Multi-Input Attribute Based Encryption and Predicate Encryption

Shweta Agrawal* Anshu Yadav† Shota Yamada‡

Abstract

Motivated by several new and natural applications, we initiate the study of multi-input predicate encryption (miPE) and further develop multi-input attribute based encryption (miABE). Our contributions are:

1. *Formalizing Security:* We provide definitions for miABE and miPE in the symmetric key setting and formalize security in the standard *indistinguishability* (IND) paradigm, against *unbounded* collusions.
2. *Two-input* ABE *for* NC_1 *from* LWE *and Pairings:* We provide the first constructions for two-input *key-policy* ABE for NC_1 from LWE and pairings. Our construction leverages a surprising connection between techniques recently developed by Agrawal and Yamada (Eurocrypt, 2020) in the context of succinct *single*-input *ciphertext-policy* ABE, to the seemingly unrelated problem of *two*-input *key-policy* ABE. Similarly to Agrawal-Yamada, our construction is proven secure in the bilinear generic group model. By leveraging inner product functional encryption and using (a variant of) the KOALA knowledge assumption, we obtain a construction in the standard model analogously to Agrawal, Wichs and Yamada (TCC, 2020).
3. *Heuristic two-input* ABE *for* P *from Lattices:* We show that techniques developed for succinct single-input ciphertext-policy ABE by Brakerski and Vaikuntanathan (ITCS 2022) can also be seen from the lens of miABE and obtain the first two-input key-policy ABE from lattices for P.
4. *Heuristic three-input* ABE *and* PE *for* NC_1 *from Pairings and Lattices:* We obtain the first *three*-input ABE for NC_1 by harnessing the powers of both the Agrawal-Yamada and the Brakerski-Vaikuntanathan constructions.
5. *Multi-input* ABE *to multi-input* PE *via Lockable Obfuscation:* We provide a generic compiler that lifts multi-input ABE to multi-input PE by relying on the hiding properties of Lockable Obfuscation (LO) by Wichs-Zirdelis and Goyal-Koppula-Waters (FOCS 2018), which can be based on LWE. Our compiler generalises such a compiler for single-input setting to the much more challenging setting of multiple inputs. By instantiating our compiler with our new two and three-input ABE schemes, we obtain the first constructions of two and three-input PE schemes.

Our constructions of multi-input ABE provide the first improvement to the compression factor of *non-trivially exponentially efficient Witness Encryption* defined by Brakerski et al. (SCN 2018) without relying on compact functional encryption or indistinguishability obfuscation. We believe that the unexpected connection between succinct single-input ciphertext-policy ABE and multi-input key-policy ABE may lead to a new pathway for witness encryption.

*IIT Madras, shweta@cse.iitm.ac.in

†IIT Madras, anshu.yadav06@gmail.com

‡National Institute of Advanced Industrial Science and Technology (AIST), Tokyo, yamada-shota@aist.go.jp

Contents

1 Introduction

Attribute based encryption (ABE) is a generalization of public key encryption which enables fine grained access control on encrypted data. In an ABE scheme, the ciphertext is associated with a secret message m and a public *attribute* vector $\mathbf{x}$ while a secret key is associated with a function f. Decryption succeeds to reveal m if and only if $f(\mathbf{x}) = 1$. Security seeks ciphertext indistinguishability in the presence of *collusion* attacks, namely an adversary possessing a collection of keys $\{\mathsf{sk}_{f_i}\}_{i\in[\text{poly}]}$ should not be able to distinguish between ciphertexts corresponding to $(\mathbf{x}, m_0)$ and $(\mathbf{x}, m_1)$ unless one of the keys $\mathsf{sk}_{f_{i^*}}$ is *individually* authorised to decrypt, i.e. $f_{i^*}(\mathbf{x}) = 1$. ABE comes in two flavours – "key-policy" and "ciphertext-policy", depending on whether the function f is embedded in the key or the ciphertext.

The stronger notion of predicate encryption (PE) [BW07, SBC+07, KSW08, GVW15] further requires the attribute vector $\mathbf{x}$ to be hidden so that ciphertexts corresponding to $(\mathbf{x}_0, m_0)$ and $(\mathbf{x}_1, m_1)$ remain indistinguishable so long as $f_i(\mathbf{x}_0) = f_i(\mathbf{x}_1) = 0$ for all secret keys $\{\mathsf{sk}_{f_i}\}_{i\in[\text{poly}]}$ seen by the adversary.

Both ABE and PE have been widely studied, and possess elegant instantiations from a variety of assumptions [SW05, GPSW06, BW07, KSW08, LOS+10, OT10, OT12, CW14, AFV11, LW11, LW12, Wat12, GVW13, Wee14, Att14, BGG+14, GVW15, GV15, BV16, BW07, SBC+07, KSW08, GVW15]. Despite all this amazing progress, however, all known constructions support the single input setting – namely, the function f embedded in the secret key sk_f has arity one, so that the secret key can be used to decrypt only a single ciphertext at a time. While the more realistic multi-input setting has been studied for other closely related notions such as fully homomorphic encryption [LATV12, CM15, MW16] and functional encryption [GGG+14, AJ15, AGRW17, DOT18, ACF+18, CDG+18, Tom19, ABKW19, ABG19, LT19, AGT21], this has not been investigated at all in the context of predicate encryption, and only sparingly [BJK+18] in the context of attribute based encryption.

Supporting Multiple Sources. We argue that the multi-input setting is important even in the context of ABE and PE and generalizing these primitives to support multiple sources enables a host of new and natural applications. At the heart of the multi-input setting, for any primitive, is the fact that data generated independently in multiple locations may be correlated in meaningful ways, and it is often pertinent to consider the input as a concatenation of these correlated partial inputs. For instance, a patient is likely to visit different medical centres for treatment of different diseases and her overall medical record is a concatenation of the medical data generated at different centers. Similarly, a company is likely to conduct research and development related to a given technology in different locations but the complete data pertaining to that technology is a concatenation of these. Moreover, to organize data logically and to benefit from cloud storage and computing, it is useful for each source to upload this encrypted data to a central server. Now, it may be desirable to provide restricted access to relevant consumers of the data, exactly as in ABE for encrypted access control (say) or PE for encrypted search (say), but with the caveat that the input was generated in a distributed manner and is encoded in multiple ciphertexts.

For concreteness, consider a doctor who is treating Covid patients and wants to understand the relation between Covid and other medical conditions such as asthma or cancer, each of which are treated at different locations. The records of a given patient are encrypted independently and stored in a central repository, and the doctor can be given a key that filters stored (encrypted) records according to criteria such as *condition = 'Covid' and condition = 'asthma' and age group ='60-80'* and enables decryption of these. Similarly, a company (e.g. IBM) which conducts research in quantum technologies is likely to have different teams for theoretical and experimental research, and these teams are likely to work in different locations – indeed, even members of the

same team may not be co-located. Data pertaining to the research could be stored encrypted in a central location where individual ciphertexts are generated independently, and the company may desire to give restricted access to a patent office. As a third example, a company may have been sued for some malpractice, and the data pertinent to the case could span multiple locations. Now the company may wish to provide restricted access to a law firm which enables decryption only of the data pertaining to the lawsuit, encrypted independently by multiple sources. A possible solution may be to gather all the information at a central entity and then use single input ABE or PE as before, but there are two problems with this approach: (i)if data is transmitted unencrypted to the central server it creates vulnerability – this can be avoided by each source encrypting to the server's public key, and the server decrypting and re-encrypting using single input schemes, but this is wasteful and cumbersome, (ii) one may desire to use an untrusted commercial cloud server to store the encrypted data, in which case the step of creating the ciphertext at a central server in step (i) is completely redundant and doubly inefficient.

Multi-input attribute based encryption (miABE) or predicate encryption (miPE) arise as natural fits to the above applications. Similarly to the single input case, the secret key corresponds to a function f but the arity of this function can now be $k > 1$ – we may have k ciphertexts generated independently encoding $(\mathbf{x}_i, m_i)_{i\in[k]}$, and decryption reveals $(m_1, \ldots, m_k)$ if and only if $f(\mathbf{x}_1, \ldots, \mathbf{x}_k) = 1$. Indeed, any application of single input ABE and PE where the underlying data is generated in multiple locations and is correlated in meaningful ways can benefit from the abstraction of multi-input ABE and PE.

Prior Work. Brakerski et al. [BJK+18] studied the notion of miABE and showed that miABE for polynomial arity implies *witness encryption* (WE). However, though they provided the first definition of miABE, they only used it as a new pathway for achieving witness encryption, not as a notion with its own applications – in their definition, only the first encryptor has any input, since this suffices for WE. They do not consider strong notions of security or provide any constructions of miABE. They also defined the notion of non-trivially exponentially efficient witness encryption (XWE), where the encryption run-time is only required to be much smaller than the trivial 2^m bound for NP relations with witness size m. They show how to construct such XWE schemes for all of NP with encryption run-time $2^{m/2}$ using the single input ABE by [GVW13]. For encryption run-time $2^{\gamma\cdot m}$, the term γ is denoted as *compression* factor, and they explicitly left open the problem of constructing XWE schemes with an improved compression factor.

Both ABE and PE can be captured as special cases of *functional encryption* [SW05, BSW11], which has been studied extensively, in both the single-input [SW05, BSW11, GVW13, BGG+14] and multi-input setting [GGG+14, AJ15, AGRW17, DOT18, ACF+18, CDG+18, Tom19, ABKW19, ABG19, LT19, AGT21]. Recall that in functional encryption (FE), a secret key is associated with a function f, a ciphertext is associated with an input $\mathbf{x}$ and decryption allows to recover $f(\mathbf{x})$ and nothing else. It is easy to see that PE and ABE are both special cases of FE – in particular, both PE and ABE achieve the same functionality but restrict the security requirements of FE. In PE, we ask that the attribute $\mathbf{x}$ be hidden but only when the adversary does not see any decrypting keys, namely $f_i(\mathbf{x}) = 0$ for all function keys f_i received by the adversary. On the other hand, in FE, the attacker may request a key sk_f, so long as f does not distinguish the challenge messages $(\mathbf{x}_0, m_0), (\mathbf{x}_1, m_1)$, namely, we may have $f(\mathbf{x}_0) = f(\mathbf{x}_1) = 1$ so long as $m_0 = m_1$[1]. In the even weaker ABE, we do not ask any notion of hiding for $\mathbf{x}$, and this may be provided in the clear with the ciphertext.

[1]We note that a message m separate from attribute $\mathbf{x}$ is not required in the definition of FE, but we include it here for simpler comparison with PE and ABE.

Why not Functional Encryption? The informed reader may wonder what is the advantage of studying primitives like miPE or miABE when these are special cases of multi-input functional encryption (miFE), which has recently been constructed from standard assumptions [JLS21, AJ15]. It was shown by [AJ15, BV15] that FE satisfying a certain efficiency property (known as *compactness*) implies multi-input functional encryption, which in turn implies the powerful primitive of *indistinguishability obfuscation* (iO) [BGI+01]. A long line of exciting works endeavoured to construct compact FE (and hence iO) from standard assumptions [Lin16, Lin17, LV16, Agr19, AJL+19, JLMS19, GJLS21], coming ever-closer, until the very recent work of Jain, Lin and Sahai closed the last remaining gap and achieved this much sought after goal [JLS21, JLS22]. In [JLS21, JLS22], the authors provide a construction for compact FE, which in turn implies miFE for polynomial arity (albeit with exponential loss in the reduction).

Going via the route of compact FE, we obtain an exciting feasibility result for miFE and hence miABE as well as miPE. However, we argue that using something as strong as miFE or iO to construct miABE and miPE is undesirable, and indeed an "overkill" for the following reasons:

- *Assumptions:* Compact FE of [JLS21] is constructed via a careful combination of 4 assumptions – Learning Parity with Noise (LPN), Learning With Errors (LWE), SXDH assumption on Pairings, and pseudorandom generators computable in constant depth. In the follow-up work of [JLS22], this set of assumptions was trimmed to exclude LWE. Therefore any construction built using compact FE must make at least this set of assumptions, which is restrictive. A major goal in the theory of cryptography is developing constructions from diverse assumptions.

- *Complexity:* The construction of compact FE is extremely complex, comprising a series of careful steps, and this must then be lifted to miFE using another complex construction [AJ15]. Unlike FE, both PE and ABE are *much simpler, "all or nothing"* primitives and permit direct constructions in the single-input setting [GVW13, BGG+14, GVW15]. Do we need the full complexity of an miFE construction to get miPE or miABE? Indeed, even in the context of miFE, there is a large body of work that studies direct constructions for smaller function classes such as linear and quadratic functions [AGRW17, DOT18, ACF+18, CDG+18, Tom19, ABKW19, ABG19, LT19, AGT21].

- *New Techniques:* Finally and most importantly, we believe that it is extremely useful to develop new techniques for simpler primitives that are not known to be in obfustopia, and provide direct constructions. While direct constructions are likely to be more efficient, and are interesting in their own right, they may also lead to new pathways even for obfustopia primitives such as witness encryption or compact FE. Note that the only known construction of FE from standard assumptions is by [JLS21, JLS22], which makes crucial (and surprising) use of LPN in order to overcome a technical barrier – is LPN necessary for other primitives implied by compact FE? We believe that exploring new methods to construct weaker primitives is of central importance in developing better understanding of cryptographic assumptions, their power and limits.

1.1 Our Results

In this work, we initiate the study of multi-input predicate and attribute based encryption (miABE and miPE) and make the following contributions:

1. *Formalizing Security:* We provide definitions for miABE and miPE in the *symmetric* key setting and formalize two security notions in the standard *indistinguishability* (IND)

paradigm, against *unbounded* collusions. The first (regular) notion of security assumes that the attacker does not receive any decrypting keys, as is standard in the case of PE/ABE. The second *strong* notion, allows some decrypting queries in restricted settings.

2. *Two-input* ABE *for* NC_1 *from* LWE *and Pairings:* We provide the first constructions for two-input *key-policy* ABE for NC_1 from LWE and pairings. Our construction leverages a surprising connection between techniques recently developed by Agrawal and Yamada [AY20] in the context of succinct *single*-input *ciphertext-policy* ABE, to the seemingly unrelated problem of *two*-input *key-policy* ABE. Similarly to [AY20], our construction is proven secure in the bilinear generic group model. By leveraging inner product functional encryption and using (a variant of) the KOALA knowledge assumption, we obtain a construction in the standard model analogously to Agrawal, Wichs and Yamada [AWY20].

3. *Heuristic two-input* ABE *for* P *from Lattices:* We show that techniques developed for succinct single-input ciphertext-policy ABE by Brakerski and Vaikuntanathan [BV22] can also be seen from the lens of miABE and obtain the first two-input key-policy ABE from lattices for P. Similarly to [BV22], this construction is heuristic.

4. *Heuristic three-input* ABE *and* PE *for* NC_1 *from Pairings and Lattices:* We obtain the first *three*-input ABE for NC_1 by harnessing the powers of both the Agrawal-Yamada [AY20] and the Brakerski-Vaikuntanathan [BV22] constructions.

5. *Multi-input* ABE *to multi-input* PE *via Lockable Obfuscation:* We provide a generic compiler that lifts multi-input ABE to multi-input PE by relying on the hiding properties of Lockable Obfuscation (LO) by Wichs-Zirdelis and Goyal-Koppula-Waters (FOCS 2018), which can be based on LWE. Our compiler generalises such a compiler for single-input setting to the much more challenging setting of multiple inputs. By instantiating our compiler with our new two and three-input ABE schemes, we obtain the first constructions of two and three-input PE schemes.

Our constructions of multi-input ABE provide the first improvement to the compression factor (from 1/2 to 1/3 or 1/4) of *non-trivially exponentially efficient Witness Encryption* defined by Brakerski et al. [BJK+18] without relying on compact functional encryption or indistinguishability obfuscation. We believe that the unexpected connection between succinct single-input ciphertext-policy ABE and multi-input key-policy ABE may lead to a new pathway for witness encryption.

1.2 Our Techniques

Modeling Multi-Input Attribute Based and Predicate Encryption. Our first contribution is to model multi-input attribute based encryption (miABE) and predicate encryption (miPE) as relevant primitives in their own right. To begin, we observe that similarly to multi-input functional encryption (miFE) [GGG+14], these primitives are meaningful primarily in the symmetric key setting where the encryptor requires a secret key to compute a ciphertext. This is to prevent the primitive becoming trivial due to excessive leakage occurring by virtue of functionality. In more detail, let us say k encryptors compute an unbounded number ciphertexts in each slot, i.e. $\{(\mathbf{x}_1^j, m_1^j), \ldots (\mathbf{x}_k^j, m_k^j)\}_{j\in[\mathrm{poly}]}$ and the adversary obtains secret keys corresponding to functions $\{f_i\}_{i\in[\mathrm{poly}]}$. In the multi-input setting, ciphertexts across slots can be combined,

allowing the adversary to compute $f_i(\mathbf{x}_1^{j_1}, \mathbf{x}_2^{j_2}, \ldots, \mathbf{x}_k^{j_k})$ for any indices $i, j_1, \ldots, j_k \in [\text{poly}]$. In the public key setting, an adversary can easily encrypt messages for various attributes of its choice and decrypt these with the challenge ciphertext in a given slot to learn a potentially unbounded amount of information. [2] Due to this, we believe that the primitives of miABE and miPE are meaningful in the symmetric key setting where encryption also requires a secret key.

For security, we require the standard notion of ciphertext indistinguishability in the presence of collusion attacks, as in the single-input setting. Recall that in the single-input setting, the adversary cannot request any decrypting keys for challenge ciphertexts to prevent trivial attacks. However, since we are in the symmetric key setting where the adversary cannot encrypt herself, we propose an additional notion of *strong* security which also permits the adversary to request decrypting ciphertexts in some cases. In more detail, for the case of miABE, our strong security game allows the attacker to request function keys for $\{f_i\}_{i\in[\text{poly}]}$ and ciphertexts for tuples $\{(\mathbf{x}_1^j, m_{\beta,1}^j), \ldots, (\mathbf{x}_k^j, m_{\beta,k}^j)\}_{\beta\in\{0,1\}, j\in[\text{poly}]}$ so that it may hold that $f_i(\mathbf{x}_1^{j_1}, \ldots, \mathbf{x}_k^{j_k}) = 1$ for some $i, j_1, \ldots, j_k \in [\text{poly}]$ as long as the challenge messages do not distinguish, i.e. $(m_{1,0}^{j_1} = m_{1,1}^{j_1}), \ldots, (m_{k,0}^{j_k} = m_{k,1}^{j_k})$. For the case of miPE, we analogously define a strong version of security by asking that if $f_i(\mathbf{x}_{1,\beta}^{j_1}, \ldots, \mathbf{x}_{k,\beta}^{j_k}) = 1$ holds for some $i, j_1, \ldots, j_k \in [\text{poly}]$ and $\beta \in \{0, 1\}$, then it is also true that $(\mathbf{x}_{1,0}^{j_1}, \ldots, \mathbf{x}_{k,0}^{j_k}) = (\mathbf{x}_{1,1}^{j_1}, \ldots, \mathbf{x}_{k,1}^{j_k})$ and $(m_{1,0}^{j_1}, \ldots, m_{k,0}^{j_k}) = (m_{1,1}^{j_1}, \ldots, m_{k,1}^{j_k})$. For more details, please see Section 3.

Constructing Two Input ABE from LWE and Bilinear GGM. In constructing two input ABE (2ABE), the main difficulty is to satisfy two seemingly contradicting requirements at the same time: (1) the two ciphertexts should be created independently, (2) these ciphertexts should be combined in a way that decryption is possible. If we look at specific ABE schemes (e.g., [GPSW06, BGG+14]), it seems that these requirements cannot be satisfied simultaneously. If we want to satisfy the second requirement, the two ciphertexts should have common randomness. However to satisfy the first requirement, the randomness in the two ciphertexts needs to be sampled independently. An approach might be to fix the randomness and put it into the master secret key which is then used by both ciphertexts – but this will compromise security since fresh randomness is crucial in safeguarding semantic security.

Generating Joint Randomness: For resolving this problem, we consider a scheme that modifies two independently generated ciphertexts so that they have common randomness and then "joins" them. This common randomness is jointly generated using independently chosen randomness in each ciphertext by using a pairing operation. Specifically, we compute a ciphertext for slot 1 with randomness t_1 and encode it in $\mathbb{G}_1$ and similarly, for slot 2 with randomness t_2 in $\mathbb{G}_2$, where $\mathbb{G} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a pairing group with prime order q. Then, both ciphertexts may be combined to form a new ciphertext with respect to the randomness $t_1 t_2$ on $\mathbb{G}_T$. This approach seems to be promising, because we can uniquely separate every pair of ciphertexts, since each pair (i, j) will have unique randomness $t_1^i t_2^j$. In the generic group model, this is sufficient to prohibit "mix and match" attacks that try to combine components of different ciphertexts in

[2] The triviality of public-key miABE depends on the function class being supported. For example, consider the inner product functionality (which is in NC_1) defined as - let $f_{\mathbf{v}}(\mathbf{x}_1, \mathbf{x}_2) = 1$ if $\langle \mathbf{v}, \mathbf{x}_1 || \mathbf{x}_2 \rangle = 0$, where $\mathbf{x}_1$ and $\mathbf{x}_2$ are inputs in the ciphertext and $\mathbf{v}$ is in the key. Given a slot-1 ciphertext $\mathsf{CT}_1(\mathbf{x}_1, m_1)$ we would like to argue that m_1 remains hidden. However, in the public key case, it is possible to compute slot-2 ciphertext $\mathsf{CT}_2(\mathbf{x}_2, m_2)$ so that for any $v, \mathbf{x}_1$ (note that these are public) the decryption condition can be satisfied and the message m_1 can be recovered. This argument can also be extended to some interesting polynomials. On the other hand, there are function classes in NC_1, such as 3-SAT where it would be hard for the attacker to find a satisfying input and even the public key setting would not create excessive leakage (although it is unclear if such a functionality is useful in practice).

the same slot.

Moving Beyond Degree 2: However, since we "used up" the pairing operation here, it appears we cannot perform more than linear operations on the generated ciphertext, which would severely restrict the function class supported by our construction. In particular, pairing based ABE schemes seem not to be compatible with the above approach, because these require additional multiplication in the exponent during decryption, which cannot be supported using a bilinear map. However, at this juncture, a trick suggested by Agrawal and Yamada [AY20] comes to our rescue – to combine lattice based ABE with bilinear maps in a way that lets us get the "best of both".

