbb{Z}_q^m$, $\boldsymbol{e}, \boldsymbol{z} \xleftarrow{\$} \chi^m$ and $e' \xleftarrow{\$} \mathcal{D}_{\beta q}$.

Further, Alperin-Sheriff and Peikert [AP12] show that LWE reduces to eLWE with a polynomial modulus and no hint-noise (i.e. $\beta = 0$), even in the case of a bounded number of *independent* hints.

We introduce the following new form of extended-LWE, called eLWE^+, which considers leaking a pair of *correlated hints* on the same noise vector. Our security proof of the AB-BTS construction relies on this new assumption.

Definition 3.6 (Extended LWE Plus)**.** *For integer $q = q(n) \geq 2, m = m(n)$, an error distribution $\chi = \chi(n)$ over $\mathbb{Z}_q$, and a matrix $\mathbf{R} \in \mathbb{Z}_q^{m \times m}$, the extended learning with errors problem $\mathsf{eLWE}^+_{n,m,q,\chi,\beta,\mathbf{R}}$ is to distinguish between the following pairs of distributions:*

$$\{\mathbf{A}, \boldsymbol{b} = \mathbf{A}^T\boldsymbol{s} + \boldsymbol{e}, \boldsymbol{z}_0, \boldsymbol{z}_1, \langle \boldsymbol{z}_0, \boldsymbol{b} - \boldsymbol{e}\rangle + e, \langle \mathbf{R}\boldsymbol{z}_1, \boldsymbol{b} - \boldsymbol{e}\rangle + e'\} \text{ and}$$

$$\{\mathbf{A}, \boldsymbol{u}, \boldsymbol{z}_0, \boldsymbol{z}_1, \langle \boldsymbol{z}_0, \boldsymbol{u} - \boldsymbol{e}\rangle + e, \langle \mathbf{R}\boldsymbol{z}_1, \boldsymbol{u} - \boldsymbol{e}\rangle + e'\}$$

where $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$, $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_q^n$, $\boldsymbol{u} \xleftarrow{\$} \mathbb{Z}_q^m$, $\boldsymbol{e}, \boldsymbol{z}_0, \boldsymbol{z}_1 \xleftarrow{\$} \chi^m$ and $e, e' \xleftarrow{\$} \mathcal{D}_{\beta q}$.

Hardness of extended-LWE$^+$. A simple observation, following prior work, is that when χ is $\mathsf{poly}(n)$-bounded and the hint noise βq (and thus, modulus q) is superpolynomial in n, then $\mathsf{LWE}_{n,m,q,\chi}$ trivially reduces to $\mathsf{eLWE}^+_{n,m,q,\chi,\beta,\mathbf{R}}$ for every $\mathbf{R} \in \mathbb{Z}_q^{m\times m}$ so that $\mathbf{R}\mathbf{z}_1$ has $\mathsf{poly}(n)$-bounded norm. This is because, for any $r = \omega(\sqrt{\log n}), c \in \mathbb{Z}$, the statistical distance between $\mathcal{D}_{\mathbb{Z},r}$ and $c + \mathcal{D}_{\mathbb{Z},r}$ is at most $O(|c|/r)$.

However, our cryptosystem will require a polynomial-size modulus q. So, we next consider the case of *prime* modulus q of $\mathsf{poly}(n)$ size and no noise on the hints (i.e. $\beta = 0$). Following [AP12][4], it will be convenient to swap to the "knapsack" form of LWE, which is: given $\mathbf{H} \leftarrow \mathbb{Z}_q^{(m-n)\times m}$ and $\boldsymbol{c} \in \mathbb{Z}_q^{m-n}$, where either $\mathbf{c} = \mathbf{H}\boldsymbol{e}$ for $\boldsymbol{e} \leftarrow \chi^m$ or $\mathbf{c}$ uniformly random and independent of $\mathbf{H}$, determine which is the case (with non-negligible advantage). The "extended-plus" form of the knapsack problem also reveals a pair of hints $(\boldsymbol{z}_0, \boldsymbol{z}_1, \langle \boldsymbol{z}_0, \boldsymbol{e}\rangle, \langle \mathbf{R}\boldsymbol{z}_1, \boldsymbol{e}\rangle)$. Note the equivalence between LWE and knapsack-LWE is proven in [MM11] for $m \geq n + \omega(\log n)$.

Theorem 3.7. *For $m \geq n + \omega(\log n)$, for every prime $q = \mathsf{poly}(n)$, for every $\mathbf{R} \in \mathbb{Z}_q^{m\times m}$, and for every $\beta \geq 0$, $\mathbf{Adv}^{\mathsf{LWE}_{n,m,q,\chi}}_{\mathcal{B}^{\mathcal{A}}}(1^\lambda) \geq (1/q^2)\mathbf{Adv}^{\mathsf{eLWE}^+_{n,m,q,\chi,\beta,\mathbf{R}}}_{\mathcal{A}}(1^\lambda)$.*

Proof. We construct an LWE to eLWE^+ reduction $\mathcal{B}$ as follows. $\mathcal{B}$ receives a knapsack-LWE instance $\mathbf{H} \in \mathbb{Z}_q^{(m-n)\times m}, \boldsymbol{c} \in \mathbb{Z}_q^{m-n}$. It samples $\boldsymbol{e}', \boldsymbol{z}_0, \boldsymbol{z}_1 \leftarrow \chi^m$ and uniform $\boldsymbol{v}_0, \boldsymbol{v}_1 \leftarrow \mathbb{Z}_q^{m-n}$. It chooses any $\mathbf{R} \in \mathbb{Z}_q^{m\times m}$, then sets

$$\mathbf{H}' := \mathbf{H} - \boldsymbol{v}_0\boldsymbol{z}_0^T - \boldsymbol{v}_1\left(\mathbf{R}\boldsymbol{z}_1\right)^T \in \mathbb{Z}_q^{(m-n)\times m},$$
$$\boldsymbol{c}' := \boldsymbol{c} - \boldsymbol{v}_0 \cdot \langle \boldsymbol{z}_0, \boldsymbol{e}'\rangle - \boldsymbol{v}_1 \cdot \langle \mathbf{R}\boldsymbol{z}_1, \boldsymbol{e}'\rangle \in \mathbb{Z}_q^{m-n}.$$

It sends $(\mathbf{H}', \boldsymbol{c}', \mathbf{z}_0, \mathbf{z}_1, \langle \boldsymbol{z}_0, \boldsymbol{e}'\rangle, \langle \mathbf{R}\boldsymbol{z}_1, \boldsymbol{e}'\rangle)$ to the knapsack-eLWE^+ adversary $\mathcal{A}$, and outputs what $\mathcal{A}$ outputs.

Notice that when $\mathbf{H}, \boldsymbol{c}$ are independent and uniform, so are $\mathbf{H}', \mathbf{c}'$, in which case $\mathcal{B}$'s simulation is perfect.

Now, consider the case when $\mathbf{H}, \mathbf{c}$ are drawn from the knapsack-LWE distribution, with $\mathbf{c} = \mathbf{H}\boldsymbol{x}$ for $\boldsymbol{e} \leftarrow \chi^m$. In this case, $\mathbf{H}'$ is uniformly random over the choice of $\mathbf{H}$, and we have

$$\begin{aligned}
\boldsymbol{c}' &= \mathbf{H}\boldsymbol{x} - \boldsymbol{v}_0 \cdot \langle \boldsymbol{z}_0, \boldsymbol{e}'\rangle - \boldsymbol{v}_1 \cdot \langle \mathbf{R}\boldsymbol{z}_1, \boldsymbol{e}'\rangle \\
&= \left(\mathbf{H}' + \boldsymbol{v}_0\boldsymbol{z}_0^T + \boldsymbol{v}_1\left(\mathbf{R}\boldsymbol{z}_1\right)^T\right)\boldsymbol{e} - \boldsymbol{v}_0 \cdot \langle \boldsymbol{z}_0, \boldsymbol{e}'\rangle - \boldsymbol{v}_1 \cdot \langle \mathbf{R}\boldsymbol{z}_1, \boldsymbol{e}'\rangle \\
&= \mathbf{H}'\boldsymbol{e} + \boldsymbol{v}_0 \cdot \langle \boldsymbol{z}_0, \boldsymbol{e} - \boldsymbol{e}'\rangle + \boldsymbol{v}_1 \cdot \langle \mathbf{R}\boldsymbol{z}_1, \boldsymbol{e} - \boldsymbol{e}'\rangle.
\end{aligned}$$

[4] We note that a higher quality reduction from LWE to eLWE is given in [BLP+13] in the case of binary secret keys. However for our cryptosystem, it will be more convenient to have secret key coordinates in $\mathbb{Z}_q$, so we extend the reduction of [AP12] to eLWE^+ instead.

Define the event $E = [E_0 \wedge E_1]$ as

$$E_0 \stackrel{\text{def}}{=} \left[\langle \boldsymbol{z}_0, \boldsymbol{e}\rangle = \langle \boldsymbol{z}_0, \boldsymbol{e}'\rangle\right],$$
$$E_1 \stackrel{\text{def}}{=} \left[\langle \mathbf{R}\boldsymbol{z}_1, \boldsymbol{e}\rangle = \langle \mathbf{R}\boldsymbol{z}_1, \boldsymbol{e}'\rangle\right].$$

If event E occurs, then the reduction $\mathcal{B}$ perfectly simulates a pseudorandom instance of knapsack-eLWE$^+$ to $\mathcal{A}$, as then $\boldsymbol{v}_0 \cdot \langle \boldsymbol{z}_0, \boldsymbol{e} - \boldsymbol{e}'\rangle + \boldsymbol{v}_1 \cdot \langle \mathbf{R}\boldsymbol{z}_1, \boldsymbol{e} - \boldsymbol{e}'\rangle$ vanishes, leaving $\boldsymbol{c}' = \mathbf{H}'\boldsymbol{e}$ for $\mathbf{H}' \leftarrow \mathbb{Z}_q^{(m-n)\times m}$ and $\boldsymbol{e} \leftarrow \chi^m$ as required. Otherwise since q is prime, the reduction $\mathcal{B}$ (incorrectly) simulates an independent and uniform instance of knapsack-eLWE$^+$ to $\mathcal{A}$, as then either one of $\boldsymbol{v}_0 \cdot \langle \boldsymbol{z}_0, \boldsymbol{e} - \boldsymbol{e}'\rangle$ or $\boldsymbol{v}_1 \cdot \langle \mathbf{R}\boldsymbol{z}_1, \boldsymbol{e} - \boldsymbol{e}'\rangle$ does not vanish, implying that $\boldsymbol{c}'$ is uniform in $\mathbb{Z}_q^{m-n}$ over the choice of $\boldsymbol{v}_0$ (resp. $\boldsymbol{v}_1$) alone, independent of the choices of $\mathbf{H}'$ and $\boldsymbol{x}$.

It remains to analyze the probability that event E occurs. Because $\boldsymbol{e}$ and $\boldsymbol{e}'$ are i.i.d., we may define the random variable $\mathcal{Z}_0$ that takes values $\langle \boldsymbol{z}_0, \boldsymbol{e}^*\rangle \in \mathbb{Z}_q$ and the random variable $\mathcal{Z}_1$ that takes values $\langle \mathbf{R}\boldsymbol{z}_1, \boldsymbol{e}^*\rangle \in \mathbb{Z}_q$ jointly over choice of $\boldsymbol{e}^* \leftarrow \chi^m$, and analyze their collision probabilities independently. Since the collision probability of *any* random variable $\mathcal{Z}$ is at least $1/|\mathsf{Supp}(\mathcal{Z})|$, we have that $\Pr[E] \geq \min CP[\mathcal{Z}_0] \cdot \min CP[\mathcal{Z}_1] = 1/q^2 = 1/\mathsf{poly}(n)$, and the theorem follows. □

4 Flexibly Bi-Deniable Attribute-Based Encryption (ABE) for Branching Programs

In this section, we present our flexibly bi-deniable ABE for bounded-length Branching Program. We organize our approach into the following three steps: **(1)** first, we recall the encoding scheme proposed in the SIM-secure ABE-BP of [GV15]; **(2)** Then, we present our flexibly bi-deniable attribute bi-translucent set (AB-BTS) scheme, as was defined in Definition 3.2. Our AB-BTS construction uses the ideas of Gorbunov and Vinayagamurthy [GV15], with essential modifications that allow us to tightly upper and lower bound evaluated noise terms. As discussed in the Introduction, this tighter analysis plays a key role in proving bi-deniability. **(3)** Finally, we show how to obtain the desired bi-deniable ABE scheme from our AB-BTS. As pointed out by Canetti et al. [CDNO97] and O'Neill et al. [OPW11], a bitranslucent set scheme implies flexibly bi-deniable PKE. We observe that the same idea generalizes to the case of an AB-BTS scheme and flexibly bi-deniable ABE in a straightforward manner.

4.1 Encoding Schemes for Branching Programs

Basic Homomorphic Encoding. Before proceeding to the public key evaluation algorithm, we first described basic homomorphic addition and multiplication over public keys and encoded ciphertexts based on the techniques in [GSW13, AP14, BGG$^+$14].

Definition 4.1 (LWE Encoding)**.** *For any matrix $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}$, we define an LWE encoding of a bit $a \in \{0,1\}$ with respect to a public key $\mathbf{A}$ and randomness $\boldsymbol{s} \leftarrow \mathbb{Z}_q^n$ as*

$$\psi_{\mathbf{A},\boldsymbol{s},a} = \boldsymbol{s}^T(\mathbf{A} + a \cdot \mathbf{G}) + \boldsymbol{e} \in \mathbb{Z}_q^m$$

for error vector $\boldsymbol{e} \leftarrow \chi^m$ and the gadget matrix $\mathbf{G} \in \mathbb{Z}_q^{n\times m}$.