At a high level, the Agrawal-Yamada trick rests on the observation that in certain lattice based ABE schemes [BGG+14, GV15], decryption is structured as follows: (i) evaluate the circuit f on ciphertext encodings of $\mathbf{x}$, (ii) compute a matrix-vector product of the ciphertext matrix and secret key vector, (iii) perform a rounding operation to recover the message. Crucially, step (i) in the above description is in fact a *linear* operation over the encodings, even for circuits in P, and the only nonlinear part of decryption is the rounding operation in step (iii). They observe that steps (i) and (ii) may be done "upstairs" in the exponent and step (iii) may be done "downstairs" by recovering the exponent brute force, when it is small enough. Importantly, the exponent is small enough when the circuit class is restricted to NC_1 using asymmetry in noise growth [GV15, GVW13]. While this idea was developed in the context of a single-input ciphertext-policy ABE, it appears to be exactly what we need for *two*-input key-policy ABE!

Perspective: Connection to Broadcast Encryption: In hindsight, the application of optimal broadcast encryption requires *succinctness* of the ciphertext, which recent constructions [BV22, AY20, AWY20] obtain by relying on the *decomposability* of specific ABE schemes [BGG+14, GV15] – this decomposability is also what the multi-input setting intrinsically requires, albeit for a different reason. In more detail, decomposability means that the ciphertext for a vector $\mathbf{x}$ can be decomposed into $|\mathbf{x}|$ ciphertext components each encoding a single bit $\mathbf{x}_i$, and these components can be tied together using common randomness to yield a complete ciphertext. The bit by bit encoding of the vector allows $2|\mathbf{x}|$ ciphertext components, each component encoding both bits for a given position, to together encode $2^{|\mathbf{x}|}$ possible values of $\mathbf{x}$, which leads to the succinctness of ciphertext in optimal broadcast encryption schemes [BV22, AY20, AWY20]. In the setting of multi-input ABE, decomposability allows to morph independently generated *full* ciphertexts with distinct randomness to *components of a single* ciphertext with common randomness. The randomness is "merged" using pairings (or lattices, see below) and the resultant ciphertext can now be treated like the ciphertext of a single input scheme.

Adapting to the 2ABE *Setting:* Let us recall the structure of the ciphertext in scheme of Boneh et al. [BGG+14], which is denoted as BGG^+ hereafter. As discussed above, a ciphertext for an attribute $\mathbf{x} \in [2\ell]$[3] in BGG^+ is generated by first generating LWE encodings (their exact structure is not important for this overview) for *all possible* values of the attribute $\mathbf{x}$, namely, $\{\psi_{i,b}\}_{i\in[2\ell],b\in\{0,1\}}$ (along with other components which are not relevant here) and then selecting $\{\psi_{i,x_i}\}_{i\in[2\ell]}$ based on $\mathbf{x}$, where x_i is the i-th bit of the attribute string $\mathbf{x}$.

Given the above structure, a candidate scheme works as follows. The setup algorithm computes encodings for all possible $\mathbf{x}$, namely $\{\psi_{i,b}\}_{i,b}$ and puts them into the master secret key. The encryptor for slot 1 chooses $t_1 \leftarrow \mathbb{Z}_q$ and encodes $(t_1, \{t_1\psi_{i,x_{1,i}}\}_{i\in[\ell]})$ in the exponent of $\mathbb{G}_1$. Similarly, the encryptor for slot 2 chooses $t_2 \leftarrow \mathbb{Z}_q$ and encodes $(t_2, \{t_2\psi_{i,x_{2,i-\ell}}\}_{i\in[\ell+1,2\ell]})$ in

[3]The length of the attribute is set to 2ℓ to match our two-input setting.

the exponent of $\mathbb{G}_2$. In decryption, we compute a pairing of matching components of the two ciphertexts to obtain $(t_1 t_2, \{t_1 t_2 \psi_{i,x_i}\}_{i\in[2\ell]})$ in the exponent of $\mathbb{G}_T$. Using the BGG^+ decryption procedure described above, we may perform linear operations to evaluate the circuit, apply the BGG^+ secret key and obtain the message plus noise in the exponent, which is brought "downstairs" by brute force to perform the rounding and recover the message.

Challenges in Proving Security. While the above sketch provides a construction template, security is far from obvious. Indeed, some thought reveals that the multi-input setting creates delicate attack scenarios that need care to handle. As an example, consider the strong security definition which allows the adversary to request ciphertexts that are decryptable by secret keys so long as they do not lead to a distinguishing attack. For simplicity, let us restrict to the setting where only the slot 1 ciphertext carries a message and slot 2 ciphertexts carry nothing except attributes (this restriction can be removed). Now, a slot 1 ciphertext may carry a message that depends on the challenger's secret bit as long as it is not decryptable by any key. However, slot 2 ciphertexts may participate in decryption with other slot 1 ciphertexts that do not encode the challenge bit, and decryption can (and does) lead to randomness leakage of participating ciphertexts. When such a "leaky" slot 2 ciphertext is combined with the challenge slot 1 ciphertext for decryption, security breaks down.

For concreteness, let us consider the setting where the adversary makes slot 1 ciphertext queries for $(\mathbf{x}_1, (m_0, m_1))$ and $(\mathbf{x}'_1, (m'_0, m'_1))$ and slot 2 ciphertext query for $(\mathbf{x}_2)$. Furthermore, the adversary makes a single key query for a circuit F such that $F(\mathbf{x}_1, \mathbf{x}_2) = 0$ (unauthorized) and $F(\mathbf{x}'_1, \mathbf{x}_2) = 1$ (authorized). Note that to prevent trivial attacks, we pose the restriction that $m'_0 = m'_1$, but we may have $m_0 \neq m_1$. In this setting, the $\mathsf{2ABE}$ construction described above is not secure since the noise associated with the slot 2 ciphertext for $\mathbf{x}_2$ leaks during decryption of the jointly generated ciphertext for $(\mathbf{x}'_1, \mathbf{x}_2)$ and this prevents using BGG^+ security for the pair $(\mathbf{x}_1, \mathbf{x}_2)$.

To resolve the above problem, we need to somehow "disconnect" randomness used in the challenge ciphertexts of slot 1 from randomness used in leaky/decrypting ciphertexts of other slots. This is tricky since the multi-input setting insists that ciphertexts be combined across slots in an unrestricted way. Fortunately, another technique developed [AY20] for a completely different reason comes to our assistance – we discontinue encoding the BGG^+ ciphertexts in $\mathsf{2ABE}$ ciphertexts for slot 2, so that even if a slot 2 ciphertext is decrypted, this does not affect the security of the BGG^+ encoding. Instead, we encode a binary "selection vector" in the exponent of $\mathbb{G}_2$, which enables the decryptor to recover $\psi_{2,x_{2,i}}$ when matching positions of slot 1 and slot 2 ciphertext components are paired. In the context of broadcast encryption (i.e. succinct ciphertext-policy ABE) [AY20] this trick was developed because the key generator could not know the randomness used by the encryptor, and moreover this randomness is unbounded across unbounded ciphertexts. In our setting, this trick instead allows to break the leakage of correlated randomness caused by combining ciphertexts across different slots, some of which may be challenge ciphertexts and others of which may be decrypting ciphertexts. However, though we made progress we are still not done and the formal security argument still be required to address several issues – please see Section 4 for more details.

Constructing $\mathsf{2ABE}$ in the Standard Model. We next turn to adapting the construction to the standard model – a natural starting point is the standard model adaptation of [AY20] by Agrawal, Wichs and Yamada [AWY20] which is based on a non-standard knowledge type assumption KOALA on bilinear groups. Our proof begins with these ideas but again departs significantly due to the nuanced security game of the multi-input setting – indeed, we will run into subtle technical issues related to the distribution of auxiliary information which will require us to formulate a variant of KOALA.

We first outline our construction, which uses a version of inner product functional encryption (IPFE), where one can directly encrypt group elements (rather than $\mathbb{Z}_q$ elements) and can generate a secret key for group elements. Thus, a ciphertext may encrypt a vector $[\mathbf{v}]_1$ and a secret key is for $[\mathbf{w}]_2$ and the decryption result of the ciphertext using the secret key is $[\langle \mathbf{v}, \mathbf{w} \rangle]_T$. Using IPFE and ideas similar to our first construction discussed above, we encode vectors into ciphertexts and secret keys so that the decryption result ends up with the BGG^+ ciphertext randomized by a secret key specific randomness t. In more detail, a slot 1 ciphertext is an IPFE ciphertext encoding $[\mathbf{v}, 0]_2$ and a slot 2 ciphertext is an IPFE secret key encoding $[t\mathbf{w}, 0]_2$ so that $[t\langle \mathbf{v}, \mathbf{w} \rangle]_T$ is recovered upon decryption, which is a corresponding BGG^+ ciphertext randomized by t on the exponent. Here, the last 0 entries are used for the security proof. We note that compared to the solution in bilinear generic group model we explained, we dropped the randomness on the ciphertext encoding and only the secret key encoding is randomized by t. The reason why the randomness on the ciphertext encoding can be removed is that the encoding is already protected by the IPFE and this change allows to simplify the construction and proof.

In the security game, we will have $\{\mathsf{ct}^{(i)} := \mathsf{IPFE.Enc}([\mathbf{v}^{(i)}, 0]_1)\}_i$ and $\{\mathsf{sk}^{(i)} := \mathsf{IPFE.sk}([t^{(i)}\mathbf{w}^{(i)}, 0]_2)\}_i$, where $\mathsf{ct}^{(i)}$ is the i-th slot 1 ciphertext and $\mathsf{sk}^{(i)}$ is the i-th slot 2 ciphertext. Let us say that the adversary requests Q ciphertexts in each slot. The security proof is by hybrid argument, where slot 1 ciphertexts are changed from ciphertexts for challenge bit 0 to 1 one by one. Now, to change the message in a slot 1 ciphertext i^*, we must account for its combination with *all* slot 2 ciphertexts – note that such a constraint does not arise in single input ABE/BE [AWY20]. To handle this, we leverage the power of IPFE so that the Q second slot ciphertexts hardcode the decryption value for the chosen slot 1 ciphertext i^* and behave as before with other slot 1 ciphertexts. A bit more explicitly, the j-th secret key may be hardwired with $([t^{(j)}]_2, [t^{(j)}\mathsf{BGG}^+.\mathsf{ct}^{(j)}]_2)$, where $\mathsf{BGG}^+.\mathsf{ct}^{(j)}$ is a set of BGG^+ ciphertexts derived from $\mathbf{v}^{(i^\star)}$ and $\mathbf{w}^{(j)}$. We note that since $\{\mathsf{BGG}^+.\mathsf{ct}^{(j)}\}_j$ are derived from the same vector $\mathbf{v}^{(i^\star)}$, their distribution is mutually correlated.

At this stage, we have a vector of BGG^+ ciphertexts encoded in the exponent, randomized with a unique random term $t^{(j)}$ and would like to change the ciphertexts $\mathsf{BGG}^+.\mathsf{ct}^{(j)}$ into random strings using the security of BGG^+. A similar situation was dealt with by [AWY20], who essentially showed that if $\mathsf{BGG}^+.\mathsf{ct}^{(j)}$ is individually pseudorandom given an auxiliary information aux, then by a variant of the KOALA assumption, $\{[t^{(j)}]_2, [t^{(j)}\mathsf{BGG}^+.\mathsf{ct}^{(j)}]_2\}_j$ looks pseudorandom, even if ciphertexts are mutually correlated for $j \in [Q]$. However, this idea is insufficient for our setting due to the distribution of the auxiliary information. In more detail, for the construction of [AWY20], it sufficed to have a single BGG^+ secret key in aux, since their construction only needed a single key secure BGG^+. By applying a standard trick in lattice cryptography, they could sample the secret key first (setting other parameters accordingly) and thus regard aux as a random string. In contrast, our scheme crucially requires multiple BGG^+ secret keys, which can no longer be considered as random strings. This necessitates formulating a variant of the KOALA assumption whose distribution of the auxiliary input is structured rather than random. We do not know how to weaken this assumption using our current techniques and leave this improvement as an interesting open problem. For more details, please see Section 5.

Compiling multi-input ABE to multi-input PE. Next, we discuss how to lift k-input miABE to k-input miPE. For the purposes of the introduction, let us focus on the case of $k = 2$. As a warm-up, we begin with the simpler setting of standard security, i.e. where there are no decrypting ciphertexts.

The natural first idea to construct miPE is to replace the single input ABE BGG^+ in our 2ABE scheme by single input PE, which has been constructed for all polynomial circuits by Gorbunov, Vaikuntanathan and Wee [GVW15]. However, this idea quickly runs into an insurmountable

hurdle – for our construction template, we need to bound the decryption noise by polynomial so that it can be recovered by brute force computation of discrete log in the final step. This is possible for ABE supporting NC_1 using asymmetric noise growth [GV15]. In the context of PE however, we do not know how to restrict the noise growth to polynomial – this is due to the usage of the fully homomorphic encryption in the scheme, which extends the depth of the evaluated circuit beyond what can be handled.

An alternative path to convert ABE to PE in the single input setting uses the machinery of *Lockable Obfuscation* (LO) [GKW17, WZ17]. Lockable obfuscation allows to obfuscate a circuit C with respect to a lock value β and a message m. The obfuscated circuit on input x outputs m if $C(x) = \beta$ and $\perp$ otherwise. For security, LO requires that if β has high entropy in the view of the adversary, the obfuscated circuit should be indistinguishable from a garbage program that does not carry any information.

Single to Multiple Inputs. The conversion in the single input setting is as follows. To encrypt a message m for an attribute $\mathbf{x}$, we first encrypt a random value β using the ABE to obtain an ABE cipheretxt ct. We then construct a circuit $C[\mathsf{ct}]$ that hardwires ct in it, takes as input an ABE secret key and decrypts the hardwired ciphertext using it. We obfuscate $C[\mathsf{ct}]$ with respect to the lock value β and the message m. The final PE ciphertext is the obfuscated circuit. It is easy to see that the PE scheme has correctness, since if the decryption is possible, β is recovered inside the obfuscated circuit and the lock is unlocked. By the correctness of LO, the message is revealed. In the security proof, we first change β encrypted inside ct to a zero string. This is possible using the security of ABE. Now the lock value β has high entropy from the view of the adversary. We then erase the information inside the obfuscated circuit, which includes the attribute information, using the security of LO.

Some thought reveals that the above conversion breaks down completely in the multi-input setting. For instance, if we apply the above conversion to a slot 1 ciphertext, the resulting obfuscation expects to receive slot 2 ciphertext in the clear. However, a slot 2 ciphertext of PE must also constitute an obfuscated circuit since otherwise the attribute associated with it will be leaked. But then there is no way to communicate between the two ciphertexts, both of which are obfuscated circuits!

To overcome this barrier, we develop a delicate *nested* approach which takes advantage of the fact that LO is powerful enough to handle general circuits. To restore communication between two ciphertexts while maintaining attribute privacy, we obfuscate a circuit for slot 1 that takes as input *another obfuscated circuit* for slot 2 and runs this inside itself. In more detail, the outer LO takes as input the "inner" LO circuit and the $\mathsf{2ABE}$ secret key $\mathsf{2ABE.sk}_f$. The inner LO instance encodes the circuit for $\mathsf{2ABE}$ decryption with the LO message as an SKE secret and the lock value as random tag β. It also has hardcoded in it the slot 2 $\mathsf{2ABE}$ ciphertext $\mathsf{2ABE.ct}_2$ with message β. The other piece of $\mathsf{2ABE}$, namely the slot 1 ciphertext $\mathsf{2ABE.ct}_1$ is hardwired in the outer LO. The outer LO encodes a circuit which runs the inner LO on inputs $\mathsf{2ABE.ct}_1$ and $\mathsf{2ABE.sk}_f$. By correctness of the inner LO, the $\mathsf{2ABE}$ decryption with $\mathsf{2ABE.ct}_1$, $\mathsf{2ABE.ct}_2$ and $\mathsf{2ABE.sk}_f$ is executed and if the functionality is satisfied, the inner LO outputs the SKE secret key. Thus, the SKE secret key signals whether the inner LO is unlocked, and if so, uses the recovered key to decrypt an SKE ciphertext which is hardcoded in the circuit. This ciphertext encrypts some random γ which is also set as the lock value of outer LO. If the SKE decryption succeeds, the lock value matches the decrypted value and outputs the message m which is the message in the outer LO. We note that the same SKE secret key must be used for both the inner and outer LO for them to effectively communicate.

Supporting Strong Security. This construction lends itself to a proof of security for the standard game where decrypting ciphertexts are not allowed, although via an intricate sequence of hybrids

especially for the case of general k. We refer the reader to Section 6 for details and turn our attention to the far more challenging case of strong security. In the setting of strong security, the proof fails – note that once any slot 2 ciphertext is decrypted, we no longer have the guarantee that the message value of the inner obfuscation is hidden. Since this message is a secret key of an SKE scheme and is used to encrypt the lock values for slot 1 ciphertexts, security breaks down once more.

To overcome this hurdle, we must make the construction more complex so that the message value of the inner obfuscation is no longer a global secret and does not compromise security even if revealed. To implement this intuition, we let the inner obfuscation output a slot 2 (strong) $\mathsf{2ABE}$ ciphertext when the lock is unlocked, which is then used to perform $\mathsf{2ABE}$ decryption in the circuit of the outer LO. Now, even if the security of a inner obfuscated circuit is compromised, this does not necessarily mean that the security of the entire system is compromised because of the guarantees of the strong security game of $\mathsf{2ABE}$. While oversimplified, this intuition may now be formalized into a proof. For more details, please see Section 7.

Constructing $\mathsf{3ABE}$ from Pairings and Lattices. Finally, we discuss our candidate construction for three input ABE scheme based on techniques developed by Brakerski and Vaikuntanathan [BV22] in conjunction with our $\mathsf{2ABE}$ construction in Section 4.1. The work of Brakerski and Vaikuntanathan [BV22] provided a clever candidate for succinct ciphertext-policy ABE for P from lattices. Their construction also uses decomposability in order to achieve succinctness which is the starting point for the multi-input setting as discussed above. Additionally, they provide novel ways to handle the lack of shared randomness between the key generator and encryptor – while [AY20] use pairings to generate shared randomness, [BV22] use lattice ideas and it is this part which makes their construction heuristic. Here, we show that the algebraic structure of their construction not only fits elegantly to the demands of the two-input setting, but can also be made compatible with our current $\mathsf{2ABE}$ construction to *amplify* arity to three! This surprising synergy between two completely different candidates of broadcast encryption, namely Agrawal-Yamada and Brakerski-Vaikuntanathan, created by decomposability and novel techniques of handling randomness, already provides an XWE of compression factor $1/4$ as against the previous best known $1/2$ [BJK+18], and may lead to other applications as well.

Recap of the Brakerski-Vaikuntanathan construction. To dig deeper into our construction, let us first recap the core ideas of [BV22]. First recall the well known fact that security of BGG^+ encodings is lost when we have two encodings for the same position encoding a different bit, namely, $\psi_{i,0} = \mathbf{s}\mathbf{B}_i + \mathbf{e}_{i,0}$ and $\psi_{i,1} = \mathbf{s}(\mathbf{B}_i + \mathbf{G}) + \mathbf{e}_{i,1}$, where $\mathbf{s}$ is a LWE secret, $\mathbf{B}_i$ is a matrix, and $\mathbf{e}_{1,b}$ is an error vector for $b \in \{0,1\}$. What [BV22] suggested is, if we augment BGG^+ encodings and mask them appropriately, then both encodings can be published and still hope to be secure. Namely, they change BGG^+ encodings to be $\psi_{i,b} = \mathbf{S}(\mathbf{B}_i + b\mathbf{G}) + \mathbf{E}_{i,b}$, where we replace the vector $\mathbf{s}$ with a matrix $\mathbf{S}$. They then mask the encodings by public (tall) matrices $\{\mathbf{C}_{i,b}\}_{i,b}$ as

$$\widehat{\psi}_{i,b} := \mathbf{C}_{i,b}\widehat{\mathbf{S}}_{i,b} + \mathbf{S}(\mathbf{B}_i + b\mathbf{G}) + \mathbf{E}_{i,b}$$

where $\{\widehat{\mathbf{S}}_{i,b}\}_{i,b}$ are random secret matrices. By releasing appropriate information, one can recover BGG^+ encodings with different LWE secrets. In more detail, we can publish a short vector $\mathbf{t}_\mathbf{x}$ for any binary string $\mathbf{x}$ that satisfies $\mathbf{t}_\mathbf{x}\mathbf{C}_{i,x_i} = \mathbf{0}$ (and $\mathbf{t}_\mathbf{x}\mathbf{C}_{i,1-x_i}$ is random) for all i. This allows us to compute

$$\mathbf{t}_\mathbf{x}\left(\mathbf{C}_{i,x_i}\widehat{\mathbf{S}}_{i,x_i} + \mathbf{S}(\mathbf{B}_i + x_i\mathbf{G}) + \mathbf{E}_{i,x_i}\right) = \mathbf{t}_\mathbf{x}\mathbf{S}(\mathbf{B}_i + x_i\mathbf{G}) + \mathbf{t}_\mathbf{x}\mathbf{E}_{i,x_i} = \mathbf{s}_\mathbf{x}(\mathbf{B}_i + x_i\mathbf{G}) + \mathbf{e}_{\mathbf{x},i,x_i}$$

where we set $\mathbf{s}_\mathbf{x} = \mathbf{t}_\mathbf{x}\mathbf{S}$ and $\mathbf{e}_{\mathbf{x},i,b} = \mathbf{t}_\mathbf{x}\mathbf{E}_{i,b}$. Namely, we can obtain BGG^+ samples specific to the string $\mathbf{x}$. This is similar to the idea of using pairings to choose the appropriate encoding based

on the attribute string, which is used in our two-input ABE with strong security. Similarly to that case, the obtained encodings are randomized by the user specific randomness. One of the heuristic aspects of [BV22] is that in order for their scheme to be secure, we have to assume that there is no meaningful way to combine the BGG^+ samples obtained from different vectors $\mathbf{t}_{\mathbf{x}}$ and $\mathbf{t}_{\mathbf{x}'}$.