In our construction, all LWE encodings will be encoded using the same LWE secret $\boldsymbol{s}$, thus for simplicity, we will simply refer to such an encoding as $\psi_{\mathbf{A},a}$.

For homomorphic addition, the addition algorithm takes as input two encodings $\psi_{\mathbf{A},a}, \psi_{\mathbf{A}',a}$, and outputs the sum of them. Let $\mathbf{A}^+ = \mathbf{A} + \mathbf{A}'$ and $a^+ = a + a'$

$$\mathsf{Add}(\psi_{\mathbf{A},a}, \psi_{\mathbf{A}',a'}) = \psi_{\mathbf{A},a} + \psi_{\mathbf{A}',a'} = \psi_{\mathbf{A}^+,a^+}$$

For homomorphic multiplication, the multiplication algorithm takes as input two encodings $\psi_{\mathbf{A},a}, \psi_{\mathbf{A}',a}$, and outputs an encoding $\psi_{\mathbf{A}^\times,a^\times}$, where $\mathbf{A}^\times = -\mathbf{A}\mathbf{G}^{-1}(\mathbf{A}')$ and $a^\times = aa'$.

$$\mathsf{Mult}(\psi_{\mathbf{A},a}, \psi_{\mathbf{A}',a'}) = -\psi \cdot \mathbf{G}^{-1}(\mathbf{A}') + a \cdot \psi' = \psi_{\mathbf{A}^\times,a^\times}$$

Public Key Evaluation Algorithm. Following the notation in [GV15], we define a public evaluation algorithm $\mathsf{Eval}_{\mathsf{pk}}$. The algorithm takes as input a description of the branching program BP, a collection of public keys $\{\mathbf{A}_i\}_{i\in[\ell]}$ (one for each attribute bit x_i), a collection of public keys $\mathbf{V}_{0,i}$ for initial state vector and an auxiliary matrix $\mathbf{A}^c$, and outputs an evaluated public key corresponding to the branching program BP.

$$\mathbf{V}_{\mathsf{BP}} \leftarrow \mathsf{Eval}_{\mathsf{pk}}(\mathsf{BP}, \{\mathbf{A}_i\}_{i\in[\ell]}, \{\mathbf{V}_{0,i}\}_{i\in[5]}, \mathbf{A}^c)$$

where the auxiliary matrix $\mathbf{A}^c$ are used to encoded constant 1 for each input wire. We also define matrix $\mathbf{A}'_i = \mathbf{A}^c - \mathbf{A}_i$ as a public key used to encode $1 - x_i$. By the definition of branching programs, the output $\mathbf{V}_{\mathsf{BP}} \in \mathbb{Z}_q^{n\times m}$ is the homomorphically generated public key $\mathbf{V}_{L,1}$ at position 1 of the state vector for the L-th step of the branching program evaluation.

Recall that in the definition of branching programs, BP is represented by the tuple $\{\mathsf{var}(t), \{\gamma_{t,i,0}, \gamma_{t,i,1}\}_{i\in[5]}\}$ for $t \in [L]$, and the initial state vector is set to be $\boldsymbol{v}_0 = (1, 0, 0, 0, 0)$. Further, for $t \in [L]$, the computation is performed as $\boldsymbol{v}_t[i] = \boldsymbol{v}_{t-1}[\gamma_{t,i,0}](1 - x_{\mathsf{var}(t)}) + \boldsymbol{v}_{t-1}[\gamma_{t,i,1}] \cdot x_{\mathsf{var}(t)}$. It is important for the security proof (among other reasons) that the evaluated state vector in each step is independent of the attribute vector.

Encoding Evaluation Algorithm. We define an encoding evaluation algorithm $\mathsf{Eval}_{\mathsf{ct}}$ that takes as input the description of a branching program BP, an attribute vector $\boldsymbol{x}$, a set of encodings for the attribute $\{\mathbf{A}_i, \psi_i := \psi_{\mathbf{A}_i,x_i}\}_{i\in[\ell]}$, encodings of the initial state vector $\{\mathbf{V}_{0,i}, \psi_{0,i} := \psi_{\mathbf{V}_{0,i}\boldsymbol{v}_0[i]}\}_{i\in[5]}$ and an encoding of a constant 1, i.e. $\psi^c := \psi_{\mathbf{A}^c,1}$. The algorithm $\mathsf{Eval}_{\mathsf{ct}}$ outputs an encoding of the result $y := \mathsf{BP}(\boldsymbol{x})$ with respect to the homomorphically derived public key $\mathbf{V}_{\mathsf{BP}} := \mathbf{V}_{L,1}$

$$\psi_{\mathsf{BP}} \leftarrow \mathsf{Eval}_{\mathsf{ct}}(\mathsf{BP}, \boldsymbol{x}, \{\mathbf{A}_i, \psi_i\}_{i\in[\ell]}, \{\mathbf{V}_{0,i}, \psi_{0,i}\}_{i\in[5]}, \{\mathbf{A}^c, \psi^c\})$$

As mentioned above, in branching program computation, for $t \in [L]$, we have for all $i \in [5]$

$$\boldsymbol{v}_t[i] = \boldsymbol{v}_{t-1}[\gamma_{t,i,0}](1 - x_{\mathsf{var}(t)}) + \boldsymbol{v}_{t-1}[\gamma_{t,i,1}] \cdot x_{\mathsf{var}(t)}$$

The evaluation algorithm proceeds inductively to update the encoding of the state vector for each step of the branching program. Next, we need to instantiate this inductive computation using the homomorphic operations described above, i.e. $\mathsf{Add}, \mathsf{Mult}$. Following the notation used in [GV15], we define $\psi'_i := \psi_{\mathbf{A}'_i,(1-x_i)} = \boldsymbol{s}^T(\mathbf{A}'_i + (1-x_i)\mathbf{G}) + \boldsymbol{e}'_i$, where $\mathbf{A}'_i = \mathbf{A}^c - \mathbf{A}_i$, to denote the encoding of $1-x_i$. This encoding can be computed using $\mathsf{Add}(\psi_{\mathbf{A}^c_i,1}, -\psi_{\mathbf{A}_i,x_i})$. Then assuming at time $t-1 \in [L]$ we hold encodings of the state vector $\{\psi_{\mathbf{V}_{t-1},i,\boldsymbol{v}_{t-1}[i]}\}_{i\in[5]}$. For $i \in [5]$, we compute the encodings of new state values as

$$\psi_{i,t} = \mathsf{Add}(\mathsf{Mult}(\psi'_{\mathsf{var}(t)}, \psi_{t-1,\gamma_0}), \mathsf{Mult}(\psi_{\mathsf{var}(t)}, \psi_{t-1,\gamma_1}))$$

where $\gamma_0 := \gamma_{t,i,0}$ and $\gamma_1 := \gamma_{t,i,1}$. We omit the correctness proof of the encoding here, which is presented in [GV15].

Simulated Public Key Evaluation Algorithm. The simulation strategy was first developed in [BGG+14], and then adapted by Gorbunov and Vinayagamurthy [GV15] in branching program scenario. In particular, set $\mathbf{A}_i = \mathbf{A}_i\mathbf{R}_i - x_i\mathbf{G}$ for some shared public key matrix $\mathbf{A}$ and low norm matrix $\mathbf{R}_i$. Similarly, the state public keys $\mathbf{A}_{t,i} = \mathbf{A}\mathbf{R}_{t,i} - \boldsymbol{v}_t[i]\mathbf{G}$, and matrices $\mathbf{A}^c = \mathbf{A}\mathbf{R}^c - \mathbf{G}$. The evaluation algorithm $\mathsf{Eval}_{\mathsf{Sim}}$ takes

as input the description of branching program BP, the attribute vector $\boldsymbol{x}$, collections of low norm matrices $\{\mathbf{R}_i\}_{i\in[\ell]}, \{\mathbf{R}_{0,i}\}_{i\in[5]}, \mathbf{R}^c$ corresponding to input public key, initial state vector and complement matrices respectively, and a shared matrix $\mathbf{A}$. It outputs a homomorphically derived low norm matrix $\mathbf{R}_{\mathsf{BP}}$:

$$\mathbf{R}_{\mathsf{BP}} \leftarrow \mathsf{Eval}_{\mathsf{Sim}}(\mathsf{BP}, \boldsymbol{x}, \{\mathbf{R}_i\}_{i\in[\ell]}, \{\mathbf{R}_{0,i}\}_{i\in[5]}, \mathbf{R}^c, \mathbf{A})$$

In particular, let $\mathbf{R}'_i = \mathbf{R}^c_i - \mathbf{R}_i$ for $i \in [\ell]$. We derive the low-norm matrices $\mathbf{R}_{t,i}$ for $i \in [5]$ as

1. Let $\gamma_0 := \gamma_{t,i,0}$ and $\gamma_1 := \gamma_{t,i,1}$.
2. Compute

$$\begin{aligned}\mathbf{R}_{t,i} = &(-\mathbf{R}'_{\mathsf{var}(t)}\mathbf{G}^{-1}(\mathbf{V}_{t-1,\gamma_0}) + (1 - x_{\mathsf{var}(t)}) \cdot \mathbf{R}_{t-1,\gamma_0}) \\ &+ (-\mathbf{R}_{\mathsf{var}(t)}\mathbf{G}^{-1}(\mathbf{V}_{t-1,\gamma_1}) + x_{\mathsf{var}(t)} \cdot \mathbf{R}_{t-1,\gamma_1})\end{aligned} \tag{1}$$

We let $\mathbf{R}_{L,1}$ be the matrix obtained at L-th step corresponding to state value 1 by the above algorithm. The correctness requires the norm of $\mathbf{R}_{\mathsf{BP}}$ remains small and the matrix $\mathbf{V}_{\mathsf{BP}}$ output by $\mathsf{Eval}_{\mathsf{pk}}$ satisfies $\mathbf{V}_{\mathsf{BP}} = \mathbf{A}\mathbf{R}_{\mathsf{BP}} - \mathsf{BP}(\boldsymbol{x})\mathbf{G}$. We refer to the counterpart in [GV15] for the detailed correctness proof.

In order to achieve correctness and deniability, it is important for us to both lower and upper bound the norm of $||\mathbf{R}_{\mathsf{BP}}||$. Here we apply the triangular inequality of the norm and obtain the following lemma:

Lemma 4.2. *Let $\mathbf{R}_{i,j}$'s be the matrices defined as above. Then for every $t \in [\ell], i \in [5]$ and every error vector $\boldsymbol{e} \in \mathbb{Z}_q^m$, we have $||\boldsymbol{e}^T \cdot \mathbf{R}_{t-1,j}|| - \Theta(m^{1.5}) \cdot ||\boldsymbol{e}|| \leq ||\boldsymbol{e}^T \cdot \mathbf{R}_{t,i}|| \leq ||\boldsymbol{e}|| \cdot ||\mathbf{R}_{t-1,j}|| + \Theta(m^{1.5}) \cdot ||\boldsymbol{e}||$, where $j = \gamma_{x_{\mathsf{var}(t)}}$.*

Proof. Recall the matrix $\mathbf{R}_{i,j}$ is computed as

$$\mathbf{R}_{t,i} = (-\mathbf{R}'_{\mathsf{var}(t)}\mathbf{G}^{-1}(\mathbf{V}_{t-1,\gamma_0}) + (1 - x_{\mathsf{var}(t)}) \cdot \mathbf{R}_{t-1,\gamma_0}) + (-\mathbf{R}_{\mathsf{var}(t)}\mathbf{G}^{-1}(\mathbf{V}_{t-1,\gamma_1}) + x_{\mathsf{var}(t)} \cdot \mathbf{R}_{t-1,\gamma_1})$$

where $x_{\mathsf{var}(t)} \in \{0, 1\}$. Without loss of generality, we assume $x_{\mathsf{var}(t)} = 1$, thus we obtain

$$\mathbf{R}_{t,i} = -\mathbf{R}'_{\mathsf{var}(t)}\mathbf{G}^{-1}(\mathbf{V}_{t-1,\gamma_0}) + \mathbf{R}_{t-1,\gamma_1} - \mathbf{R}_{\mathsf{var}(t)}\mathbf{G}^{-1}(\mathbf{V}_{t-1,\gamma_1})$$

Since $\mathbf{G}^{-1}(\mathbf{V}_{t-1,\gamma_1}) \in \{0, 1\}^{m\times m}$, we know $||\mathbf{G}^{-1}(\mathbf{V}_{t-1,\gamma_1})|| \leq m$. Since matrices $\mathbf{R}_{\mathsf{var}(t)}, \mathbf{R}'_{\mathsf{var}(t)}$ were chosen uniformly at random in $\in \{-1, 1\}^{m\times m}$, we know that their norm is bounded by $\Theta(\sqrt{m})$ with high probability by Lemma 2.3. Therefore, we can bound the norm of term $||\mathbf{R}'_{\mathsf{var}(t)}\mathbf{G}^{-1}(\mathbf{V}_{t-1,\gamma_0}) + \mathbf{R}_{\mathsf{var}(t)}\mathbf{G}^{-1}(\mathbf{V}_{t-1,\gamma_1})|| \leq \Theta(m^{1.5})$. By applying the triangular inequality, it holds for every $t \in [\ell], i \in [5]$ and vector $\boldsymbol{e} \in \mathbb{Z}_q^m$,

$$||\boldsymbol{e}^T \cdot \mathbf{R}_{t-1,j}|| - \Theta(m^{1.5})||\boldsymbol{e}|| \leq ||\boldsymbol{e}^T \cdot \mathbf{R}_{t,i}|| \leq ||\boldsymbol{e}|| \cdot ||\mathbf{R}_{t-1,j}|| + \Theta(m^{1.5})||\boldsymbol{e}||$$

where $j = \gamma_{x_{\mathsf{var}(t)}}$. □

By applying the above lemma inductively on the equation (1) of computing matrix $\mathbf{R}_{\mathsf{BP}}$ for input length ℓ, we can obtain the following theorem:

Theorem 4.3. *Let BP be a length ℓ branching program, and $\mathbf{R}_{\mathsf{BP}}$ be the matrix as defined above. Then we have $||\boldsymbol{e}^T \cdot \mathbf{R}_{0,j}|| - 2m^{1.5}\ell||\boldsymbol{e}|| \leq ||\boldsymbol{e}^T\mathbf{R}_{\mathsf{BP}}|| \leq ||\boldsymbol{e}^T|| \cdot ||\mathbf{R}_{0,j}|| + 2m^{1.5}\ell||\boldsymbol{e}||$ for some $j \in [5]$.*

Proof. Applying Lemma 4.2 inductively on input length ℓ, we have

$$||\boldsymbol{e}^T\mathbf{R}_{\mathsf{BP}}|| \geq ||\boldsymbol{e}^T \cdot \mathbf{R}_{\ell-1,j}|| - 2m^{1.5}||\boldsymbol{e}|| \geq \cdots \geq ||\boldsymbol{e}^T \cdot \mathbf{R}_{0,j}|| - 2m^{1.5}\ell||\boldsymbol{e}||$$

We can obtain the upper bound of $||\boldsymbol{e}^T\mathbf{R}_{\mathsf{BP}}||$ using similar computation. □

Lemma 4.4. *Let $\mathbf{R}$ is an $m \times m$ be a matrix chosen at random from $\{-1,1\}^{m\times m}$, and $\boldsymbol{u} = (u_1, ..., u_m) \in \mathbb{R}^m$ be a vector chosen according to the m dimensional Gaussian with width α. Then we have*

$$\Pr\left[||\boldsymbol{u}^T\mathbf{R}||^2 \in \Theta(m^2\alpha^2)\right] > 1 - \mathsf{negl}(m).$$

Proof. We know with overwhelming probability over the choice of $\boldsymbol{u}$, all of its entries have absolute value less than $B = \alpha\omega(\log m)$. Also, we know that with overwhelming probability, we have $||\boldsymbol{u}||^2 = \Theta(m\alpha^2)$. We call a sample typical if it satisfies these two conditions. Note that it is without loss of generality to just consider the typical samples, from a simple union bound argument.

Then we consider a fixed typical choice of vector $\boldsymbol{u} = (u_1, ..., u_m) \in \mathbb{R}^m$. We write the inner product of $\boldsymbol{u}^T \cdot \boldsymbol{r}$ where $\boldsymbol{r} = (r_1, \ldots, r_m)$ is sampled uniformly from $\{-1,1\}^m$. We observe that $\mathbb{E}\left[||\boldsymbol{u}^T \cdot \boldsymbol{r}||^2\right] = \mathbb{E}\left[\sum_{i=1}^m r_i^2 u_i^2 + \sum_{i<j\leq m} r_i r_j u_i u_j\right] = \sum_{i=1}^m u_i^2 = ||\boldsymbol{u}||^2$. This is because each r_i, r_j are independent and have mean 0.

Now, for such a fixed $\boldsymbol{u}$ we denote random variables $X_1, \ldots, X_m$ be i.i.d. samples of $\boldsymbol{r}^T\boldsymbol{u}$. It is not hard to see that

- $||\boldsymbol{u}^T\mathbf{R}||^2 = X_1^2 + X_2^2 + \cdots + X_m^2$, (one can view X_i as the i-th entry of $\boldsymbol{u}^T\mathbf{R}$),
- $\mathbb{E}\left[||\boldsymbol{u}^T\mathbf{R}||^2\right] = m||\boldsymbol{u}||^2$.

Next we claim that for each i, $X_i^2 \leq mB^2\omega(\log m)$ with overwhelming probability. By Hoeffding's inequality, we have

$$\Pr\left[\left|\sum_{j\in[m]} r_j u_j\right| > t\right] < 2e^{-\frac{2t^2}{m\cdot 4B^2}}.$$

This is because each $r_j u_j \in [-B, B]$. (Recall that we consider a fixed $\boldsymbol{u}$ for the typical case). By setting $t = \sqrt{m}B\omega(\log m)$, we have $\Pr[|X_i| > t] < \mathsf{negl}(m)$. Thus $X_i^2 \leq mB^2\omega(\log m)$ with overwhelming probability. So we can consider truncated versions of X_i^2's, where we cut out the large samples. This will only induce a negligible statistical distance, and change the expectation by a negligible amount. For simplicity of presentation, we still use the notation X_i^2's in the following arguments, but the reader should keep in mind that they were truncated.

Next again we apply Hoeffding's inequality to the X_i^2's to obtain

$$\Pr\left[\left|||\boldsymbol{u}^T\mathbf{R}||^2 - m||\boldsymbol{u}||^2\right| > t'\right] < 2e^{-\frac{2t'^2}{\sum_{i=1}^m (mB^2\omega(\log m))^2}} = 2e^{-\frac{2t'^2}{m^3B^4\omega(\log m)}}.$$

By taking $t' = m||\boldsymbol{u}||^2/2$, we have

$$\Pr\left[\left|||\boldsymbol{u}^T\mathbf{R}||^2 - m||\boldsymbol{u}||^2\right| > t'\right] < 2e^{-\frac{||\boldsymbol{u}||^4}{2mB^4\omega(\log m)}}.$$

Since $\boldsymbol{u}$ is typical, we know that $||\boldsymbol{u}||^2 = \Theta(m\alpha^2)$. Also recall that $B = \alpha\omega(\log m)$. So we have

$$\Pr\left[||\boldsymbol{u}^T\mathbf{R}||^2 \in \Theta(m^2\alpha^2)\right] > 1 - 2e^{-\frac{m}{\omega(\log m)}} = 1 - \mathsf{negl}(m).$$

This completes the proof. □

4.2 Construction of Flexibly Bi-Deniable ABE for Branching Programs

In this part, we present our flexibly bi-deniable AB-BTS scheme for bounded-length Branching Programs. We use a semantically-secure public key encryption $\Pi = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ with message space $\mathcal{M}_\Pi = \mathbb{Z}_q^{m\times m}$ and ciphertext space $\mathcal{C}_\Pi$. For a family of branching programs of length bounded by L and input space $\{0,1\}^\ell$, the description of BiDenAB-BTS = (Setup, DenSetup, Keygen, SampleP, SampleU, TestP, FakeRCoins, FakeSCoins) are as follows:

- $\mathsf{Setup}(1^\lambda, 1^L, 1^\ell)$: On input the security parameter λ, the length of the branching program L and length of the attribute vector ℓ,

  1. Set the LWE dimension be $n = n(\lambda)$, modulus $q = q(n, L)$. Choose Gaussian distribution parameter $s = s(n)$. Let $\mathsf{params} = (n, q, m, s)$.
  2. Sample one random matrix associated with its trapdoor as
  $$(\mathbf{A}, \mathbf{T}_\mathbf{A}) \leftarrow \mathsf{TrapGen}(q, n, m)$$
  3. Choose $\ell + 6$ random matrices $\{\mathbf{A}_i\}_{i\in[\ell]}, \{\mathbf{V}_{0,i}\}_{i\in[5]}, \mathbf{A}^c$ from $\mathbb{Z}_q^{n\times m}$.
  4. Choose a random vector $\boldsymbol{u} \in \mathbb{Z}_q^n$.
  5. Compute a public/secret key pair $(\mathsf{pk}', \mathsf{sk}')$ for a semantically secure public key encryption $(\mathsf{pk}', \mathsf{sk}') \leftarrow \mathsf{Gen}'(1^\lambda)$
  6. Output the public parameter pp and master secret key msk as
  $$\mathsf{pp} = (\mathsf{params}, \mathbf{A}, \{\mathbf{A}_i\}_{i\in[\ell]}, \{\mathbf{V}_{0,i}\}_{i\in[5]}, \mathbf{A}^c, \boldsymbol{u}, \mathsf{pk}'), \quad \mathsf{msk} = (\mathbf{T}_\mathbf{A}, \mathsf{sk}')$$

- $\mathsf{DenSetup}(1^\lambda, 1^L, 1^\ell)$: On input the security parameter λ, the length of branching program L and length of attribute vector ℓ, the deniable setup algorithm runs the same computation as setup algorithm, and outputs
$$\mathsf{pp} = (\mathsf{params}, \mathbf{A}, \{\mathbf{A}_i\}_{i\in[\ell]}, \{\mathbf{V}_{0,i}\}_{i\in[5]}, \mathbf{A}^c, \boldsymbol{u}, \mathsf{pk}'), \quad \mathsf{msk} = (\mathbf{T}_\mathbf{A}, \mathsf{sk}') \quad \mathsf{fk} = (\mathbf{T}_\mathbf{A}, \mathsf{sk}')$$

- $\mathsf{Keygen}(\mathsf{msk}, \mathsf{BP})$: On input the master secret key msk and the description of a branching program BP, $\mathsf{BP} = (\boldsymbol{v}_0, \{\mathsf{var}(t), \{\gamma_{t,i,0}, \gamma_{t,i,1}\}_{i\in[5]}\}_{t\in[L]})$.

  1. Homomorphically compute a public matrix with respect to the branching program BP: $\mathbf{V}_\mathsf{BP} \leftarrow \mathsf{Eval}_\mathsf{pk}(\mathsf{BP}, \{\mathbf{A}_i\}_{i\in[\ell]}, \{\mathbf{V}_{0,i}\}_{i\in[5]}, \mathbf{A}^c)$.
  2. Sample a low norm vector $\boldsymbol{r}_\mathsf{BP} \in \mathbb{Z}_q^{2m}$, using
  $$\boldsymbol{r}_\mathsf{BP} \leftarrow \mathsf{SampleLeft}(\mathbf{A}, \mathbf{T}_\mathbf{A}, (\mathbf{V}_\mathsf{BP} + \mathbf{G}), \boldsymbol{u}, sq)$$
  such that $\boldsymbol{r}_\mathsf{BP}^T \cdot [\mathbf{A}|\mathbf{V}_\mathsf{BP} + \mathbf{G}] = \boldsymbol{u}$.
  3. Output the secret key sk_BP for branching program as $\mathsf{sk}_\mathsf{BP} = (\boldsymbol{r}_\mathsf{BP}, \mathsf{BP})$.

- $\mathsf{SampleP}(\mathsf{pp}, \boldsymbol{x})$: On input public parameters pp and attribute $\boldsymbol{x}$,

  1. Choose an LWE secret $\boldsymbol{s} \in \mathbb{Z}_q^n$ uniformly at random.
  2. Choose noise vector $\boldsymbol{e} \leftarrow \mathcal{D}_{\mathbb{Z}_q^m,\alpha}$, and compute $\psi_0 = \boldsymbol{s}^T\mathbf{A} + \boldsymbol{e}$.
  3. Choose one random matrices $\mathbf{R}^c \leftarrow \{-1, 1\}^{m\times m}$, and let $\boldsymbol{e}^c = \boldsymbol{e}^T\mathbf{R}^c$. Compute an encoding of constant 1: $\psi^c = \boldsymbol{s}^T(\mathbf{A}^c + \mathbf{G}) + \boldsymbol{e}^c$.

4. Encode each bit $i \in [\ell]$ of the attribute vector:
   (a) Choose a random matrix $\mathbf{R}_i \leftarrow \{-1,1\}^{m\times m}$, and let $\boldsymbol{e}_i = \boldsymbol{e}^T\mathbf{R}_i$.
   (b) Compute $\psi_i = \boldsymbol{s}^T(\mathbf{A}_i + x_i\mathbf{G}) + \boldsymbol{e}_i$.
5. Encode the initial state vector $\boldsymbol{v}_0 = (1,0,0,0,0)$, for $i \in [5]$
   (a) Choose a random matrix $\mathbf{R}'_{0,i} \leftarrow \{-1,1\}^{m\times m}$, and let $\mathbf{R}_{0,i} = \eta\mathbf{R}'_{0,i}, \boldsymbol{e}_{0,i} = \boldsymbol{e}^T\mathbf{R}_{0,i}$, where the noise scaling parameter η is set in Section 4.3.
   (b) Compute $\psi_{0,i} = \boldsymbol{s}^T(\mathbf{A}_i + \boldsymbol{v}_0[i]\mathbf{G}) + \boldsymbol{e}_{0,i}$.
6. Compute $c = \boldsymbol{s}^T\boldsymbol{u} + e$, where $e \leftarrow \mathcal{D}_{\mathbb{Z}_q,s}$
7. Use PKE to encrypt randomly chosen matrices $\mathbf{R}^c, \{\mathbf{R}_i\}_{i\in[\ell]}$ and $\{\mathbf{R}_{0,i}\}_{i\in[5]}$:

$$\mathbf{T}_i \leftarrow \mathsf{Enc}'(\mathsf{pk}', \mathbf{R}_i), \mathbf{T}^c \leftarrow \mathsf{Enc}'(\mathsf{pk}', \mathbf{R}^c), \mathbf{T}_{0,i} \leftarrow \mathsf{Enc}'(\mathsf{pk}', \mathbf{R}_{0,i})$$