Let us now adapt these techniques to provide a construction of two-input ABE. In our candidate, $\{\mathbf{B}_i\}_i$ and $\{\mathbf{C}_{i,b}\}_{i,b}$ matrices are made public.[4] An encryptor for the slot 1 computes for $i \in [\ell], b \in \{0,1\}$:

$$\left\{\psi_{i,x_{1,i}} := \mathbf{S}(\mathbf{B}_i + x_{1,i}\mathbf{G}) + \mathbf{E}_{i,x_{1,i}}\right\}_i, \left\{\widehat{\psi}_{i,b} := \mathbf{C}_{\ell+i,b}\widehat{\mathbf{S}}_{\ell+i,b} + \mathbf{S}(\mathbf{B}_{\ell+i} + b\mathbf{G}) + \mathbf{E}_{\ell+i,b}\right\}_{i,b}$$

where $x_{1,i}$ denotes the i-th bit of the attribute $\mathbf{x}_1$ for slot 1, ℓ denotes the length of an attribute, and $\mathbf{S}$ and $\widehat{\mathbf{S}}_{i,b}$ are freshly chosen by the encryptor. Intuitively, this is a partially stripped off version of the encodings in [BV22]. We believe this does not harm security, because the encryptor provides one out of two encodings for each position that is not masked by $\mathbf{C}_{i,b}\widehat{\mathbf{S}}_{i,b}$. The encryptor for slot 2 generates a vector $\mathbf{t}_{\mathbf{x}_2}$ such that $\mathbf{t}_{\mathbf{x}_2}\mathbf{C}_{i,x_{2,\ell+i}} = \mathbf{0}$ for all $i \in [\ell]$. The secret key for function F is simply BGG^+ secret key for the same function. In the decryption, the decryptor uses $\mathbf{t}_{\mathbf{x}_2}$ to choose BGG^+ encodings for attribute $\mathbf{x}_2$ from $\{\widehat{\psi}_{i,b}\}_{i,b}$. The obtained encodings are with respect to the LWE secret $\mathbf{t}_{\mathbf{x}}\mathbf{S}$. The decryptor can also choose BGG^+ encodings for attribute $\mathbf{x}_1$ from $\{\psi_i\}_i$. These obtained encodings constitutes a BGG^+ ciphertext for attribute $(\mathbf{x}_1, \mathbf{x}_2)$, which can be decrypted by the BGG^+ secret key. The intuition about security in [BV22] is that the BGG^+ encodings obtained by using $\mathbf{t}_{\mathbf{x}}$ vectors cannot be combined in a meaningful way due to the different randomness.

Amplifying Arity. We now amplify arity by leveraging the above techniques in conjunction with our pairing based construction. Our idea is to develop the scheme so that the decryptor can recover the above partially stripped off version of the encoding in the exponent from slot 1 and slot 2 ciphertexts by using the pairing operations, where the encodings may be randomized. Then, slot 3 ciphertext corresponds to a vector $\mathbf{t}_{\mathbf{x}_3}$, which annihilates $\mathbf{C}_{i,b}$ matrices for corresponding positions to the attribute $\mathbf{x}_3$. To do so, an encryptor for the first slot encodes

$$\{t_1\psi_{i,x_i}\}_{i\in[\ell]}, \quad \{t_1\psi_{i,b}\}_{i\in[\ell+1,2\ell],b\in\{0,1\}}, \quad \{t_1\widehat{\psi_{i,b}}\}_{i\in[2\ell+1,3\ell],b\in\{0,1\}}$$

of the exponent of $\mathbb{G}_1$, where t_1 is freshly chosen randomness by the encryptor. An encryptor for the second slot encodes t_2, $t_2\mathbf{d}_{\mathbf{x}_2}$ in the exponent of $\mathbb{G}_2$, where t_2 is freshly chosen randomness by the encryptor and $\mathbf{d}_{\mathbf{x}_2}$ is a selector vector that chooses $\psi_{i,x_{2,i}}$ out of $(\psi_{i,0}, \psi_{i,1})$ by the pairing operation. Concretely, $\mathbf{d}_{\mathbf{x}_2} = \{d_{i,b}\}_{i,b}$, where $d_{i,b} = 1$ if $b = x_{2,i}$ and 0 otherwise. These vectors are randomized by position-wise randomness as is the case for our other schemes. Finally, an encryptor for slot 3 with attribute $\mathbf{x}_3$ chooses $\mathbf{t}_{\mathbf{x}_3}$ such that $\mathbf{t}_{\mathbf{x}_3}\mathbf{C}_{2\ell+i,x_{3,i}} = \mathbf{0}$.

A somewhat worrying aspect of the candidate above may be that both $t_1\psi_{i,0}$ and $t_1\psi_{i,1}$ are encoded on $\mathbb{G}_1$. However, this is also the case for [AY20] and as in that work, these two encodings are randomized by the position-wise randomness and cannot be combined in a meaningful way (at least in the GGM). The only way to combine them is to take a pairing product with $\mathbb{G}_2$ elements. However, after the operation, we end up with partially stripped encoding that is randomized with t_1t_2. Therefore, a successful attack against the scheme may end up with attacking a partially stripped version of [BV22], which we expect to be as secure as the original scheme. Please see Section 8 for more details.

[4] The construction described here is simplified. For example, we omit the additional message carrying part in the construction, which is not necessary for the overview.

2 Preliminaries

Notation. We begin by defining the notation that we will use throughout this work. We use bold letters to denote vectors and the notation $[a, b]$ to denote the set of integers $\{k \in \mathbb{N} \mid a \leq k \leq b\}$. We use $[n]$ to denote the set $[1, n]$. For any vector $\mathbf{x}$ of length ℓ, we let x_i denote the i-th coordinate of $\mathbf{x}$, for $i \in [\ell]$. We use $\mathbf{1}_{\ell \times m}$ (resp. $\mathbf{0}_{\ell \times m}$) to represent a matrix of dimensions $\ell \times m$ having each entry as 1 (resp. 0). Similarly, we write $\mathbf{1}_a$ (resp. $\mathbf{0}_a$) to represent $(1, \ldots, 1) \in \mathbb{Z}_q^a$ ($(0, \ldots, 0) \in \mathbb{Z}_q^a$) We say a function $f(n)$ is *negligible* if it is $O(n^{-c})$ for all $c > 0$, and we use $\mathrm{negl}(n)$ to denote a negligible function of n. We say $f(n)$ is *polynomial* if it is $O(n^c)$ for some constant $c > 0$, and we use $\mathrm{poly}(n)$ to denote a polynomial function of n. We use the abbreviation PPT for probabilistic polynomial-time. The function $\log x$ is the base 2 logarithm of x. For two distributions $\mathcal{D}_1$, $\mathcal{D}_2$ we use the notation $\mathcal{D}_1 \approx_c \mathcal{D}_2$ to denote that a PPT adversary cannot distinguish between the distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ except only with negligible distinguishing advantage.

2.1 Single User Attribute Based Encryption

For ease of readability, we define single user cpABE and kpABE below.

Let $R = \{R_\lambda : A_\lambda \times B_\lambda \to \{0,1\}\}_\lambda$ be a relation where A_λ and B_λ denote "ciphertext attribute" and "key attribute" spaces. An attribute-based encryption (ABE) scheme for R is defined by the following PPT algorithms:

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

$\mathsf{Enc}(\mathsf{mpk}, X, \mu) \to \mathsf{ct}$: The encryption algorithm takes as input a master public key mpk, a ciphertext attribute $X \in A_\lambda$, and a message bit μ. It outputs a ciphertext ct.

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

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

Definition 2.1 (Correctness)**.** An ABE scheme for relation family R is correct if for all $\lambda \in \mathbb{N}$, $X \in A_\lambda, Y \in B_\lambda$ such that $R(X, Y) = 1$, and for all messages $\mu \in \mathsf{msg}$,

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

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

Definition 2.2 ($\mathsf{Ada\text{-}IND}$ security for ABE)**.** For an ABE scheme $\mathsf{ABE} = \{\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec}\}$ for a relation family $R = \{R_\lambda : A_\lambda \times B_\lambda \to \{0,1\}\}_\lambda$ and a message space $\{\mathsf{msg}_\lambda\}_{\lambda \in \mathbb{N}}$ and an adversary $\mathcal{A}$, let us define $\mathsf{Ada\text{-}IND}$ security game as follows.

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

2. **Query phase:** During the game, $\mathcal{A}$ adaptively makes the following queries, in an arbitrary order. $\mathcal{A}$ can make unbounded many key queries, but can make only single challenge query.

(a) **Key Queries:** $\mathcal{A}$ chooses an input $Y \in B_\lambda$. For each such query, the challenger replies with $\mathsf{sk}_Y \leftarrow \mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, Y)$.

(b) **Challenge Query:** At some point, $\mathcal{A}$ submits a pair of equal length messages $(\mu_0, \mu_1) \in (\mathsf{msg})^2$ and the target $X^\star \in A_\lambda$ to the challenger. The challenger samples a random bit $b \leftarrow \{0,1\}$ and replies to $\mathcal{A}$ with $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, X^\star, \mu_b)$.

We require that $R(X^\star, Y) = 0$ holds for any Y such that $\mathcal{A}$ makes a key query for Y in order to avoid trivial attacks.

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

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

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

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

We can consider the following stronger version of the security where we require the ciphertext to be pseudorandom.

Definition 2.3 ($\mathsf{Ada\text{-}INDr}$ security for ABE)**.** We define $\mathsf{Ada\text{-}INDr}$ security game similarly to $\mathsf{Ada\text{-}IND}$ security game except that the adversary $\mathcal{A}$ chooses single message μ instead of (μ_0, μ_1) at the challenge phase and the challenger returns $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, X^\star, \mu)$ if $b = 0$ and a random ciphertext $\mathsf{ct} \leftarrow \mathcal{CT}$ from a ciphertext space $\mathcal{CT}$ if $b = 1$. We define the advantage $\mathsf{Adv}^{\mathsf{Ada\text{-}INDr}}_{\mathsf{ABE},\mathcal{A}}(1^\lambda)$ of the adversary $\mathcal{A}$ accordingly and say that the scheme satisfies $\mathsf{Ada\text{-}INDr}$ security if the quantity is negligible.

We also consider (weaker) selective versions of the above notions, where $\mathcal{A}$ specifies its target $X^\star$ at the beginning of the game.

Definition 2.4 ($\mathsf{Sel\text{-}IND}$ security for ABE)**.** We define $\mathsf{Sel\text{-}IND}$ security game as $\mathsf{Ada\text{-}IND}$ security game with the exception that the adversary $\mathcal{A}$ has to choose the challenge ciphertext attribute $X^\star$ before the setup phase but key queries $Y_1, Y_2, \ldots$ and choice of (μ_0, μ_1) can still be adaptive. We define the advantage $\mathsf{Adv}^{\mathsf{Sel\text{-}IND}}_{\mathsf{ABE},\mathcal{A}}(1^\lambda)$ of the adversary $\mathcal{A}$ accordingly and say that the scheme satisfies $\mathsf{Sel\text{-}IND}$ security (or simply *selective security*) if the quantity is negligible.

Definition 2.5 ($\mathsf{Sel\text{-}INDr}$ security for ABE)**.** We define $\mathsf{Sel\text{-}INDr}$ security game as $\mathsf{Ada\text{-}INDr}$ security game with the exception that the adversary $\mathcal{A}$ has to choose the challenge ciphertext attribute $X^\star$ before the setup phase but key queries $Y_1, Y_2, \ldots$ and choice of μ can still be adaptive. We define the advantage $\mathsf{Adv}^{\mathsf{Sel\text{-}INDr}}_{\mathsf{ABE},\mathcal{A}}(1^\lambda)$ of the adversary $\mathcal{A}$ accordingly and say that the scheme satisfies $\mathsf{Sel\text{-}INDr}$ security if the quantity is negligible.

In the following, we recall definitions of various ABEs by specifying the relation. We start with the standard notions of ciphertext-policy attribute-based encryption (cpABE) and key-policy attribute-based encryption (kpABE).

cpABE for circuits. We define cpABE for circuit class $\{\mathcal{C}_\lambda\}_\lambda$ by specifying the relation. Here, $\mathcal{C}_\lambda$ is a set of circuits with input length $\ell(\lambda)$ and binary output. We define $A^{\mathsf{CP}}_\lambda = \mathcal{C}_\lambda$ and $B^{\mathsf{CP}}_\lambda = \{0,1\}^\ell$. Furthermore, we define the relation R^{CP}_λ as $R^{\mathsf{CP}}_\lambda(C, \mathbf{x}) = \neg C(\mathbf{x})$.[5]

[5]Here, we follow the standard convention in lattice-based cryptography where the decryption succeeds when $C(\mathbf{x}) = 0$ rather than $C(\mathbf{x}) = 1$.

kpABE for circuits. To define kpABE for circuits, we simply swap key and ciphertext attributes in cpABE for circuits. More formally, to define kpABE for circuits, we define $A_\lambda^{\mathsf{KP}} = \{0,1\}^\ell$ and $B_\lambda^{\mathsf{KP}} = \mathcal{C}_\lambda$. We also define $R_\lambda^{\mathsf{KP}} : A_\lambda^{\mathsf{KP}} \times B_\lambda^{\mathsf{KP}} \to \{0,1\}$ as $R_\lambda^{\mathsf{KP}}(\mathbf{x}, C) = \neg C(\mathbf{x})$.

Remark 2.6. We observe that the symmetric key variants of the above definitions can be easily obtained by letting the encryptor have access to the master secret key and permitting the adversary to make ciphertext requests in the security game.

2.2 Lockable Obfuscation

We define lockable obfuscation [GKW17, WZ17] below. Let n, m, d be polynomials, and $\mathcal{C}_{n,m,d}(\lambda)$ be the class of depth $d(\lambda)$ circuits with $n(\lambda)$ bit input and $m(\lambda)$ bit output. A lockable obfuscator for $\mathcal{C}_{n,m,d}$ consists of algorithms Obf and Eval with the following syntax. Let $\mathcal{M}$ be the message space.

$\mathsf{Obf}(1^\lambda, P, \mathsf{msg}, \alpha) \to \widetilde{P}$: The obfuscation algorithm is a randomized algorithm that takes as input the security parameter λ, a program $P \in \mathcal{C}_{n,m,d}$, message $\mathsf{msg} \in \mathcal{M}$ and 'lock string' $\alpha \in \{0,1\}^{m(\lambda)}$. It outputs a program $\widetilde{P}$.

$\mathsf{Eval}(\widetilde{P}, x) \to y \in \mathcal{M} \cup \{\bot\}$. The evaluator is a deterministic algorithm that takes as input a program $\widetilde{P}$ and a string $x \in \{0,1\}^{n(\lambda)}$. It outputs $y \in \mathcal{M} \cup \{\bot\}$.

For correctness, it is required that if $P(x) = \alpha$, then the obfuscated program $\widetilde{P} \leftarrow \mathsf{Obf}(1^\lambda, P, \mathsf{msg}, \alpha)$, evaluated on input x, outputs msg, and if $P(x) \neq \alpha$, then $\widetilde{P}$ outputs $\bot$ on input x.

Definition 2.7 (Perfect Correctness)**.** Let n, m, d be polynomials. A lockable obfuscation scheme for $\mathcal{C}_{n,m,d}$ and message space $\mathcal{M}$ is said to be perfectly correct if it satisfies the following properties:

1. For all security parameters λ, inputs $x \in \{0,1\}^{n(\lambda)}$, programs $P \in \mathcal{C}_{n,m,d}$ and messages $\mathsf{msg} \in \mathcal{M}$, if $P(x) = \alpha$, then
$$\mathsf{Eval}(\mathsf{Obf}(1^\lambda, P, \mathsf{msg}, \alpha), x) = \mathsf{msg}.$$
2. For all security parameters λ, inputs $x \in \{0,1\}^{n(\lambda)}$, programs $P \in \mathcal{C}_{n,m,d}$ and messages $\mathsf{msg} \in \mathcal{M}$, if $P(x) \neq \alpha$, then
$$\mathsf{Eval}(\mathsf{Obf}(1^\lambda, P, \mathsf{msg}, \alpha), x) = \bot .$$

Definition 2.8 (Security)**.** Let n, m, d be polynomials. A lockable obfuscation scheme $(\mathsf{Obf}, \mathsf{Eval})$ for $\mathcal{C}_{n,m,d}$ and message space $\mathcal{M}$ is said to be secure if there exists a PPT simulator Sim such that for all PPT adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, there exists a negligible function $\mathrm{negl}(\cdot)$ such that:

$$\left| \Pr\left[\mathcal{A}_1(\widetilde{P}_b, \mathsf{st}) = b \;\middle|\; \begin{array}{r} (P, \mathsf{msg}, \mathsf{st}) \leftarrow \mathcal{A}_0(1^\lambda) \\ b \leftarrow \{0,1\}, \alpha \leftarrow \{0,1\}^{m(\lambda)} \\ \widetilde{P}_0 \leftarrow \mathsf{Obf}(1^\lambda, P, \mathsf{msg}, \alpha) \\ \widetilde{P}_1 \leftarrow \mathsf{Sim}(1^\lambda, 1^{|P|}, 1^{|\mathsf{msg}|}) \end{array} \right] - \frac{1}{2} \right| \leq \mathrm{negl}(\lambda).$$

Analogously, we can define the security for multiple queries case.

Definition 2.9 (LO security with multiple queries)**.** Let n, m, d be polynomials. A lockable obfuscation scheme $(\mathsf{Obf}, \mathsf{Eval})$ for $\mathcal{C}_{n,m,d}$ and message space $\mathcal{M}$ is said to be secure (for multiple adaptive queries) if there exists a PPT simulator Sim such that for all PPT adversaries $\mathcal{A}$, the probability of winning in the following game is $1/2 + \mathrm{negl}(\lambda)$.

The security game between challenger $\mathcal{C}$ and adversary $\mathcal{A}$ is defined as follows:

1. $\mathcal{C}$ Samples a bit $b \leftarrow \{0,1\}$.
2. $\mathcal{A}$ issues $p = p(\lambda)$ adaptive queries of the form (P^i, msg^i) to $\mathcal{C}$.
3. For each query, $\mathcal{C}$ returns $\widetilde{P}^i_b$, where
$$\widetilde{P}^i_0 \leftarrow \mathsf{Obf}(1^\lambda, P^i, \mathsf{msg}^i, \alpha^i), \alpha_i \leftarrow \{0,1\}^{m(\lambda)} \text{ and } \widetilde{P}^i_1 \leftarrow \mathsf{Sim}(1^\lambda, 1^{|P^i|}, 1^{|\mathsf{msg}^i|})$$
4. In the end, the adversary outputs a bit b'.

The adversary wins if $b' = b$.

Reduction from multi-queries definition to single query can be shown using hybrids. We sketch the reduction here. We consider $p+1$ hybrids $\mathbf{Game}_0$ to $\mathbf{Game}_p$. In $\mathbf{Game}_i$, first i programs are simulated programs and remaining $p - i$ are obfuscated programs. Indisntinguishability of $\mathbf{Game}_i$ and $\mathbf{Game}_{i+1}$ follows from the security in case of single query.

2.3 Batch Inner Product Functional Encryption

We define batch inner product functional encryption (BIPFE) in the secret key setting. This is a straightforward extension of the standard notion of the IPFE in the secret key setting [BJK15, DDM16, LV16] and is introduced for the purpose of describing our scheme with notational ease. In BIPFE, a ciphertext and a secret key are associated with matrices of the same size consisting of group components $[\mathbf{V}]_1 = [(\mathbf{v}_1^\top, \ldots, \mathbf{v}_n^\top)]_1 \in \mathbb{G}_1^{B \times n}$ and $[\mathbf{W}]_2 = [(\mathbf{w}_1^\top, \ldots, \mathbf{w}_n^\top)]_2 \in \mathbb{G}_2^{B \times n}$, respectively. Here, we refer to B as the batch size and n as the dimension. Upon decryption, the following is recovered

$$[\mathbf{V} \boxdot \mathbf{W}]_T := \left[\sum_{i \in [n]} \mathbf{v}_i \odot \mathbf{w}_i \right]_T .$$

Namely, we recover inner product of each row of $\mathbf{V}$ and $\mathbf{W}$ in parallel as a decryption result. More formal definition follows.

Let $\mathsf{GroupGen}$ be a group generator that outputs bilinear group $\mathbb{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, [1]_1, [1]_2)$. A BIPFE scheme based on $\mathbb{G}$ consists of 4 efficient algorithms:

$\mathsf{Setup}(1^\lambda, 1^B, 1^n) \to \mathsf{msk}$: The setup algorithm takes as input the security parameter, the batch size B, the dimension n all in unary and outputs master secret key msk.

$\mathsf{KeyGen}(\mathsf{msk}, [\mathbf{W}]_2) \to \mathsf{sk}_{\mathbf{W}}$: The key generation algorithm takes as input the master secret key and a matrix of group elements $[\mathbf{W}]_2 \in \mathbb{G}_2$, and outputs a secret key $\mathsf{sk}_{\mathbf{W}}$.

$\mathsf{Enc}(\mathsf{msk}, [\mathbf{V}]_1) \to \mathsf{ct}_{\mathbf{V}}$: The encryption algorithm takes as input the master secret key and a matrix of group elements $[\mathbf{V}]_1$ and outputs a ciphertext $\mathsf{ct}_{\mathbf{V}}$.

$\mathsf{Dec}(\mathsf{sk}_{\mathbf{W}}, \mathsf{ct}_{\mathbf{V}}) \to [\mathbf{Z}]_T \vee \perp$: The decryption algorithm takes as input a secret key $\mathsf{sk}_{\mathbf{W}}$ and a ciphertext $\mathsf{ct}_{\mathbf{V}}$, and outputs an element $[\mathbf{Z}]_T \in \mathbb{G}_T$ or $\perp$.

Correctness. We say the BIPFE scheme satisfies decryption correctness if for all $\lambda \in \mathbb{N}$, all batch size B, all dimension n, and all matrices $\mathbf{V}, \mathbf{W} \in \mathbb{Z}_p^{B \times n}$,

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}_{\mathbf{W}}, \mathsf{ct}_{\mathbf{V}}) = [\mathbf{W} \boxdot \mathbf{V}]_T \;\middle|\; \begin{array}{c} \mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda, 1^B, 1^n) \\ \mathsf{sk}_{\mathbf{W}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, [\mathbf{W}]_2) \\ \mathsf{ct}_{\mathbf{V}} \leftarrow \mathsf{Enc}(\mathsf{msk}, [\mathbf{V}]_1) \end{array}\right] = 1 \ .$$

Next, we define the function hiding property.

Definition 2.10 (Function Hiding Security)**.** Let $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a BIPFE scheme as defined above. The scheme is function hiding if $\mathsf{Exp}^0_{\mathsf{FH}}$ is indistinguishable from $\mathsf{Exp}^1_{\mathsf{FH}}$ for all PPT adversary $\mathcal{A}$ where $\mathsf{Exp}^b_{\mathsf{FH}}$ for $b \in \{0, 1\}$ is defined as follows:

1. **Setup**: Run the adversary $\mathcal{A}$ on input 1^λ to obtain the batch size 1^B and the dimension 1^n from $\mathcal{A}$. Let $\mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda, 1^B, 1^n)$ and return msk to $\mathcal{A}$.

2. **Challenge**: Repeat the following for arbitrarily many rounds determined by $\mathcal{A}$: In each round, $\mathcal{A}$ has 2 options:

   - $\mathcal{A}$ submits $[\mathbf{W}_0^{(i)}]_2, [\mathbf{W}_1^{(i)}]_2 \in \mathbb{G}_2^{B \times n}$ as a secret key query. Upon receiving this, compute $\mathsf{sk}^{(i)} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, [\mathbf{W}_b^{(i)}]_2)$ and return this to $\mathcal{A}$.
   - $\mathcal{A}$ submits $[\mathbf{V}_0^{(i)}]_1, [\mathbf{V}_1^{(i)}]_1 \in \mathbb{G}_1^{B \times n}$ as an encryption query. Upon receiving this, compute $\mathsf{ct}^{(i)} \leftarrow \mathsf{Enc}(\mathsf{msk}, [\mathbf{V}_b^{(i)}]_1)$ and return this to $\mathcal{A}$.

3. **Guess**: $\mathcal{A}$ outputs its guess b'.

The adversary is called admissible if $\mathbf{V}_0^{(i)} \boxdot \mathbf{W}_0^{(j)} = \mathbf{V}_1^{(i)} \boxdot \mathbf{W}_1^{(j)}$ for all combinations of i and j. We say that the BIPFE scheme is function hiding if $|\Pr[b = b'] - 1/2|$ is negligible for all admissible PPT adversaries.

Note that function hiding IPFE is captured as a special case of our notion of BIPFE with the batch size $B = 1$. It can be seen that function hiding IPFE can be converted to BIPFE by running the former in parallel for B times. Function hiding IPFE schemes are constructed from various assumptions including SXDH and DLIN [DDM16, LV16] and thus BIPFE can be constructed from the same assumptions.

2.4 Lattice Preliminaries

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

Learning with Errors. We introduce the learning with errors (LWE) problem.

Definition 2.11 (The LWE Assumption)**.** Let $n = n(\lambda)$, $m = m(\lambda)$, and $q = q(\lambda) > 2$ be integers and $\chi = \chi(\lambda)$ be a distribution over $\mathbb{Z}_q$. We say that the $\mathsf{LWE}(n, m, q, \chi)$ hardness assumption holds if for any PPT adversary $\mathcal{A}$ we have

$$|\Pr[\mathcal{A}(\mathbf{A}, \mathbf{s}\mathbf{A} + \mathbf{x}) \to 1] - \Pr[\mathcal{A}(\mathbf{A}, \mathbf{v}) \to 1]| \leq \mathrm{negl}(\lambda)$$

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

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

Trapdoors. Let us consider a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$. For all $\mathbf{V} \in \mathbb{Z}_q^{n \times m'}$, we let $\mathbf{A}_\gamma^{-1}(\mathbf{V})$ be an output distribution of $\mathsf{SampZ}(\gamma)^{m \times m'}$ conditioned on $\mathbf{A} \cdot \mathbf{A}_\gamma^{-1}(\mathbf{V}) = \mathbf{V}$. A γ-trapdoor for $\mathbf{A}$ is a trapdoor that enables one to sample from the distribution $\mathbf{A}_\gamma^{-1}(\mathbf{V})$ in time $\mathrm{poly}(n, m, m', \log q)$ for any $\mathbf{V}$. We slightly overload notation and denote a γ-trapdoor for $\mathbf{A}$ by $\mathbf{A}_\gamma^{-1}$. We also define the special gadget matrix $\mathbf{G} \in \mathbb{Z}_q^{n \times m}$ as the matrix obtained by padding $\mathbf{I}_n \otimes (1, 2, 4, 8, \ldots, 2^{\lceil \log q \rceil})$ with zero-columns. The following properties had been established in a long sequence of works [GPV08, CHKP10, ABB10a, ABB10b, MP12, BLP+13].

Lemma 2.12 (Properties of Trapdoors)**.** *Lattice trapdoors exhibit the following properties.*

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

Lattice Evaluation. The following is an abstraction of the evaluation procedure in previous LWE based FHE and ABE schemes. We follow the presentation by Tsabary [Tsa19], but with different parameters.

Lemma 2.13 (Fully Homomorphic Computation [GV15])**.** *There exists a pair of deterministic algorithms $(\mathsf{EvalF}, \mathsf{EvalFX})$ with the following properties.*

- $\mathsf{EvalF}(\mathbf{B}, F) \to \mathbf{H}_F$. *Here, $\mathbf{B} \in \mathbb{Z}_q^{n \times m\ell}$ and $F : \{0,1\}^\ell \to \{0,1\}$ is a circuit.*
- $\mathsf{EvalFX}(F, \mathbf{x}, \mathbf{B}) \to \widehat{\mathbf{H}}_{F,\mathbf{x}}$. *Here, $\mathbf{x} \in \{0,1\}^\ell$ and $F : \{0,1\}^\ell \to \{0,1\}$ is a circuit with depth d. We have*
$$[\mathbf{B} - \mathbf{x} \otimes \mathbf{G}]\widehat{\mathbf{H}}_{F,\mathbf{x}} = \mathbf{B}\mathbf{H}_F - F(\mathbf{x})\mathbf{G} \mod q,$$
where we denote $[x_1\mathbf{G}\| \cdots \|x_k\mathbf{G}]$ by $\mathbf{x} \otimes \mathbf{G}$. Furthermore, we have
$$\|\mathbf{H}_F\|_\infty \leq m \cdot 2^{O(d)}, \quad \|\widehat{\mathbf{H}}_{F,\mathbf{x}}\|_\infty \leq m \cdot 2^{O(d)}.$$
- *The running time of $(\mathsf{EvalF}, \mathsf{EvalFX})$ is bounded by $\mathrm{poly}(n, m, \log q, 2^d)$.*

The above algorithms are taken from [GV15], which is a variant of similar algorithms proposed by Boneh et al. [BGG+14]. The algorithms in [BGG+14] work for any polynomial-sized circuit F, but $\|\mathbf{H}_F\|_\infty$ and $\|\mathbf{H}_{F,\mathbf{x}}\|_\infty$ become super-polynomial even if the depth of the circuit is shallow (i.e., logarithmic depth). On the other hand, the above algorithms run in polynomial time only when F is of logarithmic depth, but $\|\mathbf{H}_F\|_\infty$ and $\|\mathbf{H}_{F,\mathbf{x}}\|_\infty$ can be polynomially bounded. The latter property is crucial for our purpose.

Modified Noise Distribution. For a distribution χ over $\mathbb{Z}$ and an integer m, we define $\widetilde{\chi^m}$ as follows. To sample from $\widetilde{\chi^m}$, we first sample $\mathbf{x} \leftarrow \chi^m$ and $\mathbf{S} \leftarrow \{-1, 1\}^{m \times m}$ and output $\mathbf{S}\mathbf{x}$. By triangular inequality, it can be seen that if the absolute value of a sample from χ is always bounded by B, the infinity norm of a sample from $\widetilde{\chi^m}$ is always bounded mB. This modified noise distribution is used in the kpABE scheme by Boneh et al. [BGG+14] described in Sec. 2.5 for the case of χ being the discrete Gaussian distribution. The modification of the noise is introduced in order to make the security proof work. We refer to their paper for the details.

2.5 kpABE Scheme by Boneh et al. [BGG+14]

We will use a variant of the kpABE scheme proposed by Boneh et al. [BGG+14]. We call the scheme BGG^+ and provide the description of the scheme in the following. We focus on the case where the policies associated with secret keys are limited to circuits with logarithmic depth rather than arbitrary polynomially bounded depth, so that we can use the evaluation algorithm due to Gorbunov and Vinayagamurthy [GV15] (see Lemma 2.13). This allows us to bound the noise growth during the decryption by a polynomial factor, which is crucial for us as in [AY20].

The scheme supports the circuit class $\mathcal{C}_{\ell(\lambda),d(\lambda)}$, which is a set of all circuits with input length $\ell(\lambda)$ and depth at most $d(\lambda)$ with arbitrary $\ell(\lambda) = \mathrm{poly}(\lambda)$ and $d(\lambda) = O(\log \lambda)$.

$\mathsf{Setup}(1^\lambda)$: On input 1^λ, the setup algorithm defines the parameters $n = n(\lambda)$, $m = m(\lambda)$, noise distributions χ over $\mathbb{Z}$, $\tau_0 = \tau_0(\lambda)$, $\tau = \tau(\lambda)$, and $B = B(\lambda)$ as specified later. It then proceeds as follows.

1. Sample $(\mathbf{A}, \mathbf{A}_{\tau_0}^{-1}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$ such that $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$.
2. Sample random matrix $\mathbf{B} = (\mathbf{B}_1, \dots, \mathbf{B}_\ell) \leftarrow (\mathbb{Z}_q^{n\times m})^\ell$ and a random vector $\mathbf{u} \leftarrow \mathbb{Z}_q^n$.
3. Output the master public key $\mathsf{mpk} = (\mathbf{A}, \mathbf{B}, \mathbf{u})$ and the master secret key $\mathsf{msk} = \mathbf{A}_{\tau_0}^{-1}$.

$\mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, F)$: The key generation algorithm takes as input the master public key mpk, the master secret key msk, and a circuit $F \in \mathcal{C}_{\ell,d}$ and proceeds as follows.

1. Compute $\mathbf{H}_F = \mathsf{EvalF}(\mathbf{B}, F)$ and $\mathbf{B}_F = \mathbf{B}\mathbf{H}_F$.
2. Compute $[\mathbf{A}\|\mathbf{B}_F]_\tau^{-1}$ from $\mathbf{A}_{\tau_0}^{-1}$ and sample $\mathbf{r} \in \mathbb{Z}^{2m}$ as $\mathbf{r}^\top \leftarrow [\mathbf{A}\|\mathbf{B}_F]_\tau^{-1}(\mathbf{u}^\top)$.
3. Output the secret key $\mathsf{sk}_F := \mathbf{r}$.

$\mathsf{Enc}(\mathsf{mpk}, \mathbf{x}, \mu)$: The encryption algorithm takes as input the master public key mpk, an attribute $\mathbf{x} \in \{0,1\}^\ell$, and a message $\mu \in \{0,1\}$ and proceeds as follows.

1. Sample $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, $e_0 \leftarrow \chi$, $\mathbf{e} \leftarrow \chi^m$, and $\mathbf{e}_{i,b} \leftarrow \widetilde{\chi^m}$ for $i \in [\ell]$ and $b \in \{0,1\}$, where $\widetilde{\chi^m}$ is defined as in Sec. 2.4 from χ.
2. Compute

$$\text{For all } i \in [\ell],\ b \in \{0,1\}, \psi_{i,b} := \mathbf{s}(\mathbf{B}_i - b\mathbf{G}) + \mathbf{e}_{i,b} \in \mathbb{Z}_q^m$$
$$\psi_{2\ell+1} := \mathbf{s}\mathbf{A} + \mathbf{e} \in \mathbb{Z}_q^m,\ \psi_{2\ell+2} := \mathbf{s}\mathbf{u}^\top + e_0 + \mu\lceil q/2\rceil \in \mathbb{Z}_q,$$

3. Output the ciphertext $\mathsf{ct}_\mathbf{x} := (\{\psi_{i,x_i}\}_{i\in[\ell]}, \psi_{2\ell+1}, \psi_{2\ell+2})$, where x_i is the i-th bit of $\mathbf{x}$.

$\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_\mathbf{x}, \mathsf{ct}_F)$: The decryption algorithm takes as input the master public key mpk, a secret key sk_F for a circuit F, and a ciphertext $\mathsf{ct}_\mathbf{x}$ for an attribute $\mathbf{x}$ and proceeds as follows.

1. Parse $\mathsf{ct}_\mathbf{x} \to (\{\psi_{i,x_i} \in \mathbb{Z}_q^m\}_{i\in[\ell]}, \psi_{2\ell+1} \in \mathbb{Z}_q^m, \psi_{2\ell+2} \in \mathbb{Z}_q)$, and $\mathsf{sk}_F \in \mathbb{Z}^{2m}$. If any of the component is not in the corresponding domain or $F(\mathbf{x}) = 1$, output $\perp$.
2. Compute $\widehat{\mathbf{H}}_{F,\mathbf{x}} = \mathsf{EvalF}(F, \mathbf{x}, \mathbf{B})$.
3. Concatenate $\{\psi_{i,x_i}\}_{i\in[\ell]}$ to form $\psi_\mathbf{x} = (\psi_{1,x_1}, \dots, \psi_{\ell,x_\ell})$.
4. Compute

$$\psi' := \psi_{2\ell+2} - [\psi_{2\ell+1}\|\psi_\mathbf{x}\widehat{\mathbf{H}}_{F,\mathbf{x}}]\mathbf{r}^\top.$$

5. Output 0 if $\psi' \in [-B, B]$ and 1 if $[-B + \lceil q/2\rceil, B + \lceil q/2\rceil]$.

Remark 2.14. We note that the encryption algorithm above computes redundant components $\{\psi_{i,\neg x_i}\}_{i\in[\ell]}$ in the second step, which are discarded in the third step. However, due to this redundancy, the scheme has the following special structure that will be useful for us. Namely, the first and the second steps of the encryption algorithm can be executed without knowing $\mathbf{x}$. Only the third step of the encryption algorithm needs the information of $\mathbf{x}$, where it chooses $\{\psi_{i,x_i}\}_{i\in[\ell]}$ from $\{\psi_{i,b}\}_{i\in[\ell],b\in\{0,1\}}$ depending on each bit of $\mathbf{x}$ and then output the former terms along with $\psi_{2\ell+1}$ and $\psi_{2\ell+2}$.

Parameters and Security. We choose the parameters for the scheme as follows:

$$m = n^{1.1}\log q, \qquad q = 2^{\Theta(\lambda)}, \qquad \chi = \mathsf{SampZ}(3\sqrt{n}),$$
$$\tau_0 = n\log q \log m, \qquad \tau = m^{3.1}\ell\cdot 2^{O(d)} \qquad B = \ell n^2 m^5 \tau\cdot 2^{O(d)}.$$

The parameter n will be chosen depending on whether we need Sel-INDr security or Ada-INDr security for the scheme. If it suffices to have Sel-INDr security, we set $n = \lambda^c$ for some constant $c > 1$. If we need Ada-INDr security, we have to enlarge the parameter to be $n = (\ell\lambda)^c$ in order to compensate for the security loss caused by the complexity leveraging.

We remark that if we were to use the above ABE scheme stand-alone, we would have been able to set q polynomially bounded as in [GV15]. The reason why we set q exponentially large is that we combine the scheme with bilinear maps of order q to lift the ciphertext components to the exponent so that they are "hidden" in some sense. In order to use the security of the bilinear map, we set the group order q to be exponentially large.

The following theorem summarizes the security and efficiency properties of the construction. There are two parameter settings depending on whether we assume subexponential hardness of LWE or not.

Theorem 2.15 (Adapted from [GV15, BGG+14])**.** *Assuming hardness of* $\mathsf{LWE}(n, m, q, \chi)$ *with* $\chi = \mathsf{SampZ}(3\sqrt{n})$ *and* $q = O(2^{n^{1/\epsilon}})$ *for some constant* $\epsilon > 1$*, the above scheme satisfies* Sel-INDr *security (Definition 2.5). Assuming* subexponential *hardness of* $\mathsf{LWE}(n, m, q, \chi)$ *with the same parameters, the above scheme satisfies* Ada-INDr *security (Definition 2.3) with respect to the ciphertext space* $\mathcal{CT} := \mathbb{Z}_q^{m(\ell+1)+1}$

2.6 Bilinear Map Preliminaries

Here, we introduce our notation for bilinear maps and the bilinear generic group model following Agrawal and Yamada [AY20], which in turn is based on [BCFG17, BFF+14] for defining generic k-linear groups to the bilinear group settings. The definition closely follows that of Maurer [Mau05], which is equivalent to the alternative formulation by Shoup [Sho97].

Notation on Bilinear Maps. A bilinear group generator GroupGen takes as input 1^λ and outputs a group description $\mathbb{G} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$, where q is a prime of $\Theta(\lambda)$ bits, $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are cyclic groups of order q, $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a non-degenerate bilinear map, and g_1 and g_2 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. We require that the group operations in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ as well as the bilinear map e can be efficiently computed. We employ the implicit representation of group elements: for a matrix $\mathbf{A}$ over $\mathbb{Z}_q$, we define $[\mathbf{A}]_1 := g_1^{\mathbf{A}}$, $[\mathbf{A}]_2 := g_2^{\mathbf{A}}$, $[\mathbf{A}]_T := g_T^{\mathbf{A}}$, where exponentiation is carried out component-wise.

We also use the following less standard notations. For vectors $\mathbf{w} = (w_1, \ldots, w_\ell) \in \mathbb{Z}_q^\ell$ and $\mathbf{v} = (v_1, \ldots, v_\ell) \in \mathbb{Z}_q^\ell$ of the same length, $\mathbf{w} \odot \mathbf{v}$ denotes the vector that is obtained by component-wise multiplications. Namely, $\mathbf{v} \odot \mathbf{w} = (v_1 w_1, \ldots, v_\ell w_\ell)$. When $\mathbf{w} \in (\mathbb{Z}_q^*)^\ell$, $\mathbf{v} \oslash \mathbf{w}$ denotes the vector $\mathbf{v} \oslash \mathbf{w} = (v_1/w_1, \ldots, v_\ell/w_\ell)$. It is easy to verify that for vectors $\mathbf{c}, \mathbf{d} \in \mathbb{Z}_q^\ell$ and

State: Lists L_1, L_2, L_T over $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ respectively.

Initializations: Lists L_1, L_2, L_T sampled according to distribution $\mathcal{D}$.

Oracles: The oracles provide black-box access to the group operations, the bilinear map, and equalities.

- For all $s \in \{1, 2, T\}$: $\mathsf{add}_s(h_1, h_2)$ appends $L_s[h_1] + L_s[h_2]$ to L_s and returns its handle $(s, |L_s|)$.
- For all $s \in \{1, 2, T\}$: $\mathsf{neg}_s(h_1, h_2)$ appends $-L_s[h_1]$ to L_s and returns its handle $(s, |L_s|)$.
- $\mathsf{map}_e(h_1, h_2)$ appends $e(L_1[h_1], L_2[h_2])$ to L_T and returns its handle $(T, |L_T|)$.
- $\mathsf{zt}_T(h)$ returns 1 if $L_T[h] = 0$ and 0 otherwise.

All oracles return $\perp$ when given invalid indices.

Figure 1: Generic group model for bilinear group setting $\mathbb{G} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ and distribution $\mathcal{D}$.

$\mathbf{w} \in (\mathbb{Z}_q^*)^\ell$, we have $(\mathbf{c} \odot \mathbf{w}) \odot (\mathbf{d} \oslash \mathbf{w}) = \mathbf{c} \odot \mathbf{d}$. For group elements $[\mathbf{v}]_1 \in \mathbb{G}_1^\ell$ and $[\mathbf{w}]_1 \in \mathbb{G}_2^\ell$, $[\mathbf{v}]_1 \odot [\mathbf{w}]_2$ denotes $([v_1 w_1]_T, \ldots, [v_\ell w_\ell]_T)$, which is efficiently computable from $[\mathbf{v}]_1$ and $[\mathbf{w}]_2$ using the bilinear map e.

Generic Bilinear Group Model. Let $\mathbb{G} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ be a bilinear group setting, L_1, L_2, and L_T be lists of group elements in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ respectively, and let $\mathcal{D}$ be a distribution over L_1, L_2, and L_T. The generic group model for a bilinear group setting $\mathbb{G}$ and a distribution $\mathcal{D}$ is described in Fig. 1. In this model, the challenger first initializes the lists L_1, L_2, and L_T by sampling the group elements according to $\mathcal{D}$, and the adversary receives handles for the elements in the lists. For $s \in \{1, 2, T\}$, $L_s[h]$ denotes the h-th element in the list L_s. The handle to this element is simply the pair (s, h). An adversary running in the generic bilinear group model can apply group operations and bilinear maps to the elements in the lists. To do this, the adversary has to call the appropriate oracle specifying handles for the input elements. The challenger computes the result of a query, stores it in the corresponding list, and returns to the adversary its (newly created) handle. Handles are not unique (i.e., the same group element may appear more than once in a list under different handles). As in [AY20], we replace the equality test oracle from Baltico et. al [BCFG17] with the zero-test oracle, which is given a handle (s, h) and returns 1 if $L_s[h] = 0$ and 0 otherwise only for the case of $s = T$.

Symbolic Group Model. The symbolic group model for a bilinear group setting $\mathbb{G}$ and a distribution $\mathcal{D}_P$ gives to the adversary the same interface as the corresponding generic group model, except that internally the challenger stores lists of element in the field $\mathbb{Z}_p(X_1, \ldots, X_n)$ instead of lists of group elements, where $X_1, \ldots, X_n$ are indeterminates. The oracles add_s, neg_s, map, and zt computes addition, negation, multiplication, and equality in the field. In our work, we will use the subring $\mathbb{Z}_p[X_1, \ldots, X_n, 1/X_1, \ldots, 1/X_n]$ of the entire field $\mathbb{Z}_p(X_1, \ldots, X_n)$. Note that any element f in $\mathbb{Z}_p[X_1, \ldots, X_n, 1/X_1, \ldots, 1/X_n]$ can be represented as

$$f(X_1, \ldots, X_n) = \sum_{(c_1, \ldots, c_n) \in \mathbb{Z}^n} a_{c_1, \ldots, c_n} X_1^{c_1} \cdots X_n^{c_n}$$

using $\{a_{c_1, \ldots, c_n} \in \mathbb{Z}_p\}_{(c_1, \ldots, c_n) \in \mathbb{Z}^n}$, where we have $a_{c_1, \ldots, c_n} = 0$ for all but finite $(c_1, \ldots, c_n) \in \mathbb{Z}^n$. Note that this expression is unique.

3 Multi-Input Attribute Based and Predicate Encryption

We define multi-input Attribute Based Encryption (ABE) and Predicate Encryption (PE) below. Since the only difference between the two notions is in the security game, we unify the syntax for the algorithms in what follows.

A k-input ABE/PE scheme is parametrized over an attribute space $\{(A_\lambda)^k\}_{\lambda\in\mathbb{N}}$ and function space $\{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$, where each function maps $\{(A_\lambda)^k\}_{\lambda\in\mathbb{N}}$ to $\{0,1\}$. Such a scheme is described by procedures $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}_1, \ldots, \mathsf{Enc}_k, \mathsf{Dec})$ with the following syntax:

$\mathsf{Setup}(1^\lambda) \to (\mathsf{pp}, \mathsf{msk})$: The Setup algorithm takes as input a security parameter and outputs some public parameters pp and a master secret key msk.

$\mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, f) \to \mathsf{sk}_f$: The KeyGen algorithm takes as input the public parameters pp, a master secret key msk and a function $f \in \mathcal{F}_\lambda$ and outputs a key sk_f.

$\mathsf{Enc}_1(\mathsf{pp}, \mathsf{msk}, \alpha, b) \to \mathsf{ct}_{\alpha,b,1}$: The encryption algorithm for slot 1 takes as input the public parameters pp, a master secret key msk, an attribute $\alpha \in A_\lambda$, and message $b \in \{0,1\}$, and outputs a ciphertext $\mathsf{ct}_{\alpha,b,1}$. For the case of ABE, the attribute string α is included as part of the ciphertext.