8. Output the ciphertext

$$\mathsf{ct}_{\boldsymbol{x}} = (\boldsymbol{x}, \psi_0, \{\psi_i\}_{i\in[\ell]}, \psi^c, \{\psi_{0,i}\}_{i\in[5]}, c, \{\mathbf{T}_i\}_{i\in[\ell]}, \mathbf{T}^c, \{\mathbf{T}_{0,i}\}_{i\in[5]})$$

- $\mathsf{SampleU}(\mathsf{pp}, \boldsymbol{x})$: Output a uniformly random vector $\mathsf{ct} \in \mathbb{Z}_q^m \times \mathbb{Z}_q^{\ell m} \times \mathbb{Z}_q^{\ell m} \times \mathbb{Z}_q^{5m} \times \mathbb{Z}_q \times \mathcal{C}_\Pi^\ell \times \mathcal{C}_\Pi \times \mathcal{C}_\Pi^5$.
- $\mathsf{TestP}(\mathsf{sk}_{\mathsf{BP}}, \mathsf{ct}_{\boldsymbol{x}})$: On input the secret key $\mathsf{sk}_{\mathsf{BP}}$ for a branching program BP and a ciphertext associated with attribute $\boldsymbol{x}$, if $\mathsf{BP}(\boldsymbol{x}) = 0$, output $\perp$, otherwise,
  1. Homomorphically compute the evaluated ciphertext of result $\mathsf{BP}(\boldsymbol{x})$

$$\psi_{\mathsf{BP}} \leftarrow \mathsf{Eval}_{\mathsf{ct}}(\mathsf{BP}, \boldsymbol{x}, \{\mathbf{A}_i, \psi_i\}_{i\in[\ell]}, \{\mathbf{V}_{0,i}, \psi_{0,i}\}_{i\in[5]}, \{\mathbf{A}_i^c, \psi_i^c\}_{i\in[\ell]})$$

  2. Then compute $\phi = [\psi_0|\psi_{\mathsf{BP}}]^T \cdot \boldsymbol{r}_{\mathsf{BP}}$. Accept $\mathsf{ct}_{\boldsymbol{x}}$ as a P-sample if $|c - \phi| < 1/4$, otherwise reject.
- $\mathsf{FakeSCoins}(r_S)$: Simply output the P-sample $\mathbf{c}$ as the randomness r_S^* that would cause $\mathsf{SampleU}$ to output $\boldsymbol{c}_{\boldsymbol{x}}$.
- $\mathsf{FakeRCoins}(\mathsf{pp}, \mathsf{fk}, \mathsf{ct}_{\boldsymbol{x}}, \mathsf{BP})$: On input the public parameters pp, the faking key fk, a ciphertext $\mathsf{ct}_{\boldsymbol{x}}$ and description of a branching program BP
  1. If $\mathsf{BP}(\boldsymbol{x}) \neq 0$, then output $\mathsf{sk}_f \leftarrow \mathsf{Keygen}(\mathsf{fk}, \mathsf{BP})$.
  2. Otherwise, parse ciphertext $\mathsf{ct}_{\boldsymbol{x}}$ as

$$\mathsf{ct}_{\boldsymbol{x}} = (\boldsymbol{x}, \psi_0, \{\psi_i\}_{i\in[\ell]}, \psi^c, \{\psi_{0,i}\}_{i\in[5]}, c, \{\mathbf{T}_i\}_{i\in[\ell]}, \mathbf{T}^c, \{\mathbf{T}_{0,i}\}_{i\in[5]})$$

  Compute $\boldsymbol{e} \leftarrow \mathsf{Invert}(\mathbf{A}, \mathbf{T}_{\mathbf{A}}, \psi_0)$. Then decrypt $(\{\mathbf{T}_i\}_{i\in[\ell]}, \mathbf{T}^c, \{\mathbf{T}_{0,i}\}_{i\in[5]})$ respectively using $\mathsf{Dec}(\mathsf{sk}', \cdot)$ to obtain $\{\mathbf{R}_i\}_{i\in[\ell]}, \mathbf{R}^c, \{\mathbf{R}_{0,i}\}_{i\in[5]}$. Compute evaluated error

$$\boldsymbol{e}_{\mathsf{BP}} \leftarrow \mathsf{Eval}_{\mathsf{ct}}(\mathsf{BP}, \boldsymbol{x}, \{\mathbf{A}_i, \boldsymbol{e}^T\mathbf{R}_i\}_{i\in[\ell]}, \{\mathbf{V}_{0,i}, \boldsymbol{e}^T\mathbf{R}_{0,i}\}_{i\in[5]}, \{\mathbf{A}^c, \boldsymbol{e}^T\mathbf{R}^c\})$$

  such that $\boldsymbol{e}_{\mathsf{BP}} = \boldsymbol{e}^T\mathbf{R}_{\mathsf{BP}}$.
  3. Homomorphically compute a public matrix with respect to the branching program BP: $\mathbf{V}_{\mathsf{BP}} \leftarrow \mathsf{Eval}_{\mathsf{pk}}(\mathsf{BP}, \{\mathbf{A}_i\}_{i\in[\ell]}, \{\mathbf{V}_{0,i}\}_{i\in[5]}, \{\mathbf{A}_i^c\}_{i\in[\ell]})$. Then sample a properly distributed secret key $\boldsymbol{r}_{\mathsf{BP}} \in \mathbb{Z}_q^{2m}$, using

$$\boldsymbol{r}_{\mathsf{BP}} \leftarrow \mathsf{SampleLeft}(\mathbf{A}, \mathbf{T}_{\mathbf{A}}, (\mathbf{V}_{\mathsf{BP}} + \mathbf{G}), \boldsymbol{u}, s)$$

4. Sample correlation vector $\mathbf{y}_0 \leftarrow \mathcal{D}_{\mathbb{Z}_q^m, \beta^2 q^2 \mathbf{I}_{m\times m}}$. Then sample correlation coefficient $\mu \leftarrow \mathcal{D}_\gamma$, and set vector $\boldsymbol{y}_1 = (\mu \boldsymbol{e}_{\mathsf{BP}} + \mathcal{D}_{\mathbf{Z}^m, \mathbf{Q}})q$, where

$$\mathbf{Q} = \beta^2 \mathbf{I}_{m\times m} - \gamma^2 \alpha^2 \mathbf{R}_{\mathsf{BP}}^T \mathbf{R}_{\mathsf{BP}} \tag{2}$$

5. Let $\boldsymbol{y} = (\boldsymbol{y}_0 | \boldsymbol{y}_1)$, then sample and output the faked secret key $\mathsf{sk}^*_{\mathsf{BP}} = \mathbf{r}^*_{\mathsf{BP}}$ as $\mathbf{r}^*_{\mathsf{BP}} \leftarrow \boldsymbol{y} + \mathcal{D}_{\Lambda + \boldsymbol{r}_{\mathsf{BP}} - \boldsymbol{y}, \sqrt{s^2 - \beta^2}}$, using $\mathsf{SampleD}(\mathsf{ExtBasis}(\mathbf{A}, \mathbf{T}_{\mathbf{A}}, \mathbf{V}_{\mathsf{BP}} + \mathbf{G}), \boldsymbol{r}_{\mathsf{BP}} - \boldsymbol{y}, \sqrt{s^2 - \beta^2})$, where $\Lambda = \Lambda^{\perp}([\mathbf{A}|\mathbf{V}_{\mathsf{BP}} + \mathbf{G}])$.

The SampleP algorithm is similar to the ABE ciphertexts in the work [GV15], except that we add another scaling factor η to the rotation matrices $\mathbf{R}_{0,i}$'s. This allows us to both upper and lower bound the noise growth, which is essential to achieve bi-deniability. As we discussed in the introduction, the FakeRCoins embeds the evaluated noise into the secret key, so that it will change the decrypted value of the targeted ciphertext, but not others. Next we present the theorem we achieve and a high level ideas of the proof. We describe the intuition of our proof as follows.

Overview of Our Security Proof. At a high level, our security proof begins at the Fake experiment (cf. Definition 3.1 for a formal description), where first a ciphertext ct^* and its associated noise terms $\boldsymbol{e}^*$ are sampled, then a fake key $\boldsymbol{r}^*$ is generated that "artificially" fails to decrypt any ciphertext with noise vector (oriented close to) $\boldsymbol{e}^*$. In the end, we will arrive at the Real experiment, where an honest key $\boldsymbol{r}$ is generated that "genuinely" fails to decrypt the honestly generated, coerced ciphertext ct^*. (Multi-ct coercion security follows by a standard hybrid argument that repeatedly modifies respective $\boldsymbol{r}^*$ to $\boldsymbol{r}$ for each coerced ct^* in order.) In order to transition from Fake to Real, we move through a sequence of computationally- or statistically-indistinguishable hybrid experiments.

The first set of intermediate experiments (represented by H_1 and H_2 in our formal proof) embeds the attribute $\boldsymbol{x}$ of the challenge ciphertext ct^* in the public parameters, in a similar fashion to the beginning of every SIM-secure proof of lattice-based ABE. Indistinguishability follows via the Leftover Hash Lemma [DRS04]. (Note that the additional hybrid in our proof is used to ensure that the random rotation matrices $\mathbf{R}$ employed by the LHL for public key embedding of $\boldsymbol{x}$ are the *exact same* matrices $\mathbf{R}$ as used to generate the noise terms of the coerced ct^*, and uses the security of any semantically-secure PKE for computational indistinguishability.)

The next set of intermediate experiments (given by $\mathsf{H}_3, \mathsf{H}_4$, and H_5 in our formal proof) perform the "main, new work" of our security proof. Specifically, they "swap the order" of the generation of the pk matrices $\{\mathbf{A}\}$, the public coset $\boldsymbol{u}$ (in the public parameters and in the coerced ciphertext), and the error vector(s) $\boldsymbol{e}$ in the coerced ciphertext components. (An additional hybrid is used to toggle the order of a "correlation vector" $\boldsymbol{y}$ – a random, planted vector used to allow for a more modular analysis of these steps.) In each case, we give a *statistical* argument that the adversary's view in adjacent hybrids is indistinguishable or identical, using elementary properties of multi-dimensional Gaussians.

In the next step (given by H_6), we apply the eLWE^+ assumption to (roughly) change every component of the coerced ciphertext ct^* to uniform – except for the final c^* component used to blind the message μ.

In the final step (given by H_7), we transition to the Real experiment by changing the c^* component to uniform (in the presence of Dual Regev decryption under honest $\boldsymbol{z}$), using our sharper noise analysis as described above to show statistical indistinguishability of the final decryption output of $\boldsymbol{z}$ on ct^*.

Theorem 4.5. *Assuming the hardness of extended-*$\mathsf{LWE}_{q,\beta'}$*, the above algorithms form a secure attribute-based bitranslucent set schemem, as in Definition 3.2.*

Lemma 4.6. *For parameters set in Section 4.3, the AB-BTS defined above satisfies the correctness property in Definition 3.2.*

Proof. As we mentioned in Remark 3.3, the correctness of faking algorithms is implied by the bi-deniability property. Therefore, we only need to prove the correctness of normal decryption algorithm. For branching program BP and input $\boldsymbol{x}$, such that $\mathsf{BP}(\boldsymbol{x}) = 1$, we compute $\psi_{t,i}$ for $t \in [\ell]$ as

$$\begin{aligned}
\psi_{t,i} &= \mathsf{Add}(\mathsf{Mult}(\psi'_{\mathsf{var}(t)}, \psi_{t-1,\gamma_0}), \mathsf{Mult}(\psi_{\mathsf{var}(t)}, \psi_{t-1,\gamma_1})) \\
&= \mathsf{Add}\Big([\boldsymbol{s}^T(-\mathbf{A}'_{\mathsf{var}(t)}\mathbf{G}^{-1}(\mathbf{V}_{t-1,\gamma_0}) + (\boldsymbol{v}_t[\gamma_0] \cdot (1 - x_{\mathsf{var}(t)})) \cdot \mathbf{G}) + \boldsymbol{e}_1], \\
&\qquad ([\boldsymbol{s}^T(-\mathbf{A}'_{\mathsf{var}(t)}\mathbf{G}^{-1}(\mathbf{V}_{t-1,\gamma_1}) + (\boldsymbol{v}_t[\gamma_1] \cdot x_{\mathsf{var}(t)}) \cdot \mathbf{G}) + \boldsymbol{e}_2]\Big) \\
&= \boldsymbol{s}^T\Big[\underbrace{\big(-\mathbf{A}'_{\mathsf{var}(t)}\mathbf{G}^{-1}(\mathbf{V}_{t-1,\gamma_0}) - \mathbf{A}'_{\mathsf{var}(t)}\mathbf{G}^{-1}(\mathbf{V}_{t-1,\gamma_1})\big)}_{\mathbf{V}_{t,i}} \\
&\qquad + \underbrace{\big(\boldsymbol{v}_t[\gamma_0] \cdot (1 - x_{\mathsf{var}(t)}) + \boldsymbol{v}_t[\gamma_1] \cdot x_{\mathsf{var}(t)}\big)}_{\boldsymbol{v}_t[i]} \cdot \mathbf{G} \Big] + \boldsymbol{e}_{t,i}
\end{aligned}$$

At the end of the ciphertext evaluation, since $\mathsf{BP}(\boldsymbol{x}) = 1$, we can obtain $\psi_{\mathsf{BP}} = \boldsymbol{s}^T(\mathbf{V}_{\mathsf{BP}} + \mathbf{G}) + \boldsymbol{e}_{\mathsf{BP}}$, where $\boldsymbol{e}_{\mathsf{BP}} = \boldsymbol{e}^T\mathbf{R}_{\mathsf{BP}}$. Recall that the secret key $\mathsf{sk} = \boldsymbol{r}_{\mathsf{BP}}$ satisfying $[\mathbf{A}|\mathbf{V}_{\mathsf{BP}} + \mathbf{G}] \cdot \boldsymbol{r}_{\mathsf{BP}} = \boldsymbol{u}$. Then for $c - [\psi_0|\psi_{\mathsf{BP}}] \cdot \boldsymbol{r}_{\mathsf{BP}}$, it holds that

$$c - [\psi_0|\psi_{\mathsf{BP}}]^T \cdot \boldsymbol{r}_{\mathsf{BP}} = e - \boldsymbol{e}^T\mathbf{R}_{\mathsf{BP}} \cdot \boldsymbol{r}_{\mathsf{BP}}$$

Now we need to compute a bound for the final noise term. By applying Theorem 4.3, we obtain that

$$||\boldsymbol{e}^T|| \cdot ||\mathbf{R}_{\mathsf{BP}}|| + 2m^{1.5}\ell||e|| \leq (2m^{1.5}\ell + \eta\sqrt{m})||\boldsymbol{e}|| \leq \alpha\sqrt{m}(2m^{1.5}\ell + \eta\sqrt{m}) \cdot sq\sqrt{m} \leq \frac{1}{4}$$

So by setting the parameters appropriately, as in Section 4.3, we have that

$$|c - [\psi_0|\psi_{\mathsf{BP}}]^T \cdot \boldsymbol{r}_{\mathsf{BP}}| \leq 1/4$$

and the lemma follows. □

Lemma 4.7. *Assuming the hardness of extended-*$\mathsf{LWE}_{q,\beta'}$*, the AB-BTS scheme described above is bi-deniable as defined in Definition 3.2.*

Proof. First, we notice that because $\mathsf{SampleU}$ simply outputs its random coins as a uniformly random ct, we can use ct itself as the coins.