$\mathsf{Enc}_i(\mathsf{pp}, \mathsf{msk}, \alpha) \to \mathsf{ct}_{\alpha,i}$ for $i \geq 2$: The encryption algorithm for the i^{th} slot where $i \in [2,k]$, takes as input the public parameters pp, a master secret key msk, and an attribute $\alpha \in A_\lambda$ and outputs a ciphertext $\mathsf{ct}_{\alpha,i}$. For the case of ABE, the attribute string α is included as part of the ciphertext.

$\mathsf{Dec}(\mathsf{pp}, \mathsf{sk}_f, \mathsf{ct}_{\alpha_1,b,1}, \mathsf{ct}_{\alpha_2,2}, \ldots, \mathsf{ct}_{\alpha_k,k}) \to b'$: The decryption algorithm takes as input the public parameters pp, a key for the function f and a sequence of ciphertext of $(\alpha_1, b), \alpha_2, \ldots, \alpha_k$ and outputs a string b'.

Next, we define correctness and security. For ease of notation, we drop the subscript λ in what follows.

Correctness: For every $\lambda \in \mathbb{N}, b \in \{0,1\}$, $\alpha_1, \ldots, \alpha_k \in A$, $f \in \mathcal{F}$, it holds that if $f(\alpha_1, \ldots, \alpha_k) = 1$, then

$$\Pr\left[\mathsf{Dec}\left(\begin{matrix}\mathsf{pp}, \mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, f),\\ \mathsf{Enc}_1(\mathsf{pp}, \mathsf{msk}, \alpha_1, b), \ldots, \mathsf{Enc}_k(\mathsf{pp}, \mathsf{msk}, \alpha_k)\end{matrix}\right) = b\right] = 1 - \mathrm{negl}(\lambda)$$

where the probability is over the choice of $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and over the internal randomness of KeyGen and $\mathsf{Enc}_1, \ldots, \mathsf{Enc}_k$.

Definition 3.1 ($\mathsf{Ada\text{-}IND}$ security for $\mathsf{k\text{-}ABE}$)**.** For a $\mathsf{k\text{-}ABE}$ scheme $\mathsf{k\text{-}ABE} = \{\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}_1, \ldots, \mathsf{Enc}_k, \mathsf{Dec}\}$ for an attribute space $\{(A_\lambda)^k\}_{\lambda\in\mathbb{N}}$, function space $\{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ and an adversary $\mathcal{A}$, we define the $\mathsf{Ada\text{-}IND}$ security game as follows.

1. **Setup phase:** On input 1^λ, the challenger samples $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and gives pp to $\mathcal{A}$.
2. **Query phase:** The challenger samples a bit $\beta \leftarrow \{0,1\}$. During the game, $\mathcal{A}$ adaptively makes the following queries, in an arbitrary order.
   (a) **Key Queries:** $\mathcal{A}$ makes polynomial number of key queries, say $p = p(\lambda)$. As an i-th key query, $\mathcal{A}$ chooses a function $f_i \in \mathcal{F}_\lambda$. The challenger replies with $\mathsf{sk}_{f_i} \leftarrow \mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, f_i)$.

(b) **Ciphertext Queries:** $\mathcal{A}$ issues polynomial number of ciphertext queries for each slot, say $p = p(\lambda)$. As an i-th query for a slot $j \in [k]$, $\mathcal{A}$ declares

$$\begin{cases} (\alpha_j^i, (b_0^i, b_1^i)) & \text{if } j = 1 \\ \alpha_j^i & \text{if } j \neq 1 \end{cases}$$

to the challenger, where $\alpha_j^i \in A_\lambda$ is an attribute and $(b_0^i, b_1^i) \in \{0,1\} \times \{0,1\}$ is the pair of messages. Then, the challenger computes

$$\mathsf{ct}_{j,\beta}^i = \begin{cases} \mathsf{Enc}_j(\mathsf{pp}, \mathsf{msk}, \alpha_j^i, b_\beta^i) & \text{if } j = 1 \\ \mathsf{Enc}_j(\mathsf{pp}, \mathsf{msk}, \alpha_j^i) & \text{if } j \neq 1 \end{cases}$$

and returns it to $\mathcal{A}$.

3. **Output phase:** $\mathcal{A}$ outputs a guess bit β' as the output of the experiment.

For the adversary to be *admissible*, we require that for every $f_1, \ldots, f_p \in \mathcal{F}$, it holds that $f_i(\alpha_1^{i_1}, \ldots, \alpha_k^{i_k}) = 0$ for every $i, i_1, \ldots, i_k \in [p]$.

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

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

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

Definition 3.2 (Ada-IND security for k-PE.)**.** For an k-PE scheme k-PE $= \{\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}_1, \ldots, \mathsf{Enc}_k, \mathsf{Dec}\}$ for an attribute space $\{(A_\lambda)^k\}_{\lambda \in \mathbb{N}}$, function space $\{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$ and an adversary $\mathcal{A}$, we define the Ada-IND security game as follows.

1. **Setup phase:** On input 1^λ, the challenger samples $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and gives pp to $\mathcal{A}$.

2. **Query phase:** The challenger samples a bit $\beta \leftarrow \{0,1\}$. During the game, $\mathcal{A}$ adaptively makes the following queries, in an arbitrary order.

   (a) **Key Queries:** $\mathcal{A}$ makes polynomial number of key queries, say $p = p(\lambda)$. For each key query $i \in [p]$, $\mathcal{A}$ chooses a function $f_i \in \mathcal{F}_\lambda$. The challenger replies with $\mathsf{sk}_{f_i} \leftarrow \mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, f_i)$.

   (b) **Ciphertext Queries:** $\mathcal{A}$ issues polynomial number of ciphertext queries for each slot, say $p = p(\lambda)$. As an i-th query for a slot $j \in [k]$, $\mathcal{A}$ declares

$$\begin{cases} ((\alpha_{j,0}^i, \alpha_{j,1}^i), (b_0^i, b_1^i)) & \text{if } j = 1 \\ (\alpha_{j,0}^i, \alpha_{j,1}^i) & \text{if } j \neq 1 \end{cases}$$

   to the challenger, where $(\alpha_{j,0}^i, \alpha_{j,1}^i)$ is a pair of attributes and (b_0^i, b_1^i) is the pair of messages. Then, the challenger computes

$$\mathsf{ct}_{j,\beta}^i = \begin{cases} \mathsf{Enc}_j(\mathsf{pp}, \mathsf{msk}, \alpha_{j,\beta}^i, b_\beta^i) & \text{if } j = 1 \\ \mathsf{Enc}_j(\mathsf{pp}, \mathsf{msk}, \alpha_{j,\beta}^i) & \text{if } j \neq 1 \end{cases}$$

   and returns it to $\mathcal{A}$.

3. **Output phase:** $\mathcal{A}$ outputs a guess bit β' as the output of the experiment.

For the adversary to be *admissible*, we require that for every $f_1, \ldots, f_p \in \mathcal{F}$, it holds that $f_i(\alpha_{1,\beta}^{i_1}, \ldots, \alpha_{k,\beta}^{i_k}) = 0$ for every $i, i_1, \ldots, i_k \in [p]$ and $\beta \in \{0,1\}$.

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

$$\mathsf{Adv}^{\mathsf{Ada\text{-}IND}}_{\mathsf{k\text{-}PE},\mathcal{A}}(1^\lambda) := \left|\Pr[\mathsf{Exp}_{\mathsf{k\text{-}PE},\mathcal{A}}(1^\lambda) = 1|\beta = 0] - \Pr[\mathsf{Exp}_{\mathsf{k\text{-}PE},\mathcal{A}}(1^\lambda) = 1|\beta = 1]\right|.$$

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

3.1 Strong Security for k-ABE and k-PE

We also consider a stronger security notion for both k-ABE as well as k-PE where the adversary is allowed to make decrypting key requests for ciphertexts so long as they do not distinguish the challenge bit.

Definition 3.3 (Strong Ada-IND security for k-ABE.)**.** The definition for strong Ada-IND security for k-ABE is the same as standard Ada-IND security (Definition 3.1) except for the following modification. For the k-ABE adversary to be *admissible* in the *strong* Ada-IND game, we require that

- If $f_i(\alpha_1^{i_1}, \ldots, \alpha_k^{i_k}) = 1$ holds for some $i, i_1, \ldots, i_k \in [p]$, then $b_0^{i_1} = b_1^{i_1}$.

Let $(\alpha^i, (b_0^i, b_1^i))$ be the i^{th} ciphertext query in slot 1. Then, if $b_0^i \neq b_1^i$, we call the ciphertext returned by the challenger as a *challenge* ciphertext as it encodes the challenge bit β. Otherwise, we refer to it as *decrypting* ciphertext, as the adversary may potentially request a key to decrypt it.

Definition 3.4 (Strong Ada-IND security for k-PE.)**.** The definition for strong Ada-IND security for k-PE is the same as standard Ada-IND security (Definition 3.2) except for the following modification. For the k-PE adversary to be *admissible* in the *strong* Ada-IND game, we require that

- If $f_i(\alpha_{1,\beta}^{i_1}, \ldots, \alpha_{k,\beta}^{i_k}) = 1$ holds for some $i, i_1, \ldots, i_k \in [p]$ and $\beta \in \{0,1\}$, then $(\alpha_{1,0}^{i_1}, \ldots, \alpha_{k,0}^{i_k}) = (\alpha_{1,1}^{i_1}, \ldots, \alpha_{k,1}^{i_k})$ and $b_0^{i_1} = b_1^{i_1}$.

Let $\big((\alpha_0^i, \alpha_1^i), (b_0^i, b_1^i)\big)$ be the i^{th} ciphertext query in slot 1. Then, if $\alpha_0^i \neq \alpha_1^i$ or $b_0^i \neq b_1^i$, we call the ciphertext returned by the challenger as a *challenge* ciphertext as it encodes the challenge bit β. Otherwise, we refer to it as *decrypting* ciphertext, as the adversary may potentially request a key to decrypt it.

Definition 3.5 (Strong VerSel-IND security for k-ABE and k-PE)**.** The definitions for strong VerSel-IND security for k-ABE and k-PE are the same as strong Ada-IND security above except that the adversary $\mathcal{A}$ is required to submit the challenge queries and secret key queries to the challenger before it samples the public key.

3.2 Generalization to Multi-Slot Message Scheme

In the above, we focus our attention on k-ABE and k-PE schemes that only contain a message in a single slot, the remaining slots being free of messages. We can also consider a generalized version

of the notions where each slot carries a message and all the messages are recovered in successful decryption. For k polynomial, it is easy to extend a construction with single slot message to the generalized version where each slot contains a message, simply by running k instances of the scheme in parallel and rotating the slot which contains the message in each instance to cover all k slots. Moreover we claim that since the k message scheme is a concatenation of k one message schemes, security of the latter implies security of the former. In more detail, suppose there exists an adversary against the k message scheme with non-negligible advantage ϵ. This can be used to construct an adversary against one of the underlying one message schemes with non-negligible advantage ϵ/k.

4 Two-Input ABE for NC_1 from Pairings and LWE

In this section, we construct two input ABE for NC_1 circuits. More formally, our construction can support attribute space $A_\lambda = \{0,1\}^{\ell(\lambda)}$, and any circuit class $\mathcal{F} = \{\mathcal{F}_\lambda\}_\lambda$ that is subclass of $\{\mathcal{C}_{2\ell(\lambda),d(\lambda)}\}_\lambda$ with arbitrary $\ell(\lambda) \leq \mathrm{poly}(\lambda)$ and $d(\lambda) = O(\log \lambda)$, where $\mathcal{C}_{2\ell(\lambda),d(\lambda)}$ is a set of circuits with input length $2\ell(\lambda)$ and depth at most $d(\lambda)$. We can prove that the scheme satisfies strong security as per Definition 3.3 assuming LWE in bilinear generic group model. Since the intuition was described in Section 1, we proceed directly with the construction. We refer to Sec. 2.4 and Sec. 2.6 for backgrounds on lattices and pairings respectively and Sec. 2.5 for description of the kpABE scheme by Boneh et al. [BGG+14] on which our construction is based.

4.1 Construction

We proceed to describe our construction.

$\mathsf{Setup}(1^\lambda)$: On input 1^λ, the setup algorithm defines the parameters $n = n(\lambda)$, $m = m(\lambda)$, noise distribution χ over $\mathbb{Z}$, $\tau_0 = \tau_0(\lambda)$, $\tau = \tau(\lambda)$, and $B = B(\lambda)$ as specified in Sec. 2.5. It samples a group description $\mathbb{G} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, [1]_1, [1]_2)$. Sets $L := (3\ell+1)m + 2$ and proceeds as follows.

1. Sample BGG^+ scheme:
   (a) Sample $(\mathbf{A}, \mathbf{A}_{\tau_0}^{-1}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$ such that $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$.
   (b) Sample random matrix $\mathbf{B} = (\mathbf{B}_1, \ldots, \mathbf{B}_{2\ell}) \leftarrow (\mathbb{Z}_q^{n\times m})^{2\ell}$ and a random vector $\mathbf{u} \leftarrow \mathbb{Z}_q^n$.
2. Sample $\mathbf{w} \leftarrow (\mathbb{Z}_q^*)^L$.
3. Output $\mathsf{pp} = (\mathbf{A}, \mathbf{B}, \mathbf{u})$, $\quad \mathsf{msk} = \left(\mathbf{A}_{\tau_0}^{-1}, \mathbf{w}, [1]_1, [1]_2\right)$.

$\mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, F)$: Given input the public parameters pp, master secret key msk and a circuit F, compute BGG^+ function key for circuit F as follows:

1. Compute $\mathbf{H}_F = \mathsf{EvalF}(\mathbf{B}, F)$ and $\mathbf{B}_F = \mathbf{B}\mathbf{H}_F$.
2. Compute $[\mathbf{A}\|\mathbf{B}_F]_\tau^{-1}$ from $\mathbf{A}_{\tau_0}^{-1}$ and sample $\mathbf{r} \in \mathbb{Z}^{2m}$ as $\mathbf{r}^\top \leftarrow [\mathbf{A}\|\mathbf{B}_F]_\tau^{-1}(\mathbf{u}^\top)$.
3. Output the secret key $\mathsf{sk}_F := \mathbf{r}$.

$\mathsf{Enc}_1(\mathsf{pp}, \mathsf{msk}, \mathbf{x}_1, b)$: Given input the public parameters pp, master secret key msk, attribute vector $\mathbf{x}_1$, message bit b, encryption for slot 1 is defined as follows:

1. Sample LWE secret $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ and noise terms $e_0 \leftarrow \chi$, $\mathbf{e} \leftarrow \chi^m$, $\mathbf{e}_i, \mathbf{e}_{\ell+i,b} \leftarrow \widetilde{\chi^m}$ for $i \in [\ell], b \in \{0,1\}$, where $\widetilde{\chi^m}$ is defined as in Sec. 2.4.
2. For $i \in [\ell]$, compute $\psi_i := \mathbf{s}(\mathbf{B}_i - x_{1,i}\mathbf{G}) + \mathbf{e}_i$.
3. For $i \in [\ell+1, 2\ell]$, $b \in \{0,1\}$, compute $\psi_{i,b} := \mathbf{s}(\mathbf{B}_i - b\mathbf{G}) + \mathbf{e}_{i,b}$.
4. Compute $\psi_{2\ell+1} := \mathbf{sA} + \mathbf{e}$ and $\psi_{2\ell+2} := \mathbf{su}^\top + e_0$.
5. Set $\mu = \lceil \frac{q}{2} \rceil b$.
6. Set $\mathbf{c} = (1, \{\psi_i\}_{i\in[\ell]}, \{\psi_{i,b}\}_{i\in[\ell+1,2\ell],b\in\{0,1\}}, \psi_{2\ell+1}, \psi_{2\ell+2} + \mu)$.
7. Sample $t_1 \leftarrow \mathbb{Z}_q^*$ and output $\mathsf{ct}_1 = [t_1\mathbf{c} \odot \mathbf{w}]_1$.

$\mathsf{Enc}_2(\mathsf{pp}, \mathsf{msk}, \mathbf{x}_2)$: Given input the public parameters pp, master secret key msk, attribute vector $\mathbf{x}_2$, encryption for slot 2 is defined as follows:

1. Let $\mathbf{1}_a := (1, \ldots, 1) \in \mathbb{Z}_q^a$ and $\mathbf{0}_a := (0, \ldots, 0) \in \mathbb{Z}_q^a$. Set
$$\hat{\psi}_{i,b} := \begin{cases} \mathbf{1}_m \in \mathbb{Z}_q^m & \text{if } b = x_{2,i} \\ \mathbf{0}_m \in \mathbb{Z}_q^m & \text{if } b \neq x_{2,i} \end{cases} \quad \text{for } i \in [\ell+1, 2\ell] \text{ and } b \in \{0,1\}.$$
2. Set $\mathbf{d} = (1, \mathbf{1}_{\ell m}, \{\hat{\psi}_{i,b}\}_{i\in[\ell+1,2\ell],b\in\{0,1\}}, \mathbf{1}_m, 1)$.
3. Sample $t_2 \leftarrow \mathbb{Z}_q^*$ and output $\mathsf{ct}_2 = [t_2\mathbf{d} \oslash \mathbf{w}]_2$.

$\mathsf{Dec}(\mathsf{pp}, \mathsf{sk}_F, \mathsf{ct}_1, \mathsf{ct}_2)$: The decryption algorithm takes as input the public parameters pp, the secret key sk_F for circuit F and ciphertexts ct_1 and ct_2 corresponding to the two attributes $\mathbf{x}_1$ and $\mathbf{x}_2$ and proceeds as follows:

1. Take the coordinate-wise pairing between ciphertexts:
Compute $[\mathbf{v}]_T = [t_1t_2\mathbf{c} \odot \mathbf{d}]_T$ as $\mathsf{ct}_1 \odot \mathsf{ct}_2$.
2. De-vectorize obtained vector:
Expand $[\mathbf{v}]_T$ for $i \in [\ell]$, $j \in [\ell+1, 2\ell], b \in \{0,1\}$, to obtain:
$$[v_0]_T = [t_1t_2]_T,\ [\mathbf{v}_i]_T = [t_1t_2\psi_i]_T,$$
$$[\mathbf{v}_{j,b}]_T = [t_1t_2\psi'_{j,b}]_T, \text{ where } \psi'_{j,b} = \begin{cases} (\mathbf{s}(\mathbf{B}_j - x_{2,j}\mathbf{G}) + \mathbf{e}_{j,b}), & \text{if } b = x_{2,j} \\ \mathbf{0}, & \text{if } b = 1 - x_{2,j} \end{cases},$$
$$[\mathbf{v}_{2\ell+1}]_T = [t_1t_2\psi_{2\ell+1}]_T,\ [v_{2\ell+2}]_T = [t_1t_2(\psi_{2\ell+2} + \mu)]_T.$$
3. Compute Evaluation function for BGG^+ ciphertexts in exponent:
Let $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$. Compute $\widehat{\mathbf{H}}_{F,\mathbf{x}} = \mathsf{EvalFX}(F, \mathbf{x}, \mathbf{B})$.
4. Perform BGG^+ decryption in the exponent:
Form $[\mathbf{v}_\mathbf{x}]_T = [\mathbf{v}_1, \ldots, \mathbf{v}_\ell, \mathbf{v}_{\ell+1,x_{2,1}}, \ldots \mathbf{v}_{2\ell,x_{2,\ell}}]_T$ and parse $\mathsf{sk}_F = \mathbf{r}$ as $\mathbf{r} = (\mathbf{r}_1 \in \mathbb{Z}_q^m, \mathbf{r}_2 \in \mathbb{Z}_q^m)$. Then compute
$$[v']_T := [(v_{2\ell+2} - (\mathbf{v}_{2\ell+1}\mathbf{r}_1^\top + \mathbf{v}_\mathbf{x}\widehat{\mathbf{H}}_{F,\mathbf{x}}\mathbf{r}_2^\top))]_T$$
5. Recover exponent via brute force if $F(\mathbf{x}) = 0$:
Find $\eta \in [-B, B] \cup [-B + \lceil q/2 \rceil, B + \lceil q/2 \rceil]$ such that $[v_0]_T^\eta = [v']_T$ by brute-force search. If there is no such η, output $\perp$. To speed up the operation, one can employ the baby-step giant-step algorithm.
6. Output 0 if $\eta \in [-B, B]$ and 1 if $[-B + \lceil q/2 \rceil, B + \lceil q/2 \rceil]$.

Correctness: To see correctness, we first make following observations:

1. $\mathbf{c} \odot \mathbf{d} = (1, \{\psi_i\}_{i\in[\ell]}, \{\psi'_{i,b}\}_{i\in[\ell+1,2\ell],b\in\{0,1\}}, \psi_{2\ell+1}, \psi_{2\ell+2} + \mu)$ where,

$$\psi'_{i,b} = \begin{cases} (\mathbf{s}(\mathbf{B}_i - x_{2,i}\mathbf{G}) + \mathbf{e}_i) & \text{if } b = x_{2,i} \\ \mathbf{0} & \text{if } b = 1 - x_{2,i} \end{cases}.$$

Recall that $[\mathbf{v}]_T = [t_1 t_2 \mathbf{c} \odot \mathbf{d}]_T$. Now, letting $\mathbf{v}_\mathbf{x} = (\mathbf{v}_1, \ldots, \mathbf{v}_\ell, \mathbf{v}_{\ell+1,x_{2,1}}, \ldots \mathbf{v}_{2\ell,x_{2,\ell}})$ and $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$, on de-vectorizing it the decryptor obtains

$$\begin{aligned} &[v_0]_T = [t_1 t_2]_T,\ [\mathbf{v}_i]_T = [t_1 t_2(\mathbf{s}(\mathbf{B}_i - x_i\mathbf{G}) + \mathbf{e}_i)]_T \ \text{ for } i \in [\ell], \\ &[\mathbf{v}_{i,x_i}]_T = [t_1 t_2(\mathbf{s}(\mathbf{B}_i - x_i\mathbf{G}) + \mathbf{e}_{i,x_i})]_T \ \text{ for } i \in [\ell+1, 2\ell], \\ &[\mathbf{v}_{2\ell+1}]_T = [t_1 t_2(\mathbf{s}\mathbf{A} + \mathbf{e})]_T,\ \ [v_{2\ell+2}]_T = [t_1 t_2(\mathbf{s}\mathbf{u}^\top + e_0 + \mu)] \end{aligned}$$

2. Next, observe that:

$$\begin{aligned} \mathbf{v}_\mathbf{x} &= t_1 t_2(\mathbf{s}(\mathbf{B}_1 - x_1\mathbf{G}) + \mathbf{e}_1, \ldots, \mathbf{s}(\mathbf{B}_{2\ell} - x_{2\ell}\mathbf{G}) + \mathbf{e}_{2\ell}) \\ &= t_1 t_2 \mathbf{s}((\mathbf{B}_1, \ldots, \mathbf{B}_{2\ell}) - (x_1\mathbf{G}, \ldots, x_{2\ell}\mathbf{G})) + t_1 t_2(\mathbf{e}_1, \ldots, \mathbf{e}_{2\ell}) \\ &= t_1 t_2 \mathbf{s}(\mathbf{B} - \mathbf{x} \otimes \mathbf{G}) + t_1 t_2 \mathbf{e}_\mathbf{x}, \\ &\quad \text{where } \mathbf{e}_i = \mathbf{e}_{i,x_i} \text{ for } i \in [\ell+1, 2\ell] \text{ and } \mathbf{e}_\mathbf{x} = (\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_{2\ell}) \end{aligned}$$

3. Performing BGG^+ evaluation and decryption in the exponent yields:

$$\begin{aligned} [v']_T &= [\big(v_{2\ell+2} - (\mathbf{v}_{2\ell+1}\mathbf{r}_1^\top + \mathbf{v}_\mathbf{x}\widehat{\mathbf{H}}_{F,\mathbf{x}}\mathbf{r}_2^\top)\big)]_T \\ &= [t_1 t_2(\mathbf{s}\mathbf{u}^\top + \mu + e_0) - t_1 t_2(\mathbf{s}\mathbf{A} + \mathbf{e})\mathbf{r}_1^\top - t_1 t_2(\mathbf{s}(\mathbf{B} - \mathbf{x} \otimes \mathbf{G}) + \mathbf{e}_\mathbf{x})\widehat{\mathbf{H}}_{F,\mathbf{x}}\mathbf{r}_2^\top]_T \\ &= [t_1 t_2(\mathbf{s}\mathbf{u}^\top + \mu - \mathbf{s}(\mathbf{A}\mathbf{r}_1^\top + (\mathbf{B}\mathbf{H}_F - F(\mathbf{x})\mathbf{G})\mathbf{r}_2^\top)) + t_1 t_2(e_0 - \mathbf{e}\mathbf{r}_1^\top - \mathbf{e}_\mathbf{x}\widehat{\mathbf{H}}_{F,\mathbf{x}}\mathbf{r}_2^\top)]_T \\ &\quad (\because (\mathbf{B} - \mathbf{x} \otimes \mathbf{G})\widehat{\mathbf{H}}_{F,\mathbf{x}} = \mathbf{B}\mathbf{H}_F - F(\mathbf{x})\mathbf{G} \text{ (Lemma 2.13)}). \end{aligned}$$

For $F(\mathbf{x}) = 0$, and replacing $\mathbf{B}\mathbf{H}_F$ by $\mathbf{B}_F$, $(\mathbf{r}_1, \mathbf{r}_2)$ by $\mathbf{r}$, we get,

$$\begin{aligned} [v']_T &= [t_1 t_2(\mathbf{s}\mathbf{u}^\top + \mu - \mathbf{s}(\mathbf{A}\|\mathbf{B}_F)\mathbf{r}^\top + e')]_T \quad (\text{replacing } (e_0 - \mathbf{e}\mathbf{r}_1^\top - \mathbf{e}_\mathbf{x}\widehat{\mathbf{H}}_{F,\mathbf{x}}\mathbf{r}_2^\top) \text{ by } e') \\ &= [t_1 t_2(\mathbf{s}\mathbf{u}^\top + \mu - \mathbf{s}\mathbf{u}^\top + e')]_T, \quad \text{because } (\mathbf{A}\|\mathbf{B}_F)\mathbf{r}^\top = \mathbf{u}. \\ &= [t_1 t_2(\mu + e')]_T = [v_0]_T^{(\mu+e')}. \end{aligned}$$

4. Error bound in v':
Recall that we set $\chi = \mathsf{SampZ}(3\sqrt{n})$. By the definition of SampZ, we have $\|e_0\|_\infty \le 3n$ and $\|\mathbf{e}\|_\infty \le 3n$. Furthermore, we have $\|\mathbf{e}_\ell\|_\infty, \|\mathbf{e}_{\ell+i,b}\|_\infty \le 3mn$ for $i \in [\ell]$ and $b \in \{0,1\}$ by the definition of $\widetilde{\chi^m}$, $\|\mathbf{r}\|_\infty \le \sqrt{n}\tau$, and $\|\widehat{\mathbf{H}}_{F,\mathbf{x}}\|_\infty \le m \cdot 2^{O(d)}$, where the last inequality follows from Lemma 2.13. Thus, we have

$$e' = e_0 - \mathbf{e}\mathbf{r}_1^\top - \mathbf{e}_\mathbf{x}\widehat{\mathbf{H}}_{F,\mathbf{x}}\mathbf{r}_2^\top \le O(\ell m^5 n^{1.5}\tau \cdot 2^{O(d)}) \le B$$

by our choice of B.

5. Finally, since $B = \mathrm{poly}(n, \ell) \cdot 2^{O(d)} = \mathrm{poly}(\lambda)$, we can recover $\eta = \mu + e'$ by brute force search in polynomial time as defined in step 5 and then the message as defined in step 6 of decryption algorithm.

4.2 Security

We prove the security via the following theorem.

Theorem 4.1. *Our* 2ABE *scheme for function class* NC_1 *satisfies strong* Ada-IND *security in the generic group model assuming that the* kpABE *scheme* BGG^+ *for function class* NC_1 *satisfies* Ada-INDr *security.*

Overview. The proof is designed via a sequence of games. To begin, we prove that it is pointless for the adversary to take pairing products between non-matching positions of the ciphertexts of the two parties and then take linear combinations among them. This may be argued because of the randomness $\mathbf{w}$ in the ciphertexts which is only cancelled when matching positions are paired. This enables us to argue that the only possible strategy for the adversary is to take linear combinations among partial decryption results yielded by computing the pairing between matching positions of the ciphertexts. Next we show that taking partial decryption results between matching positions of different pairs of ciphertexts is useless, because the randomness $t_1 t_2$ will change across multiple ciphertexts. This step excludes mix and match attacks between different pairs of ciphertexts and reduces the adversary strategy to gaining information about the message(s) via results obtained by legitimate pairing of two entire ciphertexts. At this point, we invoke the security of BGG^+ to argue that the message is hidden.

Proof. Consider a PPT adversary $\mathcal{A}$ that makes at most $Q_{\mathsf{ct}}(\lambda)$ ciphertext queries (in both slots) and $Q_{\mathsf{zt}}(\lambda)$ zero-test queries during the game. We denote the event that $\mathcal{A}$ outputs correct guess for the challenge bit β at the end of $\mathbf{Game}_{\mathrm{x}}$ as E_{x}.

$\mathbf{Game}_0$: This is the real game in the generic group model. Without loss of generality, we assume that the challenger simulates the generic group oracle for $\mathcal{A}$. At the beginning of the game, the challenger samples the public parameters $\mathsf{pp} = (\mathbf{A}, \mathbf{B}, \mathbf{u})$ and master secret key $\mathsf{msk} = \left(\mathbf{A}_{\tau_0}^{-1}, \mathbf{w},\ [1]_1,\ [1]_2\right)$ as described in the scheme. It also samples a random bit β and keeps it with itself. Then, it returns the public parameters pp to $\mathcal{A}$. It handles $\mathcal{A}$'s queries as follows:

1. Slot 1 ciphertext queries: To answer the i-th slot 1 ciphertext query $(\mathbf{x}_1^i, b_0^i, b_1^i)$, it samples $t_1^i \leftarrow \mathbb{Z}_q^*$, computes $\mathbf{c}^i = (c_1^i, \ldots, c_L^i)$ as specified by Enc_1 for message b_β^i and returns handles to $\mathsf{ct}_1^i = [t_1^i \mathbf{c}^i \odot \mathbf{w}]_1$.
2. Slot 2 ciphertext queries: To answer the i-th slot 2 ciphertext query $\mathbf{x}_2^i$, the challenger samples $t_2^i \leftarrow \mathbb{Z}_q^*$, computes $\mathbf{d}^i$ as specified by Enc_2 and returns handles to $\mathsf{ct}_2^i = [t_2^i \mathbf{d}^i \oslash \mathbf{w}]_2$.
3. Secret Key queries: To respond to the j-th key query F^j made by $\mathcal{A}$, the challenger computes $\mathbf{r}^j$ as specified in the KeyGen algorithm and returns it to $\mathcal{A}$.

By definition, the advantage of $\mathcal{A}$ against the scheme is $\left|\Pr[\mathsf{E}_0] - \frac{1}{2}\right|$.

$\mathbf{Game}_1$: In this game, we switch partially to the symbolic group model and change the variables $(w_1, \ldots, w_L)$, $(t_1^1, \ldots, t_1^{Q_{\mathsf{ct}}})$, $(t_2^1, \ldots, t_2^{Q_{\mathsf{ct}}})$ and $(c_1^i, \ldots, c_L^i)$ to formal variables $(W_1, \ldots, W_L)$, $(T_1^1, \ldots, T_1^{Q_{\mathsf{ct}}})$, $(T_2^1, \ldots, T_2^{Q_{\mathsf{ct}}})$, $(C_1^i, \ldots, C_L^i)$. As a result, all handles given to $\mathcal{A}$ refer to elements in the ring

$$\mathbb{T} := \mathbb{Z}_q \left[\begin{matrix} W_1, \ldots, W_L, 1/W_1, \ldots, 1/W_L, T_1^1, \ldots, T_1^{Q_{\mathsf{ct}}}, T_2^1, \ldots, T_2^{Q_{\mathsf{ct}}}, \\ C_1^1, \ldots, C_L^1, \ldots, C_1^{Q_{\mathsf{ct}}}, \ldots, C_L^{Q_{\mathsf{ct}}} \end{matrix} \right].$$

where $\{1/W_i\}_i$ are needed to represent the components in the secret keys. However, when the challenger answers the zero-test queries, it substitutes the formal variables with corresponding elements in $\mathbb{Z}_q$. In doing so, if the variable is not assigned a value in $\mathbb{Z}_q$, we sample corresponding value from the same distribution as in the real world. Once a value is assigned to a variable, we use the same value throughout the rest of the game. As we argue in Lemma 4.2, we have:

$$\Pr[\mathsf{E}_0] = \Pr[\mathsf{E}_1].$$

Here, we list all the components in $\mathbb{T}$ for which corresponding handles are given to $\mathcal{A}$ in $\mathbf{Game}_1$ as handles to the group elements in ciphertexts of both slots:

$$\mathcal{S}_1 := \left\{ \{T_1^i\, C_k^i\, W_k\}_{k\in[L],i\in[Q_{\mathsf{ct}}]} \right\}, \quad \mathcal{S}_2 := \left\{ \{T_2^i/W_k\}_{k\in[L],i\in[Q_{\mathsf{ct}}]} \text{ s.t. } d_k^i = 1 \right\}$$

Note that $\mathcal{S}_1$ and $\mathcal{S}_2$ correspond to handles for elements in $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. We then define $\mathcal{S}_T$ as $\mathcal{S}_T := \{X \cdot Y : X \in \mathcal{S}_1, Y \in \mathcal{S}_2, X \cdot Y \neq 0\}$. If we explicitly write down $\mathcal{S}_T$, we have $\mathcal{S}_T = \mathcal{S}_{T,1} \cup \mathcal{S}_{T,2}$, where

$$\mathcal{S}_{T,1} := \left\{ T_1^i\, T_2^j\, C_k^i\, W_k/W_{k'}, \quad \text{for } k,k' \in [L],\ i,j \in [Q_{\mathsf{ct}}] \right\}$$
$$\text{and } \mathcal{S}_{T,2} := \{ T_1^i\, T_2^j\, C_k^i \quad \text{for } k \in [L],\ i,j \in [Q_{\mathsf{ct}}], \text{ s.t. } d_k^j = 1 \quad \}.$$

Note that any handle submitted to the zero-test oracle by $\mathcal{A}$ during the game refers to an element f in $\mathbb{T}$ that can be represented as

$$f(W_1,\ldots,W_L,T_1^1,\ldots,T_1^{Q_{\mathsf{ct}}},T_2^1,\ldots,T_2^{Q_{\mathsf{ct}}},C_1^1,\ldots,C_L^{Q_{\mathsf{ct}}}) = \sum_{Z\in\mathcal{S}_T} a_Z Z \tag{4.1}$$

where the coefficients $\{a_Z \in \mathbb{Z}_q\}_{Z\in\mathcal{S}_T}$ can be efficiently computed. Furthermore, $\{a_Z \in \mathbb{Z}_q\}_{Z\in\mathcal{S}_T}$ satisfying the above equation is unique since all monomials in $\mathcal{S}_T$ are distinct.

$\mathbf{Game}_2$: In this game, we use the formal variables $(W_1,\ldots,W_L)$, $(T_1^1,\ldots,T_1^{Q_{\mathsf{ct}}})$, $(T_2^1,\ldots,T_2^{Q_{\mathsf{ct}}})$ even while answering zero test queries. However, $C_1^1,\ldots,C_L^{Q_{\mathsf{ct}}}$ are still replaced by $c_1^1,\ldots,c_L^{Q_{\mathsf{ct}}}$. Namely, given a zero test query $f \in \mathbb{T}$, the challenger returns 1 if:

$$f(W_1,\ldots,W_L,T_1^1,\ldots,T_1^{Q_{\mathsf{ct}}},T_2^1,\ldots,T_2^{Q_{\mathsf{ct}}},c_1^1,\ldots,c_L^{Q_{\mathsf{ct}}}) = 0. \tag{4.2}$$

We show in Lemma 4.3 that

$$|\Pr[\mathsf{E}_1] - \Pr[\mathsf{E}_2]| \leq Q_{\mathsf{zt}}(L+3)/q.$$

$\mathbf{Game}_3$: In this game, we further change the way zero-test queries are answered. In particular, when $\mathcal{A}$ makes a zero-test query for a handle corresponding to $f \in \mathbb{T}$ that can be represented as Eq. (4.1), the challenger returns 0 if there exists $Z \in \mathcal{S}_{T,1}$ such that $a_Z \neq 0$. Otherwise, the challenger answers the query as in the previous game. As we prove in Lemma 4.4, we have $\Pr[\mathsf{E}_2] = \Pr[\mathsf{E}_3]$.

$\mathbf{Game}_4$: In this game, we partition the set $\mathcal{S}_{T,2}$ by (i,j) pairs as:

$$\mathcal{S}_{T,2} = \cup_{i,j\in[Q_{\mathsf{ct}}]}\mathcal{S}_{T,2,i,j} \quad \text{where} \quad \mathcal{S}_{T,2,i,j} = \{ T_1^i\, T_2^j\, C_k^i \quad \text{for } k \in [L] \text{ s.t } d_k^j = 1 \ \}.$$

We note the that every term Z in $\mathcal{S}_{T,2,i,j}$ can be represented by the variables T_1^i, T_2^j, and $C_1^i, \ldots, C_L^i$ by the definition of $\mathcal{S}_{T,2,i,j}$. For a zero test query f that is represented as

$$f(W_1, \ldots, W_L, T_1^1, \ldots, T_1^{Q_{\mathsf{ct}}}, T_2^1, \ldots, T_2^{Q_{\mathsf{ct}}}, C_1^1, \ldots, C_L^{Q_{\mathsf{ct}}}) = \sum_{i,j \in [Q_{\mathsf{ct}}]} \sum_{Z \in \mathcal{S}_{T,2,i,j}} a_Z Z, \quad (4.3)$$

we change the game so that the challenger returns 0 if there exists a pair (i, j) such that

$$\sum_{Z \in \mathcal{S}_{T,2,i,j}} a_Z Z(T_1^i, T_2^j, c_1^i, \ldots, c_L^i) \neq 0 \quad \text{over } \mathbb{T}. \quad (4.4)$$

As we prove in Lemma 4.5, we have $\Pr[\mathsf{E}_3] = \Pr[\mathsf{E}_4]$.

Game$_5$: Recall that in the previous game, for a zero test query that is represented as Eq. (4.3), the challenger returns 0 unless Eq. (4.4) holds for all i, j. In this game, for (i, j) such that the i-th ciphertext for slot 1 is a challenge ciphertext (please see Section 3.1 to recall the definition of challenge ciphertext), we replace the check with the new one that checks whether

$$\sum_{Z \in \mathcal{S}_{T,2,i,j}} a_Z Z(T_1^i, T_2^j, C_1^i, \ldots, C_L^i) = 0$$

holds over $\mathbb{T}$. Namely, for such (i, j), we stop replacing the variables $\{C_k^i\}_k$ with the corresponding values $\{c_k^i\}_k$ in $\mathbb{Z}_q$. As we prove in Lemma 4.6, we have $|\Pr[\mathsf{E}_4] - \Pr[\mathsf{E}_5]| \leq \mathrm{negl}(\lambda)$ assuming $\mathsf{Ada\text{-}INDr}$ security of BGG^+, which follows from LWE.

Next, we observe that the adversary cannot obtain any information about the encrypted messages in **Game**$_5$ since the challenge ciphertexts are replaced by formal variables that do not contain any information of the challenge bit, and the answers to the zero test queries do not depend on the challenge bit either.

Indistinguishability of Hybrids. We next argue that consecutive hybrids are indistinguishable.

Lemma 4.2 (**Game**$_0 \equiv$ **Game**$_1$)**.** *We have* $\Pr[\mathsf{E}_0] = \Pr[\mathsf{E}_1]$.

Proof. Since zero-test queries in **Game**$_1$ are answered by using variables that are sampled from exactly the same distribution as that in **Game**$_0$, the view of $\mathcal{A}$ in **Game**$_1$ is not altered from that in **Game**$_0$. The lemma therefore follows. □

Lemma 4.3 (**Game**$_1 \approx_s$ **Game**$_2$)**.** *We have* $|\Pr[\mathsf{E}_1] - \Pr[\mathsf{E}_2]| \leq Q_{\mathsf{zt}}(L+3)/q$.

Proof. The two games are different only when $\mathcal{A}$ submits a zero test query corresponding to a polynomial $f \in \mathbb{T}$ such that

$$f(w_1, \ldots, w_L, t_1^1, \ldots, t_1^{Q_{\mathsf{ct}}}, t_2^1, \ldots, t_2^{Q_{\mathsf{ct}}}, c_1^1, \ldots, c_L^{Q_{\mathsf{ct}}}) = 0 \quad (4.5)$$

but

$$f(W_1, \ldots, W_L, T_1^1, \ldots, T_1^{Q_{\mathsf{ct}}}, T_2^1, \ldots, T_2^{Q_{\mathsf{ct}}}, c_1^1, \ldots, c_L^{Q_{\mathsf{ct}}}) \neq 0 \quad (4.6)$$

We will use the Schwartz-Zippel lemma to bound the probability that this occurs. We define a new polynomial $g \in \mathbb{T}$ to clear the denominators as:

$$g(W_1, \ldots, W_L, T_1^1, \ldots, T_1^{Q_{\mathsf{ct}}}, T_2^1, \ldots, T_2^{Q_{\mathsf{ct}}}) = \left(\prod_{i \in [L]} W_i\right) \cdot f(W_1, \ldots, W_L, T_1^1, \ldots, T_1^{Q_{\mathsf{ct}}}, T_2^1, \ldots, T_2^{Q_{\mathsf{ct}}}, c_1^1, \ldots, c_L^{Q_{\mathsf{ct}}})$$

Observe that the polynomial has degree $L+3$ where L comes from the leading product of W_i and 3 comes from the degree of the terms in $\mathcal{S}_T$. We can bound the probability by $\frac{Q_{\mathsf{zt}}(L+3)}{q}$. □

Lemma 4.4 ($\mathbf{Game}_2 \equiv \mathbf{Game}_3$). *We have* $\Pr[\mathsf{E}_2] = \Pr[\mathsf{E}_3]$.

Proof. We observe that $\mathbf{Game}_2$ and $\mathbf{Game}_3$ differ only when $\mathcal{A}$ makes a zero-test query for a handle $f \in \mathbb{T}$ represented as Eq. (4.1) such that $a_Z \neq 0$ for some $Z \in \mathcal{S}_{T,1}$ and corresponding f equals to 0 in $\mathbb{T}$. We claim that such f does not exist and two games are actually equivalent. This is because monomials in $\mathcal{S}_{T,1}$ and $\mathcal{S}_{T,2}$ are distinct even if we replace the formal variables $\{C_j^i\}_{i,j}$ with values $\{c_j^i\}_{i,j}$ in $\mathbb{Z}_q$. Hence, if there exists $Z \in \mathcal{S}_{T,1}$ such that $a_Z \neq 0$, there would be no way to cancel this term using the remaining monomials. □

Lemma 4.5 ($\mathbf{Game}_3 \approx_s \mathbf{Game}_4$). *We have* $|\Pr[\mathsf{E}_3] - \Pr[\mathsf{E}_4]| \leq 2Q_{\mathsf{zt}}/q$.

Proof. We observe that the two games differ only when the adversary submits a query $f \in \mathbb{T}$ represented as Eq. (4.1) such that Eq. (4.2) holds, but there is (i,j) such that Eq. (4.4) holds. Note that for such f, i, and j, we have

$$\sum_{Z \in \mathcal{S}_{T,2,i,j}} a_Z Z(T_1^i, T_2^j, c_1^i \ldots c_L^i) = - \sum_{(i',j') \neq (i,j)} \sum_{Z \in \mathcal{S}_{T,2,i',j'}} a_Z Z(T_1^{i'}, T_2^{j'}, c_1^{i'} \ldots c_L^{i'}).$$

However, the above is impossible unless the left hand side equals to 0 since any monomial in $\mathcal{S}_{T,2,i,j}$ never appears in $\mathcal{S}_{T,2,i',j'}$ for $(i,j) \neq (i',j')$ (since the product $T_1^i T_2^j \neq T_1^{i'} T_2^{j'}$) even if we replace the formal variables $\{C_k^i\}_k$ and $\{C_k^{i'}\}_k$ with values $\{c_k^i\}_k$ and $\{c_k^{i'}\}_k$ in $\mathbb{Z}_q$. Therefore, the change made in this game is only conceptual and $\Pr[\mathsf{E}_3] = \Pr[\mathsf{E}_4]$. □

Lemma 4.6 ($\mathbf{Game}_4 \approx_c \mathbf{Game}_5$). *There exists a PPT adversary* $\mathcal{B}$ *against* Ada-INDr *security of* BGG^+ *such that* $|\Pr[\mathsf{E}_4] - \Pr[\mathsf{E}_5]| \leq Q_{\mathsf{ct}}^2 Q_{\mathsf{zt}} \cdot \left(\mathcal{A}^{\mathsf{Ada\text{-}INDr}}_{\mathsf{BGG}^+,\mathcal{B}}(1^\lambda) + 1/q\right)$.

Proof. We call a zero test query $f \in \mathbb{T}$ as *bad* if f is represented as Eq. (4.3) and there exists (i,j) such that i-th ciphertext for slot 1 is a challenge ciphertext (i.e., $b_0^i \neq b_1^i$) and the following equations hold:

$$\sum_{Z \in \mathcal{S}_{T,2,i,j}} a_Z Z(T_1^i, T_2^j, c_1^i, \ldots, c_L^i) = 0 \quad \text{and} \quad \sum_{Z \in \mathcal{S}_{T,2,i,j}} a_Z Z(T_1^i, T_2^j, C_1^i, \ldots, C_L^i) \neq 0. \tag{4.7}$$

It is easily seen that $\mathbf{Game}_4$ and $\mathbf{Game}_5$ are equivalent unless the adversary makes a bad query. To bound the probability of a bad query being issued, consider the following sequence of games. Below, we define F_{x} as the event that the challenger does not abort in $\mathbf{Game}_{4,\mathsf{x}}$.