We prove the bi-deniability property by a sequence of hybrids H_i with details as follows:

Hybrid H_0: Hybrid H_0 is the same as the view of adversary $\mathcal{A}$ in the right-hand faking experiment in the definition of bi-deniability. We use the fact that algorithm Invert successfully recovers $\boldsymbol{e}$ from ct with overwhelming probability over all randomness in the experiment.

Hybrid H_1: In hybrid H_2, we switch the encryptions of matrices $(\{\mathbf{R}_i\}_{i\in[\ell]}, \{\mathbf{R}_{0,i}\}_{i\in[5]}, \mathbf{R}^c)$ in the ciphertext to encryptions of zero.

Recall that in hybrid H_0, we encrypt the randomness matrices $(\{\mathbf{R}_i\}_{i\in[\ell]}, \{\mathbf{R}_{0,i}\}_{i\in[5]}, \mathbf{R}^c)$ using semantically secure PKE Π, i.e.

$$\mathbf{T}_i \leftarrow \mathsf{Enc}'(\mathsf{pk}', \mathbf{R}_i), \quad \mathbf{T}^c \leftarrow \mathsf{Enc}'(\mathsf{pk}', \mathbf{R}^c), \quad \mathbf{T}_{0,i} \leftarrow \mathsf{Enc}'(\mathsf{pk}', \mathbf{R}_{0,i})$$

In hybrid H_1, we just set

$$\mathbf{T}_i \leftarrow \mathsf{Enc}'(\mathsf{pk}', \mathbf{0}), \quad \mathbf{T}^c \leftarrow \mathsf{Enc}'(\mathsf{pk}', \mathbf{0}), \quad \mathbf{T}_{0,i} \leftarrow \mathsf{Enc}'(\mathsf{pk}', \mathbf{0})$$

to be encryptions of $\mathbf{0} \in \mathbb{Z}^{m\times m}$ to replace encryptions of matrices $(\{\mathbf{R}_i\}_{i\in[\ell]}, \{\mathbf{R}_{0,i}\}_{i\in[5]}, \mathbf{R}^c)$.

Hybrid H_2: In hybrid H_2, we embed random matrices $(\{\mathbf{R}_i\}_{i\in[\ell]}, \{\mathbf{R}_{0,i}\}_{i\in[5]}, \mathbf{R}^c)$ and challenge attribute $\boldsymbol{x}^*$ in the public parameters pp.

Recall that in hybrid H_1 the matrices $(\{\mathbf{A}_i\}_{i\in[\ell]}, \{\mathbf{V}_{0,i}\}_{i\in[5]}, \mathbf{A}^c)$ are sampled at random. In hybrid H_2, we slightly change how these matrices are generated. Let $\boldsymbol{x}^* = (x_1^*, ..., x_\ell^*)$ be the challenge attribute that the adversary $\mathcal{A}$ intends to attack. We sample matrices $(\{\mathbf{R}_i\}_{i\in[\ell]}, \{\mathbf{R}'_{0,i}\}_{i\in[5]}, \mathbf{R}^c)$ uniformly random from $\{-1,1\}^{m\times m}$ and set $\mathbf{R}_{0,i} = \eta\mathbf{R}'_{0,i}$, which would be used both in the generation of public parameters and challenge ciphertext. We set $(\{\mathbf{A}_i\}_{i\in[\ell]}, \{\mathbf{V}_{0,i}\}_{i\in[5]}, \mathbf{A}^c)$ respectively as

$$\mathbf{A}_i = \mathbf{A}\mathbf{R}_i - x_i^*\mathbf{G}, \quad \mathbf{V}_{0,i} = \mathbf{A}\mathbf{R}_{0,i} - \boldsymbol{v}_0[i]\mathbf{G}, \quad \mathbf{A}^c = \mathbf{A}\mathbf{R}^c - \mathbf{G}$$

where $\boldsymbol{v}_0 = [1,0,0,0,0]$. The rest of the hybrid remains unchanged.

Hybrid H_3: In hybrid H_3, we change the generation of matrix $\mathbf{A}$ and vector $\boldsymbol{u}$ in public parameters pp.

Let $\mathbf{A}$ be a random matrix in $\mathbb{Z}_q^{n\times m}$. The construction of matrices $(\{\mathbf{A}_i\}_{i\in[\ell]}, \{\mathbf{V}_{0,i}\}_{i\in[5]}, \mathbf{A}^c)$ remains the same, as in hybrid H_2. Sample error vectors $\boldsymbol{e}$ that would be used in algorithm $\mathsf{SampleP}$ later. Then compute the error vector

$$\boldsymbol{e}_{\mathsf{BP}^*} \leftarrow \mathsf{Eval}_{\mathsf{ct}}(\mathsf{BP}, \boldsymbol{x}, \{\mathbf{A}_i, \boldsymbol{e}^T\mathbf{R}_i\}_{i\in[\ell]}, \{\mathbf{V}_{0,i}, \boldsymbol{e}^T\mathbf{R}_{0,i}\}_{i\in[5]}, \{\mathbf{A}^c, \boldsymbol{e}^T\mathbf{R}^c\})$$

and choose a correlation coefficient $\mu \leftarrow \mathcal{D}_\gamma$, and set vector $\boldsymbol{y}_1 = (\mu\boldsymbol{e}_{\mathsf{BP}^*} + \mathcal{D}_{\mathbb{Z}^m,\mathbf{Q}})q$, where

$$\mathbf{Q} = \beta^2\mathbf{I}_{m\times m} - \gamma^2\alpha^2\mathbf{R}_{\mathsf{BP}^*}^T\mathbf{R}_{\mathsf{BP}^*}$$

Then let $\boldsymbol{y} = (\boldsymbol{y}_0|\boldsymbol{y}_1)$, where $\boldsymbol{y}_0 \leftarrow \mathcal{D}_{\mathbf{Z}_q^m,\beta^2q^2\mathbf{I}_{m\times m}}$. Sample vector $\mathbf{r}_{\mathsf{BP}^*} \leftarrow \boldsymbol{y} + \mathcal{D}_{\mathbb{Z}^{2m}-\boldsymbol{y},(s^2-\beta^2)q^2\mathbf{I}_{2m\times 2m}}$, and compute matrix

$$\mathbf{V}_{\mathsf{BP}^*} \leftarrow \mathsf{Eval}_{\mathsf{pk}}(\mathsf{BP}, \{\mathbf{A}_i\}_{i\in[\ell]}, \{\mathbf{V}_{0,i}\}_{i\in[5]}, \mathbf{A}^c)$$

Set vector $\boldsymbol{u}$ in public parameters pp as $\mathbf{u} = [\mathbf{A}|\mathbf{V}_{\mathsf{BP}^*}]\cdot\mathbf{r}_{\mathsf{BP}^*}$. Since $\mathbf{A}$ is a random matrix without trapdoor $\mathbf{T}_\mathbf{A}$ to answer key queries, we will use trapdoor $\mathbf{T}_\mathbf{G}$ to answer queries as follows. Consider a secret key query for branching program BP such that $\mathsf{BP}(\boldsymbol{x}^*) = 0$. To respond, we do the following computations:

1. First, we compute
$$\mathbf{R}_{\mathsf{BP}} \leftarrow \mathsf{Eval}_{\mathsf{Sim}}(\mathsf{BP}, \boldsymbol{x}, \{\mathbf{R}_i\}_{i\in[\ell]}, \{\mathbf{R}_{0,i}\}_{i\in[5]}, \mathbf{R}^c, \mathbf{A})$$
to obtain a low-norm matrix $\mathbf{R}_{\mathsf{BP}} \in \mathbb{Z}_q^{m\times m}$ satisfying $\mathbf{A}\mathbf{R}_{\mathsf{BP}} - \mathsf{BP}(\boldsymbol{x}^*)\mathbf{G} = \mathbf{V}_{\mathsf{BP}}$.
2. Then, we sample $\boldsymbol{r}_{\mathsf{BP}}$ using
$$\boldsymbol{r}_{\mathsf{BP}} \leftarrow \mathsf{SampleRight}(\mathbf{A}, \mathbf{G}, \mathbf{R}_{\mathsf{BP}}, \mathbf{T}_\mathbf{G}, \boldsymbol{u}, sq)$$
such that
$$\boldsymbol{r}_{\mathsf{BP}}^T \cdot [\mathbf{A}|\mathbf{V}_{\mathsf{BP}} + \mathbf{G}] = \boldsymbol{u}$$
By Lemma 2.7, vector $\boldsymbol{r}_{\mathsf{BP}}$ is distributed as required.

The computation of answering P-sampler query, $\mathsf{SampleP}$ is the same as hybrid H_1 with error vectors $\boldsymbol{e}$, For faking receiver coins, $\mathsf{FakeRCoins}$, simply output the vector $\boldsymbol{r}_{\mathsf{BP}^*}$ pre-sampled in the generation of vector $\boldsymbol{u}$ before.

Hybrid H_4: In hybrid H_4, we change the generation order of vector $\boldsymbol{y}$ and error vector $\boldsymbol{e}$.

First sample vector $\boldsymbol{y} = (\boldsymbol{y}_0|\boldsymbol{y}_1) \leftarrow \mathcal{D}_{\mathbb{Z}^{2m},\beta^2q^2\mathbf{I}_{2m\times 2m}}$ and compute $\boldsymbol{r}_{\mathsf{BP}^*}$ from $\boldsymbol{y}$ as in previous hybrid. Next, we compute error term $\boldsymbol{e}$ as $\boldsymbol{e} = \nu\boldsymbol{y}_1^T\mathbf{R}_{\mathsf{BP}^*}/q + \mathcal{D}_{\mathbb{Z}^m,\mathbf{Q}'}$, where $\nu \leftarrow \mathcal{D}_\tau, \tau = \gamma\alpha^2/\beta^2$, and $\mathcal{D}_{\mathbb{Z}^m,\mathbf{Q}'}$ is sampled as $\mathbf{L}'\mathcal{D}_{\mathbf{Z}_1^m,\mathbf{I}_{m\times m}}$ for

$$\mathbf{Q}' = \mathbf{L}'\mathbf{L}'^T = \alpha^2\mathbf{I} - \tau^2\beta^2\mathbf{R}_{\mathsf{BP}^*}^T\mathbf{R}_{\mathsf{BP}^*} \tag{3}$$

Additionally, we modify the challenge ciphertext to be

$$\psi_0^* = \boldsymbol{s}^T\mathbf{A}/q + \boldsymbol{e}, \quad \psi_i^* = \psi_0^{*T}\mathbf{R}_i/q, \quad \psi_{0,i}^* = \psi_0^{*T}\mathbf{R}_{0,i}/q, \quad \psi^{*c} = \psi_0^{*T}\mathbf{R}^c/q$$

and $c^* = \boldsymbol{s}^T\boldsymbol{u} + \mathcal{D}_{\mathbf{Z}^m,\alpha\mathbf{I}_{m\times m}}$.

Hybrid H_5: In hybrid H_5, we change the generation order of secret key $\boldsymbol{r}_{\mathsf{BP}^*}$ and vector $\boldsymbol{y}$.