$\mathbf{Game}_{4.0}$: This game is the same as $\mathbf{Game}_4$. However, the challenger checks whether $\mathcal{A}$ has made a bad query and aborts if not at the end of the game. By definition, the probability that $\mathcal{A}$ makes a bad query in $\mathbf{Game}_4$ is $\Pr[\mathsf{F}_0]$.

Game$_{4.1}$: In this game, we change the previous game so that the challenger picks a random guess k^* for the first bad query as $k^* \leftarrow [Q_{\mathsf{zt}}]$ at the beginning of the game. Furthermore, we change the game so that the challenger aborts if the k^*-th zero-test query is not the first bad query. Since k^* is chosen uniformly at random and independent from the view of $\mathcal{A}$, the guess is correct with probability $1/Q_{\mathsf{zt}}$ conditioned on F_0. Therefore, we have $\Pr[\mathsf{F}_1] = \Pr[\mathsf{F}_0]/Q_{\mathsf{zt}}$.

Game$_{4.2}$: This game is the same as the previous game except that the challenger aborts the game immediately after $\mathcal{A}$ makes the k^*-th zero-test query. Since the occurrence of F_1 is irrelevant to how the game proceeds after the k^*-th zero-test query, we clearly have $\Pr[\mathsf{F}_2] = \Pr[\mathsf{F}_1]$.

Game$_{4.3}$: In this game, we change the game so that the the challenger stop aborting even if a bad query occurs before the k^*-th zero test query. Furthermore, any query before the k^*-th one, regardless of whether bad or not, is answered as in **Game**$_5$. Since **Game**$_4$ and **Game**$_5$ proceed exactly the same until the first bad query and removing the abort condition simply increases the chance of making a bad query, we have $\Pr[\mathsf{F}_3] \geq \Pr[\mathsf{F}_2]$.

Game$_{4.4}$: In this game, we change the previous game so that the challenger picks $(i^*, j^*) \leftarrow [Q_{\mathsf{ct}}^2]$ uniformly at random at the beginning of the game. Furthermore, we change the abort condition so that the challenger aborts if Eq. (4.7) does not hold with respect to $(i, j) = (i^*, j^*)$ when the k^*-th zero-test query f is represented as Eq. (4.3). We note that the challenger does not check the equations with respect to other indices. Since there exists at least one pair of $(i, j) \in [Q_{\mathsf{ct}}^2]$ that satisfies Eq. (4.7) as long as F_4 occurs and (i^*, j^*) is chosen uniformly at random and independent from the view of $\mathcal{A}$, we have $\Pr[\mathsf{F}_4] \geq \Pr[\mathsf{F}_3]/Q_{\mathsf{ct}}^2$.

Game$_{4.5}$: In this game, we have the challenger abort if the (i^*, j^*)-th ciphertext queries were not made at the point when the k^*-th zero-test query was made. We claim that conditioned on F_5 happens, the challenger never aborts. To see this, we observe that if the (i^*, j^*)-th ciphertext queries have not been made, then terms that contain $T_1^{i^*}, T_2^{j^*}$ have not been given to $\mathcal{A}$ and there is no way to make a zero-test query for f such that $\sum_{Z \in \mathcal{S}_{T,2,i^*,j^*}} a_Z Z \neq 0$, since all terms in $\mathcal{S}_{T,2,i^*,j^*}$ are multiples of $T_1^{i^*} T_2^{j^*}$. We therefore have $\Pr[\mathsf{F}_5] = \Pr[\mathsf{F}_4]$.

Game$_{4.6}$: Recall in the previous game, Eq. (4.7) is checked with respect to $(i, j) = (i^*, j^*)$ and $\mathbf{c}^{i^*} = (c_1^{i^*}, \ldots, c_L^{i^*})$ is used there. In this hybrid, we only compute $\mathbf{c}^{i^*} \odot \mathbf{d}^{j^*}$ instead of $\mathbf{c}^{i^*}$ for the k^*-th challenge query. Equivalently, we only compute $c_k^{i^*}$ for k such that $d_k^{j^*} = 1$. We claim that the game is still well-defined. To see this, we first observe that the only place in the game where we need actual ciphertexts (not formal variables) is for the k^*-th zero-test query. Furthermore, for the k^*-th zero-test query, we observe that only the terms

$$\left\{ Z(T_1^{i^*}, T_2^{j^*}, c_1^{i^*}, \ldots c_L^{i^*}) \;\middle|\; Z \in \mathcal{S}_{T,2,i^*,j^*} \right\} = \left\{ c_k^{i^*} T_1^{i^*} T_2^{j^*} \;\middle|\; k \in [L] \text{ s.t. } d_k^{j^*} = 1 \right\}$$

are required, where the equality follows from the definition of $\mathcal{S}_{T,2,i^*,j^*}$. We therefore can see that the game is well-defined and the change is only conceptual. Hence we have $\Pr[\mathsf{F}_6] = \Pr[\mathsf{F}_5]$.

Game$_{4.7}$: Recall that $\mathbf{c}^{i^*} \odot \mathbf{d}^{j^*}$ constitutes a vector that is obtained by padding the ciphertext of BGG^+ for the attribute $(\mathbf{x}_1^{i^*}, \mathbf{x}_2^{j^*})$ with 1 and 0. In this game, we pick the ciphertext elements in $\mathbf{c}^{i^*} \odot \mathbf{d}^{j^*}$ uniformly at random from $\mathbb{Z}_q$ except for the positions that are fixed to be 0 or 1. As we prove in Lemma 4.7, there exists a PPT adversary $\mathcal{B}$ such that $\mathsf{Adv}^{\mathsf{Ada\text{-}INDr}}_{\mathsf{BGG}^+,\mathcal{B}}(1^\lambda) \geq |\Pr[\mathsf{F}_7] - \Pr[\mathsf{F}_6]|$.

We also show in Lemma 4.8 that $\Pr[\mathsf{F}_7] = 1/q$. This allows us to bound $\Pr[\mathsf{F}_0]$ as:

$$\Pr[\mathsf{F}_0] \leq Q_{\mathsf{ct}}^2 Q_{\mathsf{zt}} \cdot (\mathsf{Adv}^{\mathsf{Ada\text{-}INDr}}_{\mathsf{BGG}^+,\mathcal{B}}(1^\lambda) + 1/q),$$

where $\mathcal{B}$ is a PPT adversary. □

Lemma 4.7. *There exists a PPT adversary $\mathcal{B}$ such that* $\mathsf{Adv}^{\mathsf{Ada\text{-}INDr}}_{\mathsf{BGG}^+,\mathcal{B}}(1^\lambda) \geq |\Pr[\mathsf{F}_6] - \Pr[\mathsf{F}_7]|$.

Proof. We show that if $\mathcal{A}$ can distinguish **Game**$_{4.6}$ from **Game**$_{4.7}$, we can build another adversary $\mathcal{B}$ against Ada-INDr security of BGG^+. The adversary $\mathcal{B}$ acts as the challenger and simulates the game for $\mathcal{A}$.

Setup phase. At the beginning of the game, $\mathcal{B}$ is given 1^λ and the master public key of BGG^+ $(\mathbf{A}, \mathbf{B}, \mathbf{u})$ which it returns to $\mathcal{A}$. $\mathcal{B}$ also samples $(i^*, j^*) \leftarrow [Q_{\mathsf{ct}}^2]$, $k^* \leftarrow [Q_{\mathsf{zt}}]$, and $\beta \leftarrow \{0,1\}$ and keeps them secret.

Key Queries. Given the j-th secret key query for F^j made by $\mathcal{A}$, $\mathcal{B}$ makes a *secret key query* for F^j to its challenger and is given $\mathbf{r}$ sampled as $\mathbf{r} \leftarrow [\mathbf{A}\|\mathbf{B}_F]_\tau^{-1}(\mathbf{u})$.

Ciphertext Queries. When $\mathcal{A}$ makes ciphertext queries, $\mathcal{B}$ prepares handles for the ciphertexts components and returns them to $\mathcal{A}$. In more detail, $\mathcal{B}$ returns $\{T_1^i C_k^i W_k\}_{k\in[L]}$ for i-th ciphertext query in slot 1 and $\{T_2^i / W_k\}_{k\in[L]}$, s.t. $d_k^i = 1$ for i-th ciphertext query in slot 2.

Generic Group Queries. $\mathcal{B}$ honestly handles the queries for the generic group oracle corresponding to addition, negation, and multiplication (bilinear map) made by $\mathcal{A}$ by keeping track of the underlying encodings in $\mathbb{T}$ associated with the handles.

When $\mathcal{A}$ makes a k-th zero test query that refers to an element $f \in \mathbb{T}$, $\mathcal{B}$ returns 0 if f cannot be represented as Eq.(4.3) (i.e., there is $a_Z \neq 0$ such that $Z \in \mathcal{S}_{T,1}$). Otherwise, $\mathcal{B}$ proceeds as follows:

1. If f is the k-th zero-test query with $k < k^*$, it runs the following test for all $i, j \in [Q_{\mathsf{ct}}]$.
   (a) If the i-th ciphertext query is the challenge query (i.e., $b_0^i \neq b_1^i$), it checks whether $\sum_{Z\in\mathcal{S}_{T,2,i,j}} a_Z Z = 0$ or not (without replacing the formal variables $(C_1^i, \ldots, C_L^i)$ in Z with values $(c_1^i, \ldots, c_L^i)$).
   (b) If the i-th ciphertext query is not a challenge query, it checks whether the variables $\{C_k^i\}_k$ are already assigned values. If the values are already assigned, it will use the values. Otherwise, it samples $\mathbf{c}^i = \{c_k^i\}_k$ as in **Game**$_4$. We then check $\sum_{Z\in\mathcal{S}_{T,2,i,j}} a_Z Z(T_1^i, T_2^j, c_1^i, \ldots, c_L^i) = 0$ or not.

   If all the equations hold, $\mathcal{B}$ returns 1 to $\mathcal{A}$. Otherwise, it returns 0.
2. If f is the k^*-th zero-test query, $\mathcal{B}$ first checks whether the (i^*, j^*)-th ciphertext queries have already been made and the i^*-th query for slot 1 is the challenge query and aborts otherwise. It then requests the BGG^+ challenger for ciphertexts for attributes $(\mathbf{x}_1^{i^*}\|\mathbf{x}_2^{j^*})$ and message bit $b_\beta^{i^*}$. It constructs $\mathbf{c}^{i^*} \odot \mathbf{d}^{j^*}$ using the received BGG^+ ciphertexts. Then it checks whether Eq. (4.7) holds with respect to $(i, j) = (i^*, j^*)$. As we observed, the above check can be done only given $\mathbf{c}^{i^*} \odot \mathbf{d}^{j^*}$. It outputs 1 if they hold and 0 otherwise.

Analysis. Observe that $\mathcal{B}$ simulates $\mathbf{Game}_{4.6}$ if the challenge ciphertext for $\mathcal{B}$ is the real one and $\mathbf{Game}_{4.7}$ if it is chosen uniformly at random from the ciphertext space. Therefore, it can be seen that $\mathcal{B}$ outputs 1 with probability $\Pr[\mathsf{F}_6]$ if the BGG^+ challenger returned real ciphertexts and $\Pr[\mathsf{F}_7]$ if it returned random. Therefore, $\mathcal{B}$'s advantage against BGG^+ is $|\Pr[\mathsf{F}_6] - \Pr[\mathsf{F}_7]|$. This completes the proof of the lemma. □

Lemma 4.8. *We have* $\Pr[\mathsf{F}_7] = 1/q$.

Proof. We recall that F_7 occurs only when $\mathcal{A}$ makes a zero-test query that refers to a handle f that is represented as Eq. (4.3) and satisfies Eq.(4.7) with respect to $(i, j) = (i^*, j^*)$ for random $\{c_1^{i^*}, \dots c_L^{i^*}\}$. However, this can happen only with probability at most $1/q$ by the Schwartz-Zippel lemma because f is linear in the variables $C_1^{i^*}, \dots, C_L^{i^*}$. □

□

5 Two-Input ABE for NC_1 in Standard Model

In this section, we propose two input ABE construction for NC^1 in the standard model. The construction is shown to be strong very selective secure under the LWE assumption and a variant of bilinear KOALA assumption introduced in [AWY20], which is proven to hold under the bilinear generic group model, assuming function hiding BIPFE is available. Note that function hiding BIPFE can be instantiated from various standard assumptions on bilinear maps including SXDH and DLIN (See Sec. 2.3). The construction is similar to both our construction in Sec. 4 and the construction in [AWY20] in high level.

5.1 Assumption

Here, we introduce a variant of the bilinear KOALA assumption introduced in [AWY20], which in turn is a pairing group variant of the KOALA assumption introduced in [BW19]. The security of our two input ABE scheme in the standard model will be based on the assumption.

Definition 5.1 (Bilinear KOALA Assumption)**.** Let $\mathsf{Samp}_0 = \{\mathsf{Samp}_{0,\lambda}\}_\lambda$ be an efficient sampling algorithm that takes as input an integer q and outputs a string aux and $\mathsf{Samp}_1 = \{\mathsf{Samp}_{1,\lambda}\}_\lambda$ be an efficient sampling algorithm that takes as input an integer q and a string aux and outputs a matrix $\mathbf{V} \in \mathbb{Z}_q^{\ell_1 \times \ell_2}$ with $\ell_1 < \ell_2$. For an efficient adversary $\mathcal{A} = \{\mathcal{A}_\lambda\}$, let us define

$$\mathsf{Adv}^{\mathsf{BKOALA},\mathsf{dist}}_{\mathcal{A},\mathbb{G},\mathsf{Samp}}(\lambda) := |\Pr[\mathcal{A}_\lambda(\mathbb{G}, \mathsf{aux}, [\mathbf{sV}]_2) \to 1] - \Pr[\mathcal{A}_\lambda(\mathbb{G}, \mathsf{aux}, [\mathbf{r}]_2) \to 1]|.$$

where the probabilities are taken over the choice of $\mathbb{G} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, [1]_1, [1]_2) \leftarrow \mathsf{GroupGen}(1^\lambda)$, $\mathsf{aux} \leftarrow \mathsf{Samp}_0(q)$, $\mathbf{V} \leftarrow \mathsf{Samp}_1(q, \mathsf{aux})$, $\mathbf{s} \leftarrow \mathbb{Z}_q^{\ell_1}$, $\mathbf{r} \leftarrow \mathbb{Z}_q^{\ell_2}$, and the coin of $\mathcal{A}_\lambda$.

Furthermore, for an efficient adversary $\mathcal{B} = \{\mathcal{B}_\lambda\}_\lambda$, we also define

$$\mathsf{Adv}^{\mathsf{BKOALA},\mathsf{find}}_{\mathcal{B},\mathbb{G},\mathsf{Samp}}(\lambda) := \Pr[\mathcal{B}_\lambda(\mathbb{G}, \mathsf{aux}) \to \mathbf{x} \wedge \mathbf{V}\mathbf{x}^\top = \mathbf{0} \wedge \mathbf{x} \neq \mathbf{0}]$$

where the probability is taken over the choice of $\mathbb{G} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, [1]_1, [1]_2) \leftarrow \mathsf{GroupGen}(1^\lambda)$, $\mathsf{aux} \leftarrow \mathsf{Samp}_0(q)$, $\mathbf{V} \leftarrow \mathsf{Samp}_1(q, \mathsf{aux})$, and the coin of $\mathcal{B}_\lambda$.

We say that the bilinear KOALA assumption holds with respect to $\mathsf{GroupGen}$ and efficient samplers $\mathsf{Samp}_0, \mathsf{Samp}_1$ if for any efficient adversary $\mathcal{A}$, there exists another efficient adversary $\mathcal{B}$ and a polynomial function $Q(\lambda)$ such that

$$\mathsf{Adv}^{\mathsf{BKOALA},\mathsf{find}}_{\mathcal{B},\mathbb{G},\mathsf{Samp}}(\lambda) \geq \mathsf{Adv}^{\mathsf{BKOALA},\mathsf{dist}}_{\mathcal{A},\mathbb{G},\mathsf{Samp}}(\lambda)/Q(\lambda) - \mathrm{negl}(\lambda).$$

Remark 5.2. The above assumption is defined with respect to the non-uniform sampling algorithms $\mathsf{Samp}_0, \mathsf{Samp}_1$ and a non-uniform adversary $\mathcal{A}$. All the security assumptions and proofs in this paper except for the ones that use the above assumption can work in the uniform setting.

Remark 5.3 (Comparison with [AWY20]). Compared to the original version of the bilinear KOALA assumption [AWY20], our assumption allows Samp_0 to be an efficient sampler that possibly samples aux from some structured distribution, whereas Samp_0 is restricted to output a random string in their assumption. The reason why they restrict the distribution is to avoid a kind of attacks that embeds obfuscation into aux. In particular, as observed by the authors, if we relax the above setting so that aux is chosen along with $\mathbf{V}$, there is a concrete attack assuming sufficiently secure obfuscation. In more detail, let us consider a sampler that outputs random $\mathbf{V}$ along with auxiliary information $\mathsf{aux} = \mathcal{O}(C_{\mathbf{V}})$, which is an obfuscation of circuit $C_{\mathbf{V}}$ that takes as input group description $\mathbb{G}$ and elements $[\mathbf{v}]_2$ and returns whether $\mathbf{v}$ is in the space spanned by the rows of $\mathbf{V}$ or not. Using $\mathcal{O}(C_{\mathbf{V}})$, one can easily distinguish $[\mathbf{sV}]_2$ from $[\mathbf{r}]_2$ with high probability. However, an efficient adversary may not be able to find a vector $\mathbf{x} \neq \mathbf{0}$ that satisfies $\mathbf{Vx}^\top = \mathbf{0}$ even given $\mathcal{O}(C_{\mathbf{V}})$, if we use sufficiently strong obfuscator to obfuscate the circuit $C_{\mathbf{V}}$.[6] This specific attack does not work against our assumption above, since $\mathbf{V}$ is chosen *after* aux, rather than chosen at the same time. However, as a safeguard against the future attacks, we assume the assumption to hold for some specific samplers Samp_0 and Samp_1 rather than for general Samp_0 and Samp_1. We note that [AWY20] assumes that the assumption holds for all the efficient samplers Samp_1 rather than a specific one, while restricting Samp_0 to the specific sampler that outputs a random string.

To justify the assumption, [AWY20] proves that the assumption holds in the generic group model. Though they prove the theorem for the case where aux is chosen uniformly at random, the proof does not depend on this fact and the same proof works for the case where aux is chosen from general distributions. Namely, we have the following theorem.

Theorem 5.4 (Adapted from [AWY20])**.** *The bilinear KOALA assumption holds under the bilinear generic group model with respect to all efficient samplers* Samp_0 *and* Samp_1*, where* $\mathcal{A}$ *has access to the generic group oracles but* Samp_0 *and* Samp_1 *do not.*

The following lemma is from [AWY20] with slightly different formulation. The lemma essentially says that for a sampler that outputs a set of vectors such that the vectors are individually pseudorandom but mutually correlated, it holds that the vectors appear mutually pseudorandom when they are lifted to the exponent and randomized by vector-wise randomness assuming the bilinear KOALA assumption for the related samplers.

Lemma 5.5 (Adapted from Theorem 4.7 in [AWY20])**.** *Let* $\mathsf{Samp}_0 = \{\mathsf{Samp}_{0,\lambda}\}_\lambda$ *be an efficient sampling algorithm that takes as input an integer* q *and outputs a string* aux *and* $\mathsf{Samp}_1 = \{\mathsf{Samp}_{1,\lambda}\}_\lambda$ *be an efficient sampling algorithm that takes as input an integer* q *and a string* aux *and outputs a set of vectors* $\{\mathbf{u}^{(j)} \in \mathbb{Z}_q^m\}_{j\in[t]}$*. For an efficient adversary* $\mathcal{A} = \{\mathcal{A}_\lambda\}_\lambda$ *and* $i := i(\lambda) \in \mathbb{N}$*, let us assume that*

$$|\Pr[\mathcal{A}_\lambda(\mathbb{G}, \mathsf{aux}, \mathbf{u}^{(i)}) \to 1] - \Pr[\mathcal{A}_\lambda(\mathbb{G}, \mathsf{aux}, \mathbf{v}) \to 1]| \tag{5.1}$$

is negligible, where the probabilities are taken over the choice of $\mathbb{G} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, [1]_1, [1]_2) \leftarrow \mathsf{GroupGen}(1^\lambda)$*,* $\mathsf{aux} \leftarrow \mathsf{Samp}_0(q)$*,* $\{\mathbf{u}^{(j)}\}_{j\in[t]} \leftarrow \mathsf{Samp}_\lambda(q, \mathsf{aux})$*,* $\mathbf{v} \leftarrow \mathbb{Z}_q^m$*, and the coin of* $\mathcal{A}_\lambda$*. In the above, we set* $\mathbf{u}^{(i)} := \mathbf{v}$ *if* $i > t$*. Furthermore, assume that*

[6] The explanation on the attack is taken verbatim from [AWY20, Remark 4.3].

the bilinear KOALA assumption holds with respect to $\mathsf{GroupGen}$, Samp_0, *and* Samp'_1 *that runs* Samp_1 *to obtain* $\{\mathbf{u}^{(j)}\}_{j\in[t]}$ *and then outputs*

$$\mathbf{V} = \begin{bmatrix} 1 & \mathbf{u}^{(1)} & & & & \\ & & 1 & \mathbf{u}^{(2)} & & \\ & & & & \ddots & \\ & & & & 1 & \mathbf{u}^{(t)} \end{bmatrix} \in \mathbb{Z}_q^{t\times(1+m)t}. \tag{5.2}$$

Then, for any efficient adversary $\mathcal{B} = \{\mathcal{B}_\lambda\}$,

$$\left|\Pr\left[\mathcal{B}_\lambda\left(\begin{array}{c}\mathbb{G}, \mathsf{aux},\\ \{[\gamma^{(j)}]_2, [\gamma^{(j)}\mathbf{u}^{(j)}]_2\}_{j\in[t]}\end{array}\right)\to 1\right] - \Pr\left[\mathcal{B}_\lambda\left(\begin{array}{c}\mathbb{G}, \mathsf{aux},\\ \{[\gamma^{(j)}]_2, [\mathbf{v}^{(j)}]_2\}_{j\in[t]}\end{array}\right)\to 1\right]\right|, \tag{5.3}$$

is negligible, where the probabilities are taken over the choice of $\mathbb{G}$, $\mathsf{aux}\leftarrow\mathsf{Samp}_0(q)$, $\{\mathbf{u}^{(j)}\}_{j\in[t]}\leftarrow \mathsf{Samp}_1(q,\mathsf{aux})$, $\gamma^{(j)}\leftarrow\mathbb{Z}_q$, $\mathbf{v}^{(j)}\leftarrow\mathbb{Z}_q^m$ *for* $j\in[t]$, *and the coin of* $\mathcal{B}_\lambda$.