We first sample matrix $\boldsymbol{r}_{\mathsf{BP}^*}$ from discrete Gaussian distribution $\mathcal{D}_{\mathbb{Z}^{2m},s^2q^2\mathbf{I}_{2m\times 2m}}$, and set vector $\boldsymbol{u}$ in public parameters pp to be $\boldsymbol{u} = [\mathbf{A}|\mathbf{V}_{\mathsf{BP}^*}]\cdot\boldsymbol{r}_{\mathsf{BP}^*}$, where

$$\mathbf{V}_{\mathsf{BP}^*} \leftarrow \mathsf{Eval}_{\mathsf{pk}}(\mathsf{BP}, \{\mathbf{A}_i\}_{i\in[\ell]}, \{\mathbf{V}_{0,i}\}_{i\in[5]}, \{\mathbf{A}_i^c\}_{i\in[\ell]})$$

Then set $\boldsymbol{y} = (\boldsymbol{y}_0|\boldsymbol{y}_1) = \boldsymbol{r}_{\mathsf{BP}^*}/2 + \mathcal{D}_{\mathbb{Z}^{2m},(\beta^2-s^2/4)q^2\mathbf{I}_{2m\times 2m}}$. The remainder of the hybrid remains roughly the same. In particular, the challenge ciphertext ct^* is generated in the same manner as Hybrid H_4. We break the noise term $\boldsymbol{e}$ into two terms $\boldsymbol{e} = \boldsymbol{e}_0^{(1)} + \boldsymbol{e}_0^{(2)} + \nu\boldsymbol{y}_1^T\mathbf{R}_{\mathsf{BP}^*}/q$, where $\boldsymbol{e}_0^{(1)} \leftarrow \mathcal{D}_{\mathbb{Z}^m,\beta'\mathbf{I}_{m\times m}}$, $\boldsymbol{e}_0^{(2)} \leftarrow \mathcal{D}_{\mathbb{Z}^m,\mathbf{Q}'-\beta'^2\mathbf{I}_{m\times m}}$ and $\beta' = \alpha/2$.

Hybrid H_6: In hybrid H_6, we change how the challenge ciphertext is generated by using the Extended-LWE$^+$ instance.

First sample uniformly random vector $\boldsymbol{b} \in \mathbb{Z}^m$ and set the challenge ciphertext as

$$\psi_0^* = \boldsymbol{b}/q + \boldsymbol{e}_0^{(2)}, \quad \psi_i^* = \psi_0^{*T}\mathbf{R}_i/q, \quad \psi_{0,i}^* = \psi_0^{*T}\mathbf{R}_{0,i}/q, \quad \psi^{*c} = \psi_0^{*T}\mathbf{R}^c/q$$

and $c^* = \boldsymbol{r}_{\mathsf{BP}^*}^T[\mathbf{I}_{m\times m}|\mathbf{R}_{\mathsf{BP}^*}](\boldsymbol{b}/q - \boldsymbol{e}_0^{(1)}) + \mathcal{D}_{\mathbb{Z}^m,\alpha\mathbf{I}_{m\times m}}$.

Hybrid H_7: In hybrid H_7, we change the challenge ciphertext to be uniformly random.

In algorithm $\mathsf{SampleP}$, sample uniformly random vectors $\mathsf{ct} \in \mathbb{Z}_q^m \times \mathbb{Z}_q^{\ell m} \times \mathbb{Z}_q^m \times \mathbb{Z}_q^{5m} \times \mathbb{Z}_q$ and outputs ct.

Claim 4.8. *Assuming the semantic security of PKE* $\Pi = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$*, hybrid* H_0 *and* H_1 *are computationally indistinguishable.*

Proof. Observe there is only one difference between hybrids H_0 and H_1 occurs in the challenge ciphertext, i.e. the encryption (under PKE Π) of the random matrices $\mathbf{S}_i$ are replaced by encryption of 0. If a PPT adversary $\mathcal{A}$ distinguishes between the H_0-encryptions of $(\{\mathbf{R}_i\}_{i\in[\ell]}, \{\mathbf{R}_{0,i}\}_{i\in[5]}, \{\mathbf{R}^c\})$ and the H_1-encryptions of $\mathbf{0}$ with non-negligible probability, then we can construct an efficient reduction $\mathcal{B}$ that uses $\mathcal{A}$ to break the semantic security of PKE Π with similar probability. □

Claim 4.9. *Hybrids* H_1 *and* H_2 *are statistically indistinguishable.*

Proof. Observe the only difference between hybrids H_1 and H_2 is the generation of matrices

$$(\{\mathbf{A}_i\}_{i\in[\ell]}, \{\mathbf{V}_{0,i}\}_{i\in[5]}, \{\mathbf{A}_i^c\}_{i\in[\ell]})$$

The random matrices $(\{\mathbf{R}_i\}_{i\in[\ell]}, \{\mathbf{R}_{0,i}\}_{i\in[5]}, \{\mathbf{R}_i^c\}_{i\in[\ell]})$ are used in the generation of public parameters pp:

$$\mathbf{A}_i = \mathbf{A}\mathbf{R}_i - x_i^*\mathbf{G}, \quad \mathbf{V}_{0,i} = \mathbf{A}\mathbf{R}_{0,i} - \boldsymbol{v}_0[i]\mathbf{G}, \quad \mathbf{A}^c = \mathbf{A}\mathbf{R}^c - \mathbf{G}$$

and the construction of errors in challenge ciphertext

$$\boldsymbol{e}_i = \boldsymbol{e}^T\mathbf{R}_i, \quad \boldsymbol{e}^c = \boldsymbol{e}^T\mathbf{R}^c, \quad \boldsymbol{e}_{0,i} = \boldsymbol{e}^T\mathbf{R}_{0,i}$$

Then by Leftover Hash Lemma 2.4, the following two distributions are statistically indistinguishable

$$(\mathbf{A}, \{\mathbf{A}\mathbf{R}_i\}_{i\in[\ell]}, \{\mathbf{A}\mathbf{R}_{0,i}\}_{i\in[5]}, \{\mathbf{A}\mathbf{R}^c\}, \widetilde{\boldsymbol{e}}) \approx (\mathbf{A}, \{\mathbf{A}_i\}_{i\in[\ell]}, \{\mathbf{V}_{0,i}\}_{i\in[5]}, \{\mathbf{A}^c\}, \widetilde{\boldsymbol{e}})$$

where $\widetilde{\boldsymbol{e}} = (\{\boldsymbol{e}_i\}_{i\in[\ell]}, \{\boldsymbol{e}_{0,i}\}_{i\in[5]}, \{\boldsymbol{e}^c\})$. Hence, hybrid H_0 and H_1 are statistically indistinguishable. □

Claim 4.10. *Hybrids* H_2 *and* H_3 *are statistically indistinguishable.*

Proof. Observe there are three differences between hybrid H_2 and H_3: The generation of matrix $\mathbf{A}$ and vector $\boldsymbol{u}$ in pp, challenge secret key $\mathsf{sk}_{\mathsf{BP}^*}$ and the computation methods to answer secret key queries. By the property of algorithm $\mathsf{TrapGen}(q, n, m)$ in Lemma 2.6, the distribution of matrix $\mathbf{A}$ in hybrid H_2 is statistically close to uniform distribution, from which matrix $\mathbf{A}$ in hybrid H_3 is sampled.

For secret key queries regarding branching program BP, in hybrid H_2, we sample vector $\boldsymbol{r}_{\mathsf{BP}}$, using

$$\boldsymbol{r}_{\mathsf{BP}} \leftarrow \mathsf{SampleLeft}(\mathbf{A}, \mathbf{T}_{\mathbf{A}}, (\mathbf{V}_{\mathsf{BP}} + \mathbf{G}), \boldsymbol{u}, s)$$

While in hybrid H_3, we sample vector $\boldsymbol{r}_{\mathsf{BP}}$, using

$$\boldsymbol{r}_{\mathsf{BP}} \leftarrow \mathsf{SampleRight}(\mathbf{A}, \mathbf{G}, \mathbf{R}_{\mathsf{BP}}, \mathbf{T}_{\mathbf{G}}, \boldsymbol{u}, sq)$$

By setting the parameters appropriately as specified in Section 4.3, and the properties of algorithms $\mathsf{SampleLeft}$ and $\mathsf{SampleRight}$ in Lemma 2.7, the answers to secret key queries are statistically close.

By Leftover Hash Lemma 2.4, the distribution $([\mathbf{A}|\mathbf{V}_{\mathsf{BP}^*}], [\mathbf{A}|\mathbf{V}_{\mathsf{BP}^*}] \cdot \boldsymbol{r}_{\mathsf{BP}^*})$ and $([\mathbf{A}|\mathbf{V}_{\mathsf{BP}^*}], \boldsymbol{u})$ are statistically close. Hence, hybrid H_2 and H_3 are statistically indistinguishable. □

Claim 4.11. *Hybrids* H_3 *and* H_4 *are statistically indistinguishable.*

Proof. The only difference between the two experiments is in the choice of $\boldsymbol{y}$ and $\boldsymbol{e}$, specifically, the choice of the $\boldsymbol{y}_1$ component of $\boldsymbol{y} = (\boldsymbol{y}_0|\boldsymbol{y}_1)$. We will show that the joint distribution of $(\boldsymbol{e}, \boldsymbol{y}_1)$ is identically distributed in these two hybrids:

In hybrid H_3, $\boldsymbol{y}_1$ is set as $\boldsymbol{y}_1 = (\mu \boldsymbol{e}_{\mathsf{BP}^*} + \mathcal{D}_{\mathbb{Z}^m, \mathbf{Q}})q$, where $\mathbf{Q} = \beta^2 \mathbf{I}_{m\times m} - \gamma^2\alpha^2 \mathbf{R}_{\mathsf{BP}^*}^T \mathbf{R}_{\mathsf{BP}^*}$ with $\boldsymbol{e} \leftarrow \mathcal{D}_{\mathbb{Z}^m, \alpha^2 \mathbf{I}_{m\times m}}$ and

$$\boldsymbol{e}_{\mathsf{BP}^*} \leftarrow \mathsf{Eval}_{\mathsf{ct}}(\mathsf{BP}, \boldsymbol{x}, \{\mathbf{A}_i, \boldsymbol{e}^T\mathbf{R}_i\}_{i\in[\ell]}, \{\mathbf{V}_{0,i}, \boldsymbol{e}^T\mathbf{R}_{0,i}\}_{i\in[5]}, \{\mathbf{A}^c, \boldsymbol{e}^T\mathbf{R}^c\})$$

Therefore, in hybrid H_3, we may write the joint distribution of $(\boldsymbol{e}, \boldsymbol{y}_1)$ as $\mathbf{T}_1 \cdot \mathcal{D}_{\mathbb{Z}^{2m}, \mathbf{I}_{2m\times 2m}}$, where $\mathbf{T}_1 \stackrel{\text{def}}{=} \begin{pmatrix} \alpha \mathbf{I}_{m\times m} & \mathbf{0}_{m\times m} \\ \gamma\alpha q \mathbf{R}_{\mathsf{BP}^*}^T & q\mathbf{L} \end{pmatrix}$ for $\mathbf{Q} = \mathbf{L}\mathbf{L}^T \in \mathbb{Z}^{m\times m}$ via the Cholesky decomposition due to Lemma 2.1.

In hybrid H_4, vector $\boldsymbol{y} = (\boldsymbol{y}_0|\boldsymbol{y}_1)$ is sampled as $\boldsymbol{y} = (\boldsymbol{y}_0|\boldsymbol{y}_1) \leftarrow \mathcal{D}_{\mathbb{Z}^{2m}, \beta^2 q^2 \mathbf{I}_{2m\times 2m}}$. Then $\boldsymbol{e}$ is computed as $\boldsymbol{e} = \nu \boldsymbol{y}_1^T \mathbf{R}_{\mathsf{BP}^*}/q + \mathcal{D}_{\mathbb{Z}^m, \mathbf{Q}'}$, where $\nu \leftarrow \mathcal{D}_\tau, \tau = \gamma\alpha^2/\beta^2$, and $\mathbf{Q}' = \alpha^2\mathbf{I} - \tau^2\beta^2 \mathbf{R}_{\mathsf{BP}^*}^T \mathbf{R}_{\mathsf{BP}^*}$. Then in hybrid H_4, we may write the joint distribution of $(\boldsymbol{e}, \boldsymbol{y}_1)$ as $\mathbf{T}_2 \cdot \mathcal{D}_{\mathbb{Z}^{2m}, \mathbf{I}_{2m\times 2m}}$, where $\mathbf{T}_2 \stackrel{\text{def}}{=} \begin{pmatrix} \mathbf{L}' & \tau\beta \mathbf{R}_{\mathsf{BP}^*} \\ \mathbf{0}_{m\times m} & \beta q \mathbf{I}_{m\times m} \end{pmatrix}$ for $\mathbf{Q}' = \mathbf{L}'\mathbf{L}'^T \in \mathbb{Z}^{m\times m}$ via the Cholesky decomposition due to Lemma 2.1.

We claim equality of the following systems of equations:

$$\begin{aligned} \mathbf{T}_1\mathbf{T}_1^T &= \begin{pmatrix} \alpha^2\mathbf{I}_{m\times m} & \gamma\alpha^2 q \mathbf{R}_{\mathsf{BP}^*} \\ \gamma\alpha^2 q \mathbf{R}_{\mathsf{BP}^*}^T & \gamma^2\alpha^2 q^2 \mathbf{R}_{\mathsf{BP}^*}^T \mathbf{R}_{\mathsf{BP}^*} + q^2 \mathbf{L}\mathbf{L}^T \end{pmatrix} \\ &= \begin{pmatrix} \mathbf{L}'\mathbf{L}'^T + \tau^2\beta^2 \mathbf{R}_{\mathsf{BP}^*}\mathbf{R}_{\mathsf{BP}^*}^T & \tau\beta^2 q \mathbf{R}_{\mathsf{BP}^*} \\ \tau\beta^2 q \mathbf{R}_{\mathsf{BP}^*}^T & \beta^2 q^2 \mathbf{I}_{m\times m} \end{pmatrix} = \mathbf{T}_2\mathbf{T}_2^T. \end{aligned}$$

This fact may be seen quadrant-wise by our choice of $\tau = \gamma\alpha^2/\beta^2$ and the settings of $\mathbf{Q} = \mathbf{L}\mathbf{L}^T$ and $\mathbf{Q}' = \mathbf{L}'\mathbf{L}'^T$ in Equations (2) and (3). It then follows that $(\mathbf{T}_2^{-1}\mathbf{T}_1)(\mathbf{T}_2^{-1}\mathbf{T}_1)^T = \mathbf{I}_{2m\times 2m}$, implying $\mathbf{T}_1 = \mathbf{T}_2\mathbf{Q}^*$ for some orthogonal matrix $\mathbf{Q}^*$. Because the spherical Gaussian $\mathcal{D}_{\mathbb{Z}^{2m}, \mathbf{I}_{2m\times 2m}}$ is invariant under rigid transformations, we have $\mathbf{T}_1 \cdot \mathcal{D}_{\mathbb{Z}^{2m}, \mathbf{I}_{2m\times 2m}} = \mathbf{T}_2\mathbf{Q}^* \cdot \mathcal{D}_{\mathbb{Z}^{2m}, \mathbf{I}_{2m\times 2m}} = \mathbf{T}_2 \cdot \mathcal{D}_{\mathbb{Z}^{2m}, \mathbf{I}_{2m\times 2m}}$, and the claim follows. □

Claim 4.12. *Hybrids* H_4 *and* H_5 *are statistically indistinguishable.*

Proof. Observe the main difference between hybrids H_4 and H_5 is the order of generation of vectors $\boldsymbol{y}$ and $\boldsymbol{r}_{\mathsf{BP}^*}$: In hybrid H_4, we first sample $\boldsymbol{y} = (\boldsymbol{y}_0|\boldsymbol{y}_1) \leftarrow \mathcal{D}_{\mathbb{Z}^{2m},\beta^2 q^2 \mathbf{I}_{2m\times 2m}}$ and set $\boldsymbol{r}_{\mathsf{BP}^*} \leftarrow \boldsymbol{y} + \mathcal{D}_{\mathbb{Z}^{2m}-\boldsymbol{y},q^2(s^2-\beta^2)\mathbf{I}_{2m\times 2m}}$, while in hybrid H_5, we first sample $\boldsymbol{r}_{\mathsf{BP}^*} \leftarrow \mathcal{D}_{\mathbb{Z}^{2m},s^2 q^2 \mathbf{I}_{2m\times 2m}}$ and set $\boldsymbol{y} = (\boldsymbol{y}_0|\boldsymbol{y}_1) \leftarrow \boldsymbol{r}_{\mathsf{BP}^*}/2 + \mathcal{D}_{\mathbb{Z}^{2m},(\beta^2-s^2/4)q^2\mathbf{I}_{2m\times 2m}}$. By setting parameters appropriately as in Section 4.3, these two distributions are statistically close. □

Claim 4.13. *Assuming the hardness of extended-*$\mathsf{LWE}^+_{n,m,q,D_{\mathbb{Z}^m,\beta'},\mathbf{R}}$ *for any adversarially chosen distribution over matrices* $\mathbf{R} \in \mathbb{Z}_q^{m\times m}$*, then hybrids* H_5 *and* H_6 *are computationally indistinguishable.*

Proof. Suppose $\mathcal{A}$ has non-negligible advantage in distinguishing hybrid H_5 and H_6, then we use $\mathcal{A}$ to construct an extended-LWE$^+$ algorithm $\mathcal{B}$ as follows:

Invocation. $\mathcal{B}$ invokes adversary $\mathcal{A}$ to commit to a challenge attribute vector $\boldsymbol{x}^* = (x_1^*, ..., x_\ell^*)$ and challenge branching program BP^*. Then $\mathcal{B}$ generates $\mathbf{R}_{\mathsf{BP}^*}$ by first sampling $(\{\mathbf{R}_i\}_{i\in[\ell]}, \{\mathbf{R}_{0,i}\}_{i\in[5]}, \{\mathbf{R}^c\})$ as in the hybrid, and computes

$$\mathbf{R}_{\mathsf{BP}} \leftarrow \mathsf{Eval}_{\mathsf{Sim}}(\mathsf{BP}, \boldsymbol{x}, \{\mathbf{R}_i\}_{i\in[\ell]}, \{\mathbf{R}_{0,i}\}_{i\in[5]}, \{\mathbf{R}^c\}, \mathbf{A})$$

Then it receives an extended-LWE$^+$ instance for the matrix $\mathbf{R} = \mathbf{R}_{\mathsf{BP}^*}$ as follows:

$$\{\mathbf{A}, \boldsymbol{b} = \boldsymbol{s}^T\mathbf{A} + \boldsymbol{e}, \boldsymbol{z}_0, \boldsymbol{z}_1, \langle \boldsymbol{z}_0, \boldsymbol{b} - \boldsymbol{e}\rangle + e, \langle \boldsymbol{z}_1^T\mathbf{R}, \boldsymbol{b} - \boldsymbol{e}\rangle + e'\}$$

where $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$, $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_q^n$, $\boldsymbol{u} \xleftarrow{\$} \mathbb{Z}_q^m$, $\boldsymbol{e}, \boldsymbol{z}_0, \boldsymbol{z}_1 \xleftarrow{\$} \chi^n$ and $e, e' \xleftarrow{\$} \chi$. Algorithm $\mathcal{B}$ aims to leverage adversary $\mathcal{A}$'s output to solve the extended-LWE$^+$ assumption.

Setup. $\mathcal{B}$ generates matrices $(\{\mathbf{A}_i\}_{i\in[\ell]}, \{\mathbf{V}_{0,i}\}_{i\in[5]}, \{\mathbf{A}^c\})$ as specified in hybrid H_1. Then, $\mathcal{B}$ sets challenge secret key $\mathsf{sk}_{\mathsf{BP}^*} = \boldsymbol{r}_{\mathsf{BP}^*} = (\boldsymbol{r}_0^*|\boldsymbol{r}_1^*) = (\boldsymbol{z}_0|\boldsymbol{z}_1)$ from extended-LWE$^+$ instance and computes vector $\boldsymbol{u}$ as in hybrid H_5.

Secret key queries. $\mathcal{B}$ answers adversary $\mathcal{A}$'s secret key queries as in hybrid H_2.

Challenge ciphertext. $\mathcal{B}$ answers adversary $\mathcal{A}$'s P-sample query by setting

$$\psi_0^* = \boldsymbol{b}/q + \boldsymbol{e}_0^{(2)} + \nu\boldsymbol{y}_1^T\mathbf{R}_{\mathsf{BP}^*}/q, \quad \psi_i^* = \psi_0^{*T}\mathbf{R}_i/q, \quad \psi_{0,i}^* = \psi_0^{*T}\mathbf{R}_{0,i}/q, \quad \psi^{*c} = \psi_0^{*T}\mathbf{R}^c/q$$

and $c^* = \boldsymbol{r}_{\mathsf{BP}^*}^T[\mathbf{I}_{m\times m}|\mathbf{R}_{\mathsf{BP}^*}](\boldsymbol{b}/q - \boldsymbol{e}^{(1)}) + \mathcal{D}_{\mathbb{Z}^m,\alpha\mathbf{I}_{m\times m}}$.

Faking receiver coin query. $\mathcal{B}$ answers adversary $\mathcal{A}$'s faking receiver coin query by outputting the extended-LWE instance's vector $\mathsf{sk}_{\mathsf{BP}^*} = \boldsymbol{r}_{\mathsf{BP}^*}$.

Output. $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs.

We can rewrite the expression of $c^{*'}$ to be

$$\begin{aligned} c^{*'} &= ([\mathbf{A}^*|\mathbf{A}^*\mathbf{R}_{\mathsf{BP}^*}]\left(\begin{smallmatrix}\boldsymbol{z}_0\\ \boldsymbol{z}_1\end{smallmatrix}\right))^T\boldsymbol{s}/q + \mathcal{D}_{\mathbb{Z}_1,\alpha} \\ &= ((\boldsymbol{z}_0|\boldsymbol{z}_1)\left(\begin{smallmatrix}\mathbf{A}^{*T}\\ \mathbf{R}_{\mathsf{BP}^*}^T\mathbf{A}^{*T}\end{smallmatrix}\right))\boldsymbol{s}/q + D_{\mathbb{Z}_1,\alpha} = \boldsymbol{z}_0\mathbf{A}^{*T}\boldsymbol{s}/q + \boldsymbol{z}_1\mathbf{R}_{\mathsf{BP}^*}^T\mathbf{A}^{*T}\boldsymbol{s}/q + \mathcal{D}_{\mathbb{Z}_1,\alpha} \\ &= \langle \boldsymbol{z}_0, \boldsymbol{b}/q - \boldsymbol{e}^{(1)}\rangle + \langle \boldsymbol{z}_1^T\mathbf{R}_{\mathsf{BP}^*}, \boldsymbol{b}/q - \boldsymbol{e}^{(1)}\rangle + \mathcal{D}_{\mathbb{Z}_1,\alpha} \end{aligned}$$

We can see that if the eLWE$^+$ instance's vector $\boldsymbol{b}$ is pseudorandom, then the distribution simulated by $\mathcal{B}$ is exactly the same as H_5. If $\boldsymbol{b}$ is truly random and independent, then the distribution simulated by $\mathcal{B}$ is exactly the same as H_6. Therefore, if $\mathcal{A}$ can distinguish H_5 from H_6 with non-negligible probability, then $\mathcal{B}$ can break the $\mathsf{eLWE}^+_{n,m,q,\mathcal{D}_{(\alpha/2)q},\alpha',\mathbf{S}_{f^*}}$ problem for some $\alpha' \geq 0$ with non-negligible probability. □

Claim 4.14. *Hybrids* H_6 *and* H_7 *are statistically indistinguishable.*

Proof. Recall the only difference between hybrids H_6 and H_7 is the generation of challenge ciphertext. In hybrid H_7, we observe if ψ_0^* is chosen from uniform distribution, then by Leftover Hash Lemma 2.4, it holds

$$\psi_i^* = \psi_0^{*T}\mathbf{R}_i/q, \quad \psi_{0,i}^* = \psi_0^{*T}\mathbf{R}_{0,i}/q, \quad \psi^{*c} = \psi_0^{*T}\mathbf{R}^c/q$$

is also uniformly random (in their marginal distribution). Therefore, it remains to show that c^* is still uniformly random even conditioned on fixed samples of $(\psi_0^*, \{\psi_i^*\}_i, \{\psi_{0,i}^*\}_i, \{\psi^c\})$.

As calculated above, we can unfold the expression of c^* as

$$c^* = \langle \boldsymbol{z}_0, \boldsymbol{b}/q - \boldsymbol{x}^{(1)} \rangle + \langle \boldsymbol{z}_1^T \mathbf{R}_{\mathsf{BP}^*}, \boldsymbol{b}/q - \boldsymbol{x}^{(1)} \rangle + \mathcal{D}_{\mathbb{Z}_1,\alpha}$$

We note that $\boldsymbol{b}/q - \boldsymbol{x}^{(1)} = \psi_0^* - \boldsymbol{x}^{(1)} - \boldsymbol{x}^{(2)} - \nu\mathbf{R}_{\mathsf{BP}^*}\boldsymbol{y}_1/q$, thus if we show that

$$\langle \mathbf{R}_{\mathsf{BP}^*}\boldsymbol{z}_1, \nu\mathbf{R}_{\mathsf{BP}^*}\boldsymbol{y}_1/q \rangle$$

is close to uniform distribution (modulo 1), then c^* will also be close to the uniform distribution (modulo 1), as c^* is masked by this uniformly random number. Recall in hybrids, we set $\boldsymbol{y}_1 = \boldsymbol{z}_1/2 + (\text{shift})$, so it is sufficient to analyze

$$\langle \mathbf{R}_{\mathsf{BP}^*}\boldsymbol{z}_1, \nu\mathbf{R}_{\mathsf{BP}^*}\boldsymbol{y}_1/q \rangle = \nu\langle \mathbf{R}_{\mathsf{BP}^*}\boldsymbol{z}_1, \mathbf{R}_{\mathsf{BP}^*}\boldsymbol{z}_1/q \rangle = \nu||\mathbf{R}_{\mathsf{BP}^*}^*\boldsymbol{z}_1||^2/q$$

By applying Lemma 4.2 inductively on matrix $\mathbf{R}_{\mathsf{BP}^*}$, we can obtain that

$$||\mathbf{R}_{\mathsf{BP}^*}^*\boldsymbol{z}_1||^2/q \geq \frac{(||\mathbf{R}_{0,j}\boldsymbol{z}_1|| - \Theta(m^{1.5})\ell||\boldsymbol{z}_1||)^2}{q}$$

where $\mathbf{R}_{0,j} \in \{-1,1\}^{m\times m}$. Since vector $\boldsymbol{z}_1$ is sampled from Gaussian with width sq, so its two-norm is at least $\sqrt{m}(sq)$ with overwhelming probability. Then by Lemma 4.4, the distribution $\nu||\mathbf{R}_{\mathsf{BP}^*}^*\boldsymbol{z}_1||^2/q$ is a Gaussian distribution with width at least

$$d = \tau\frac{(\eta sqm - \Theta(m^2\ell)sq)^2}{q} = \frac{\gamma\alpha^2(\eta sqm - \Theta(m^2\ell)sq)^2}{\beta^2 q}$$

We recall again that ν was sampled from a Gaussian with parameter $\tau = \gamma\alpha^2/\beta^2$. By our setting of parameters, we have $d/\omega(\log(n)) \geq 1$. A Gaussian with such width is statistically close to uniform in the domain $\mathbb{Z}_1$. This completes the proof. □

This completes the proof of Lemma 4.7. Further, Theorem 4.5 follows from Lemmas 4.6 and 4.7. A (flexibly) bi-deniable ABE from LWE then follows. □

4.3 Parameter Setting

The parameters in Table 1 are selected in order to satisfy the following constraints:

Parameters	Description	Setting
n, m	lattice dimension	$n = \lambda, m = n^2 \log n$
ℓ	length of input to branching program	$\ell = n$
q	modulus (resp. bit-precision)	smallest prime $\geq n^{1.5} m^{2.5} \omega(\log n)$
α	sampling error terms $\boldsymbol{e}, e$	$\frac{1}{n^{2.5} \log^3 n}$
β	sampling correlation vector $\boldsymbol{y}$	$\alpha/2$
γ	sampling correlation coefficient μ	$\frac{1}{n \log^{1.5} n}$
s	sampling secret key $\boldsymbol{r}$	$3\beta/2$
η	scaling parameter for $\mathbf{R}_{0,j}$	$\Theta(m\ell)$

Table 1: Parameter Description and Simple Example Setting

- To ensure correctness in Lemma 4.6, we have $\alpha s q m(\eta\sqrt{m} + 2m^{1.5}\ell) \leq 1/4$.
- To ensure deniability in Hybrid H_7, we have $d/\omega(\log(n)) > \frac{\gamma\alpha^2(\eta s q m - \Theta(m^2 \ell s q))^2}{\beta^2 q \omega(\log(n))} > 1$.
- To ensure large enough LWE noise, we need $\alpha \geq (\sqrt{n}\log^{1+\delta} n)/q$.
- To apply the leftover hash lemma, we need $m \geq 2n\log(q)$.
- To ensure that the matrix $\mathbf{Q}$ in FakeRCoins is positive definite, we have $\beta \geq \alpha\gamma\sqrt{\eta\sqrt{m} + 2m^{1.5}\ell}$; To ensure that the matrix $\mathbf{Q}'$ in the security proof is positive definite, we have $\alpha \geq \tau\beta\sqrt{\eta\sqrt{m} + 2m^{1.5}\ell}$. This constraint will also imply that in the security proof, both $\mathbf{Q}'$ and $\mathbf{Q}' - \beta'\mathbf{I}_{m\times m}$ are positive definite (note $\beta' = \alpha/2$).
- To ensure hybrids H_3 and H_5 are well-defined, we have $s > \beta$ and $\beta > s/2$. Let $s := (3/2)\beta$.

Regev [Reg05] showed that for $q > \sqrt{m}/\beta'$, an efficient algorithm for $\mathsf{LWE}_{n,m,q,\chi}$ for $\chi = \mathcal{D}_{\beta' q}$ $\big($and $\beta' q \geq \sqrt{n}\omega(\log(n))\big)$ implies an efficient quantum algorithm for approximating the SIVP and GapSVP problems, to within $\tilde{O}(n/\beta')$ approximation factors in the worst case. Our example parameter setting yields a bi-deniable AB-BTS based on the (quantum) hardness of solving $\mathsf{SIVP}_{\widetilde{O}(n^{9.5})}$, respectively $\mathsf{GapSVP}_{\widetilde{O}(n^{9.5})}$. (We write this term to additionally absorb the $(1/q^2)$ loss from our LWE to eLWE^+ reduction.) We leave further optimizing the lattice problem approximation factor to future work.

4.4 From AB-BTS to Flexible Bi-Deniable ABE

We present the instantiation of a flexible bi-deniable ABE using our AB-BTS scheme described above. We let $\Sigma' = (\mathsf{Setup}', \mathsf{DenSetup}', \mathsf{Keygen}', \mathsf{SampleP}', \mathsf{SampleU}', \mathsf{TestP}', \mathsf{FakeRCoins}', \mathsf{FakeSCoins}')$ be an AB-BTS scheme. Then the flexible bi-deniable ABE $\Sigma = (\mathsf{Setup}, \mathsf{DenSetup}, \mathsf{Keygen}, \mathsf{Enc}, \mathsf{DenEnc}, \mathsf{Dec}, \mathsf{SendFake}, \mathsf{RecFake})$ is:

- $\mathsf{Setup}(1^\lambda)$: Run algorithm $(\mathsf{pp}', \mathsf{msk}') \leftarrow \mathsf{Setup}'(1^\lambda)$ in AB-BTS and set $\mathsf{pp} = \mathsf{pp}', \mathsf{msk} = \mathsf{msk}'$.
- $\mathsf{DenSetup}(1^\lambda)$: Run algorithm $(\mathsf{pp}', \mathsf{msk}', \mathsf{fk}') \leftarrow \mathsf{DenSetup}'(1^\lambda)$ in AB-BTS and set $\mathsf{pp} = \mathsf{pp}', \mathsf{msk} = \mathsf{msk}', \mathsf{fk} = (\mathsf{fk}', \mathsf{msk}')$.
- $\mathsf{Keygen}(\mathsf{msk}, f)$: Run algorithm $\mathsf{sk}'_f \leftarrow \mathsf{Keygen}'(\mathsf{msk}, f)$ in AB-BTS and set $\mathsf{sk}_f = \mathsf{sk}'_f$.
- $\mathsf{Enc}(\mathsf{pp}, \boldsymbol{x}, \mu; (r_S^{(1)}, r_S^{(2)}))$: On input the message $\mu \in \{0,1\}$, if $\mu = 0$, then run $c_i \leftarrow \mathsf{SampleU}'(\mathsf{pp}, \boldsymbol{x}; r_S^{(i)})$ for $i = 1, 2$, otherwise, $\mu = 1$, run $c_1 \leftarrow \mathsf{SampleU}'(\mathsf{pp}, \boldsymbol{x}; r_S^{(1)})$ and $c_2 \leftarrow \mathsf{SampleP}'(\mathsf{pp}, \boldsymbol{x}; r_S^{(2)})$. Output $\mathsf{ct}_{\boldsymbol{x}} = (c_1, c_2)$.

- $\mathsf{DenEnc}(\mathsf{pp}, \boldsymbol{x}, \mu; (r_S^{(1)}, r_S^{(2)}))$: On input the message $\mu \in \{0,1\}$, then run $c_i \leftarrow \mathsf{SampleP}'(\mathsf{pp}, \boldsymbol{x}; r_S^{(i)})$ for $i = 1, 2$, otherwise, $\mu = 1$, run $c_1 \leftarrow \mathsf{SampleU}'(\mathsf{pp}, \boldsymbol{x}; r_S^{(1)})$ and $c_2 \leftarrow \mathsf{SampleP}'(\mathsf{pp}, \boldsymbol{x}; r_S^{(2)})$. Output $\mathsf{ct}_{\boldsymbol{x}} = (c_1, c_2)$.

- $\mathsf{Dec}(\mathsf{ct}_{\boldsymbol{x}}, \mathsf{sk}_f)$: If $f(\boldsymbol{x}) \neq 0$, then output $\perp$. Otherwise, parse $\mathsf{ct}_{\boldsymbol{x}} = (c_1, c_2)$ and run $b_i \leftarrow \mathsf{TestP}'(\mathsf{sk}_f, c_i)$ for $i = 1, 2$. Output 0 if the $b_1 = b_2$ and 1 if $b_1 \neq b_2$.

- $\mathsf{SendFake}(\mathsf{pp}, r_S, \mu, \mu')$: If $\mu = \mu'$, return r_S. If $(\mu, \mu') = (0, 1)$, then run $r_S^{*(2)} \leftarrow \mathsf{FakeSCoins}'(\mathsf{pp}, r_S^{(2)})$ and return $(r_S^{(1)}, r_S^{*(2)})$. Else if $(\mu, \mu') = (1, 0)$, run $r_S^{*(1)} \leftarrow \mathsf{FakeSCoins}'(\mathsf{pp}, r_S^{(1)})$ and return $(r_S^{*(1)}, r_S^{(2)})$.

- $\mathsf{RecFake}(\mathsf{pp}, \mathsf{fk}, \mathsf{ct}_{\boldsymbol{x}}, f, \mu')$: Parse $\mathsf{ct}_{\boldsymbol{x}} = (c_1, c_2)$ and use fk to decrypt the ciphertext $\mathsf{ct}_{\boldsymbol{x}}$ then obtain the plaintext μ. If $\mu = \mu'$, then run the honest key generation of the BTS scheme, i.e. $\mathsf{sk}'_f \leftarrow \mathsf{Keygen}'(\mathsf{msk}', f)$. Otherwise, run $\mathsf{sk}'_f \leftarrow \mathsf{FakeRCoins}'(\mathsf{pp}, \mathsf{fk}, c_{\mu+1}, f)$. Return sk'_f.

Similar to the work by Canetti et al. [CDNO97] and O'Neil et al. [OPW11], the following, desired theorem can be proven in a straightforward manner.

Theorem 4.15. *Assume that Σ' is a flexible bi-deniable AB-BTS, as in Definition 3.2. Then Σ is a flexibly bi-deniable ABE, as in Definition 3.1.*

References

[ABB10] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In Gilbert [Gil10], pages 553–572.

[ABSV15] Prabhanjan Ananth, Zvika Brakerski, Gil Segev, and Vinod Vaikuntanathan. From selective to adaptive security in functional encryption. In *CRYPTO 2015, Part II*, LNCS, pages 657–677. Springer, August 2015.

[AFL15] Daniel Apon, Xiong Fan, and Feng-Hao Liu. Bi-deniable inner product encryption from LWE. *IACR Cryptology ePrint Archive*, 2015:993, 2015.

[AP12] Jacob Alperin-Sheriff and Chris Peikert. Circular and KDM security for identity-based encryption. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 334–352. Springer, May 2012.

[AP14] Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 297–314. Springer, August 2014.

[Bar86] David A Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in nc 1. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 1–5. ACM, 1986.

[Bar89] David A. Mix Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in nc^1. *J. Comput. Syst. Sci.*, 38(1):150–164, 1989.

[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, May 2014.

[BHY09] Mihir Bellare, Dennis Hofheinz, and Scott Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 1–35. Springer, April 2009.

[BLP+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In Boneh et al. [BRF13], pages 575–584.

[BLW15] Dan Boneh, Kevin Lewi, and David J. Wu. Constraining pseudorandom functions privately. *IACR Cryptology ePrint Archive*, 2015:1167, 2015.

[BNNO11] Rikke Bendlin, Jesper Buus Nielsen, Peter Sebastian Nordholt, and Claudio Orlandi. Lower and upper bounds for deniable public-key encryption. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 125–142. Springer, December 2011.

[BRF13] Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors. *45th ACM STOC*. ACM Press, June 2013.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 253–273. Springer, March 2011.

[BV14] Zvika Brakerski and Vinod Vaikuntanathan. Lattice-based FHE as secure as PKE. In Moni Naor, editor, *ITCS 2014*, pages 1–12. ACM, January 2014.

[CDNO97] Ran Canetti, Cynthia Dwork, Moni Naor, and Rafail Ostrovsky. Deniable encryption. In Burton S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 90–104. Springer, August 1997.

[CFGN96] Ran Canetti, Uriel Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *28th ACM STOC*, pages 639–648. ACM Press, May 1996.

[CGP] Ran Canetti, Shafi Goldwasser, and Oxana Poburinnaya. Adaptively secure two-party computation from indistinguishability obfuscation. pages 557–585.

[CHKP10] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In Gilbert [Gil10], pages 523–552.

[CIO16] Angelo De Caro, Vincenzo Iovino, and Adam O'Neill. Deniable functional encryption. In *Public-Key Cryptography - PKC 2016 - 19th IACR International Conference on Practice and Theory in Public-Key Cryptography, Taipei, Taiwan, March 6-9, 2016, Proceedings, Part I*, pages 196–222, 2016.

[DKR] Dana Dachman-Soled, Jonathan Katz, and Vanishree Rao. Adaptively secure, universally composable, multiparty computation in constant rounds. pages 586–613.

[DLZ] Dana Dachman-Soled, Feng-Hao Liu, and Hong-Sheng Zhou. Leakage-resilient circuits revisited - optimal number of computing components without leak-free hardware. pages 131–158.

[DRS04] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 523–540. Springer, May 2004.

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, October 2013.

[Gil10] Henri Gilbert, editor. *EUROCRYPT 2010*, volume 6110 of *LNCS*. Springer, May 2010.

[GKP+13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In Boneh et al. [BRF13], pages 555–564.

[GP] Sanjam Garg and Antigoni Polychroniadou. Two-round adaptively secure MPC from indistinguishability obfuscation. pages 614–637.

[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 06*, pages 89–98. ACM Press, October / November 2006. Available as Cryptology ePrint Archive Report 2006/309.

[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.

[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, August 2013.

[GV15] Sergey Gorbunov and Dhinakaran Vinayagamurthy. Riding on asymmetry: Efficient abe for branching programs. In *Advances in Cryptology–ASIACRYPT 2015*, pages 550–574. Springer, 2015.

[GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from LWE. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, pages 503–523, 2015.

[KTZ13] Jonathan Katz, Aishwarya Thiruvengadam, and Hong-Sheng Zhou. Feasibility and infeasibility of adaptively secure fully homomorphic encryption. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 14–31. Springer, February / March 2013.

[MM11] Daniele Micciancio and Petros Mol. Pseudorandom knapsacks and the sample complexity of LWE search-to-decision reductions. In Rogaway [Rog11], pages 465–484.

[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, April 2012.

[OPW11] Adam O'Neill, Chris Peikert, and Brent Waters. Bi-deniable public-key encryption. In Rogaway [Rog11], pages 525–542.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005.

[Rog11] Phillip Rogaway, editor. *CRYPTO 2011*, volume 6841 of *LNCS*. Springer, August 2011.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *46th ACM STOC*, pages 475–484. ACM Press, May / June 2014.