The lemma is shown for the case where aux is chosen uniformly at random in [AWY20], but the same proof works for our more general setting.

5.2 Construction

Here, we show our construction of two input ABE for NC^1 in the standard model. The function class that the scheme supports is exactly the same as that of Sec. 4. In the construction, we use a BIPFE scheme $\mathsf{BIPFE} = (\mathsf{BIPFE.Setup}, \mathsf{BIPFEKeyGen}, \mathsf{BIPFE.Enc}, \mathsf{BIPFE.Dec})$. We refer to Sec. 2.3 for the definition and instantiations of BIPFE.

$\mathsf{Setup}(1^\lambda)$: On input 1^λ, the setup algorithm defines the parameters $n = n(\lambda)$, $m = m(\lambda)$, noise distributions χ over $\mathbb{Z}$, $\tau_0 = \tau_0(\lambda)$, $\tau = \tau(\lambda)$, and $B = B(\lambda)$ as specified in Sec. 2.5. It samples a group description $\mathbb{G} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, [1]_1, [1]_2)$. It then proceeds as follows.

1. Sample BGG^+ scheme:
   (a) Sample $(\mathbf{A}, \mathbf{A}_{\tau_0}^{-1}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$ such that $\mathbf{A}\in\mathbb{Z}_q^{n\times m}$.
   (b) Sample random matrix $\mathbf{B} = (\mathbf{B}_1, \ldots, \mathbf{B}_{2\ell}) \leftarrow (\mathbb{Z}_q^{n\times m})^{2\ell}$ and a random vector $\mathbf{u}\leftarrow\mathbb{Z}_q^n$.
2. Sample BIPFE instances $\mathsf{BIPFE.msk}_1 \leftarrow \mathsf{BIPFE.Setup}(1^\lambda, 1^{m(\ell+1)+2}, 1^2)$ and $\mathsf{BIPFE.msk}_2 \leftarrow \mathsf{BIPFE.Setup}(1^\lambda, 1^{m\ell}, 1^3)$.
3. Output

$$\mathsf{pp} = (\mathbf{A}, \mathbf{B}, \mathbf{u}), \quad \mathsf{msk} = \big([1]_1, [1]_2, \mathbf{A}_\tau^{-1}, \{\mathsf{BIPFE.msk}_i\}_{i\in\{1,2\}}\big).$$

$\mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, F)$: Given input the public parameters pp, master secret key msk and a circuit F, compute BGG^+ function key for circuit F as follows:

1. Compute $\mathbf{H}_F = \mathsf{EvalF}(\mathbf{B}, F)$ and $\mathbf{B}_F = \mathbf{B}\mathbf{H}_F$.
2. Compute $[\mathbf{A}\|\mathbf{B}_F]_\tau^{-1}$ from $\mathbf{A}_{\tau_0}^{-1}$ and sample $\mathbf{r}\in\mathbb{Z}^{2m}$ as $\mathbf{r}^\top \leftarrow [\mathbf{A}\|\mathbf{B}_F]_\tau^{-1}(\mathbf{u}^\top)$.
3. Output the secret key $\mathsf{sk}_F := \mathbf{r}$.

$\mathsf{Enc}_1(\mathsf{pp}, \mathsf{msk}, \mathbf{x}_1, b)$: Given input the public parameters pp, master secret key msk, attribute vector $\mathbf{x}_1$, message bit b, encryption for slot 1 is defined as follows:

1. Sample LWE secret $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ and noises $e_0 \leftarrow \chi$, $\mathbf{e} \leftarrow \chi^m$, $\mathbf{e}_L, \mathbf{e}_R \leftarrow (\widetilde{\chi^m})^\ell$, where $\widetilde{\chi^m}$ is defined as in Sec. 2.4.
2. Compute

$$\mathbf{c}_{1,1} := \left(1,\ \mathbf{sA} + \mathbf{e},\ \mathbf{su}^\top + e_0 + \left\lceil \frac{q}{2} \right\rceil \cdot b,\ \mathbf{s}\left(\mathbf{B}_L - \mathbf{x}_1 \otimes \mathbf{G}\right) + \mathbf{e}_L\right),\ \mathbf{c}_{1,2} := \mathbf{0}_{m(\ell+1)+2},$$
$$\mathbf{c}_{2,1} := \mathbf{sB}_R + \mathbf{e}_R, \quad \mathbf{c}_{2,2} := -\mathbf{1}_\ell \otimes \mathbf{sG}, \quad \mathbf{c}_{2,3} := \mathbf{0}_{m\ell}$$

   where we define $\mathbf{B}_L := [\mathbf{B}_1, \ldots, \mathbf{B}_\ell]$ and $\mathbf{B}_R := [\mathbf{B}_{\ell+1}, \ldots, \mathbf{B}_{2\ell}]$.
3. Set $\mathbf{C}_1 = (\mathbf{c}_{1,1}^\top, \mathbf{c}_{1,2}^\top)$ and $\mathbf{C}_2 = (\mathbf{c}_{2,1}^\top, \mathbf{c}_{2,2}^\top, \mathbf{c}_{2,3}^\top)$.
4. For $i = 1, 2$, compute $\mathsf{BIPFE.ct}_i := \mathsf{BIPFE.Enc}(\mathsf{BIPFE.msk}_i, [\mathbf{C}_i]_1)$.
5. Output $\mathsf{ct}_1 = (\mathsf{BIPFE.ct}_1, \mathsf{BIPFE.ct}_2)$.

$\mathsf{Enc}_2(\mathsf{pp}, \mathsf{msk}, \mathbf{x}_2)$: Given input the public parameters pp, master secret key msk, attribute vector $\mathbf{x}_2$, encryption for slot 2 is defined as follows:

1. Sample $t \leftarrow \mathbb{Z}_q$.
2. Compute

$$\mathbf{d}_{1,1} := t \cdot \mathbf{1}_{m(\ell+1)+2}, \quad \mathbf{d}_{1,2} := \mathbf{0}_{m(\ell+1)+2}$$
$$\mathbf{d}_{2,1} := t \cdot \mathbf{1}_{m\ell},\ \mathbf{d}_{2,2} := t\left(\mathbf{x}_2 \otimes \mathbf{1}_m\right),\ \mathbf{d}_{2,3} := \mathbf{0}_{m\ell}.$$

3. Set $\mathbf{D}_1 = (\mathbf{d}_{1,1}^\top, \mathbf{d}_{1,2}^\top)$ and $\mathbf{D}_2 = (\mathbf{d}_{2,1}^\top, \mathbf{d}_{2,2}^\top, \mathbf{d}_{2,3}^\top)$.
4. For $i = 1, 2$, compute $\mathsf{BIPFE.sk}_i := \mathsf{BIPFE.KeyGen}(\mathsf{BIPFE.msk}_i, [\mathbf{D}]_2)$.
5. Output $\mathsf{ct}_2 = (\mathsf{BIPFE.sk}_1, \mathsf{BIPFE.sk}_2)$.

$\mathsf{Dec}(\mathsf{pp}, \mathsf{sk}_F, \mathsf{ct}_1, \mathsf{ct}_2)$: The decryption algorithm takes as input the public parameters pp, the secret key sk_F for circuit F and ciphertexts ct_1 and ct_2 corresponding to the two attributes $\mathbf{x}_1$ and $\mathbf{x}_2$ and proceeds as follows:

1. Parse the ciphertext:
   Parse the ciphertexts as $\mathsf{ct}_1 \to (\mathsf{BIPFE.ct}_1, \mathsf{BIPFE.ct}_2)$ and $\mathsf{ct}_2 \to (\mathsf{BIPFE.sk}_1, \mathsf{BIPFE.sk}_2)$.
2. Decrypt the BIPFE ciphertexts:
   Compute $[\mathbf{w}_i]_T = \mathsf{BIPFE.Dec}(\mathsf{BIPFE.sk}_i, \mathsf{BIPFE.ct}_i)$ for $i = 1, 2$.
3. Reorganize the obtained vector:
   Let $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$. Reorganize $[\mathbf{w}_i]_T$ for $i = 1, 2$ to obtain:

$$[v_0, \mathbf{v}_{2\ell+1}, v_{2\ell+2}, \mathbf{v}_{\mathbf{x}_1}]_T := [\mathbf{w}_1]_T, \quad [\mathbf{v}_{\mathbf{x}_2}]_T := [\mathbf{w}_2]_T, \quad [\mathbf{v}_{\mathbf{x}}]_T := [\mathbf{v}_{\mathbf{x}_1}, \mathbf{v}_{\mathbf{x}_2}]_T,$$

   where $v_0 \in \mathbb{Z}_q$, $\mathbf{v}_{2\ell+1} \in \mathbb{Z}_q^m$, $v_{2\ell+2} \in \mathbb{Z}_q$, and $\mathbf{v}_{\mathbf{x}_1}, \mathbf{v}_{\mathbf{x}_2} \in \mathbb{Z}_q^{m\ell}$.
4. Evaluate function on BGG^+ ciphertexts in exponent:
   Compute $\widehat{\mathbf{H}}_{F,\mathbf{x}} = \mathsf{EvalFX}(F, \mathbf{x}, \mathbf{B})$.

5. Perform BGG^+ decryption in the exponent:
Form $\mathbf{r} = (\mathbf{r}_1 \in \mathbb{Z}_q^m, \mathbf{r}_2 \in \mathbb{Z}_q^m)$. Then compute

$$[v']_T := [(v_{2\ell+2} - \left(\mathbf{v}_{2\ell+1}\mathbf{r}_1^\top + \mathbf{v}_{\mathbf{x}}\widehat{\mathbf{H}}_{F,\mathbf{x}}\mathbf{r}_2^\top\right))]_T.$$

6. Recover exponent via brute force if $F(\mathbf{x}) = 0$:
Find $\eta \in [-B, B] \cup [-B + \lceil q/2 \rceil, B + \lceil q/2 \rceil]$ such that $[v_0]_T^\eta = [v']_T$ by brute-force search. If there is no such η, output $\perp$. To speed up the operation, one can employ the baby-step giant-step algorithm.

7. Output 0 if $\eta \in [-B, B]$ and 1 if $[-B + \lceil q/2 \rceil, B + \lceil q/2 \rceil]$.

Correctness: Correctness is argued by observing that vectors $\{\mathbf{w}_i\}_{i\in\{1,2\}}$ form the randomized version of the BGG+ ciphertext w.r.t $\mathbf{s}$ on the exponent. Namely, we have

$$\begin{aligned}\mathbf{w}_1 &= \mathbf{c}_{1,1} \odot \mathbf{d}_{1,1} + \mathbf{c}_{1,2} \odot \mathbf{d}_{1,2} \\ &= t\left(1,\ \mathbf{sA} + \mathbf{e},\ \mathbf{su}^\top + e_0 + \left\lceil \frac{q}{2} \right\rceil \cdot b,\ \mathbf{s}\left(\mathbf{B}_L - \mathbf{x}_1 \otimes \mathbf{G}\right) + \mathbf{e}_L\right)\end{aligned}$$

and

$$\begin{aligned}\mathbf{w}_2 &= \mathbf{c}_{2,1} \odot \mathbf{d}_{2,1} + \mathbf{c}_{2,2} \odot \mathbf{d}_{2,2} + \mathbf{c}_{2,3} \odot \mathbf{d}_{2,3} \\ &= t\left(\mathbf{sB}_R + \mathbf{e}_R\right) - t\left(\mathbf{1}_\ell \otimes \mathbf{sG}\right) \odot \left(\mathbf{x}_2 \otimes \mathbf{1}_m\right) \\ &= t\left(\mathbf{s}\left(\mathbf{B}_R - \mathbf{x}_2 \otimes \mathbf{G}\right) + \mathbf{e}_R\right),\end{aligned}$$

where we used $(\mathbf{1}_\ell \otimes \mathbf{sG}) \odot (\mathbf{x}_2 \otimes \mathbf{1}_m) = \mathbf{s}\left(\mathbf{x}_2 \otimes \mathbf{G}\right)$ in the third equation above. Having established above, the rest of the argument for proving the correctness is exactly the same as Sec. 4 and thus omitted.

5.3 Security

We prove the security of the construction via following theorem:

Theorem 5.6. *Our* 2ABE *scheme for function class* NC_1 *satisfies strong very selective security assuming that* BIPFE *satisfies function hiding security and the bilinear KOALA assumption for certain samplers*[7] *and the LWE assumption hold.*

Proof. Consider a PPT adversary $\mathcal{A}$ that makes at most $Q_{\mathsf{ct}}(\lambda)$ ciphertext queries (in both slots) and $Q_{\mathsf{kq}}(\lambda)$ key queries during the game. We denote the event that $\mathcal{A}$ outputs 1 at the end of $\mathbf{Game}_{\mathrm{x}}$ as E_{x}.

$\mathbf{Game}_0$: This is the real game with $\beta = 0$. At the beginning of the game, the adversary declares its challenge queries $\{(\mathbf{x}_1^{(i)}, b_0^{(i)}, b_1^{(i)})\}_{i\in[Q_{\mathsf{ct}}]}$ for the first slot, $\{\mathbf{x}_2^{(i)}\}_{i\in[Q_{\mathsf{ct}}]}$ for the second slot, and the key queries $\{F^{(j)}\}_{j\in[Q_{\mathsf{kq}}]}$. Then, the challenger samples the public parameters $\mathsf{pp} = (\mathbf{A}, \mathbf{B}, \mathbf{u})$ and master secret key $\mathsf{msk} = \left([1]_1, [1]_2, \mathbf{A}_\tau^{-1}, \{\mathsf{BIPFE.msk}_i\}_{i\in\{1,2\}}\right)$ as described in the scheme. Then, it computes the ciphertexts $\mathsf{ct}_1^{(i)} \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{msk}, \mathbf{x}_1^{(i)}, b_0^{(i)})$, $\mathsf{ct}_2^{(i)} \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{msk}, \mathbf{x}_2^{(i)})$, for $i \in [Q_{\mathsf{ct}}]$ and secret keys $\mathsf{sk}^{(j)} \leftarrow \mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, F^{(j)})$ for $j \in [Q_{\mathsf{kq}}]$ and returns $\mathsf{pp}, \{\mathsf{ct}_1^{(i)}, \mathsf{ct}_2^{(i)}\}_i, \{\mathsf{sk}^{(j)}\}_j$ to $\mathcal{A}$.

[7] We refer to the proof of Lemma 5.8 for the description of the sampler.

Game$_j$: This game is defined for $j \leq Q_{\mathsf{ct}}$. In this game, $\mathsf{ct}_1^{(i)}$ is computed as

$$\mathsf{ct}_1^{(i)} \leftarrow \begin{cases} \mathsf{Enc}(\mathsf{pp}, \mathsf{msk}, \mathbf{x}_1^{(i)}, b_1^{(i)}) & \text{if } i \leq j \\ \mathsf{Enc}(\mathsf{pp}, \mathsf{msk}, \mathbf{x}_1^{(i)}, b_0^{(i)}) & \text{if } i > j \end{cases}$$

Except for the above, the game is the same as **Game**$_0$.

We have that **Game**$_{Q_{\mathsf{ct}}}$ equals to the real game with $\beta = 1$. Therefore, we have to show that $|\Pr[\mathsf{E}_0] - \Pr[\mathsf{E}_{Q_{\mathsf{ct}}}]|$ is negligible. To do so, it suffices to show that $|\Pr[\mathsf{E}_{j-1}] - \Pr[\mathsf{E}_j]|$ is negligible for all $j \in [Q_{\mathsf{ct}}]$. We introduce the following sequence of games to prove this. In the following, we can assume that the j-th ciphertext for slot 1 is the challenge ciphertext (i.e., $b_0^{(j)} \neq b_1^{(j)}$), since otherwise **Game**$_{j-1}$ and **Game**$_j$ are equivalent.

Game$_{j-1,0}$: This game is the same as **Game**$_{j-1}$. To fix the notation, we describe how the challenger answers the challenge queries below.

1. Slot 1 ciphertext queries: To answer a slot 1 ciphertext request for $(\mathbf{x}_1^{(i)}, b_0^{(i)}, b_1^{(i)})$, the challenger samples $\mathbf{s}^{(i)}$, $\mathbf{e}^{(i)}$, $\mathbf{e}_L^{(i)}$, $\mathbf{e}_R^{(i)}$, and $e_0^{(i)}$ as specified and computes the vectors as below:

   $$\mathbf{c}_{1,1}^{(i)} := \left(1,\ \mathbf{s}^{(i)}\mathbf{A} + \mathbf{e}^{(i)},\ \mathbf{s}^{(i)}\mathbf{u}^\top + e_0^{(i)} + \left\lceil \frac{q}{2} \right\rfloor \cdot b^{(i)},\ \mathbf{s}^{(i)}\left(\mathbf{B}_L - \mathbf{x}_1^{(i)} \otimes \mathbf{G}\right) + \mathbf{e}_L^{(i)}\right),$$
   $$\mathbf{c}_{1,2}^{(i)} = \mathbf{0}_{m(\ell+1)+2}$$

   where $b^{(i)} = 1$ if $i \leq j-1$ and $b^{(i)} = 0$ otherwise,

   $$\mathbf{c}_{2,1}^{(i)} := \mathbf{s}^{(i)}\mathbf{B}_R + \mathbf{e}_R^{(i)}, \quad \mathbf{c}_{2,2}^{(i)} = -\mathbf{1}_\ell \otimes \mathbf{s}^{(i)}\mathbf{G}, \quad \mathbf{c}_{2,3}^{(i)} := \mathbf{0}_{m\ell}.$$

   Then, the challenger sets $\mathbf{C}^{(i)}$ and computes $\mathsf{ct}_1^{(i)}$ as specified using the vectors.
2. Slot 2 ciphertext queries: To answer a slot 2 ciphertext request for $\mathbf{x}_2^{(i)}$, the challenger samples $t^{(i)}$ as specified and computes the vectors as below:

   $$\mathbf{d}_{1,1}^{(i)} = t^{(i)}\mathbf{1}_{m(\ell+1)+2}, \quad \mathbf{d}_{1,2}^{(i)} = \mathbf{0}_{m(\ell+1)+2},$$
   $$\mathbf{d}_{2,1}^{(i)} = t^{(i)}\mathbf{1}_{m\ell}, \quad \mathbf{d}_{2,2}^{(i)} = t^{(i)}\left(\mathbf{x}_2^{(i)} \otimes \mathbf{1}_m\right), \quad \mathbf{d}_{2,3}^{(i)} = \mathbf{0}_{m\ell}.$$

   Then, the challenger sets $\mathbf{D}^{(i)}$ and computes $\mathsf{ct}_2^{(i)}$ as specified using the vectors.

By definition, we have

$$\Pr[\mathsf{E}_{j-1}] = \Pr[\mathsf{E}_{j-1,0}].$$

Game$_{j-1,1}$: In this game, we change how the challenger computes the j-th ciphertext for slot 1 and all the ciphertexts for slot 2. In particular, the challenger sets $\mathbf{C}^{(j)}$ by setting the vectors as

$$\mathbf{c}_{1,1}^{(j)} = \mathbf{0}_{m(\ell+1)+2}, \quad \mathbf{c}_{1,2}^{(j)} = \mathbf{1}_{m(\ell+1)+2},$$
$$\mathbf{c}_{2,1}^{(j)} = \mathbf{c}_{2,2}^{(j)} = \mathbf{0}_{m\ell}, \quad \mathbf{c}_{2,3}^{(j)} = \mathbf{1}_{m\ell}.$$

Furthermore, we change the vectors $\mathbf{d}_{1,2}^{(i)}$ and $\mathbf{d}_{2,3}^{(i)}$ for all $i \in [Q_{\mathsf{ct}}]$ as follows:

$$\mathbf{d}_{1,2}^{(i)} = t^{(i)}\left(1, \mathbf{s}^{(j)}\mathbf{A} + \mathbf{e}^{(j)}, \mathbf{s}^{(j)}\mathbf{u}^\top + e_0^{(j)} + \left\lceil \frac{q}{2} \right\rfloor \cdot b_0^{(j)}, \mathbf{s}^{(j)}\left(\mathbf{B}_L - \mathbf{x}_1^{(j)} \otimes \mathbf{G}\right) + \mathbf{e}_L^{(j)}\right),$$
$$\mathbf{d}_{2,3}^{(i)} = t^{(i)}\left(\mathbf{s}^{(j)}\left(\mathbf{B}_R - \mathbf{x}_2^{(i)} \otimes \mathbf{G}\right) + \mathbf{e}_R^{(j)}\right).$$

We note that other terms in $\mathbf{D}_1^{(i)}$ and $\mathbf{D}_2^{(i)}$ are unchanged. We show in Lemma 5.7 that

$$|\Pr[\mathsf{E}_{j-1,0}] - \Pr[\mathsf{E}_{j-1,1}]| \leq \mathrm{negl}(\lambda).$$

Game$_{j-1,2}$: In this game, we further change $\mathbf{d}_{1,2}^{(i)}$ and $\mathbf{d}_{2,3}^{(i)}$ for all $i \in [Q_{\mathsf{ct}}]$ as follows:

$$\mathbf{d}_{1,2}^{(i)} = t^{(i)} \left(1, \mathbf{c}^{(i)}, c_0^{(i)}, \mathbf{c}_L^{(i)}\right), \quad \mathbf{d}_{2,3}^{(i)} = t^{(i)} \cdot \mathbf{c}_R^{(i)},$$

where $\mathbf{c}^{(i)} \leftarrow \mathbb{Z}_q^m$, $c_0^{(i)} \leftarrow \mathbb{Z}_q$, $\mathbf{c}_L^{(i)}, \mathbf{c}_R^{(i)} \leftarrow \mathbb{Z}_q^{m\ell}$ are freshly chosen for each i. We show in Lemma 5.8 that

$$|\Pr[\mathsf{E}_{j-1,1}] - \Pr[\mathsf{E}_{j-1,2}]| \leq \mathrm{negl}(\lambda).$$

Game$_{j-1,3}$: This game is the same as **Game**$_j$. We show in Lemma 5.9 that

$$|\Pr[\mathsf{E}_{j-1,2}] - \Pr[\mathsf{E}_{j-1,3}]| \leq \mathrm{negl}(\lambda).$$

Indistinguishability of Hybrids. We next argue that consecutive hybrids are indistinguishable.

Lemma 5.7 (**Game**$_{j-1,0} \approx_c$ **Game**$_{j-1,1}$)**.** *We have* $|\Pr[\mathsf{E}_{j-1,0}] - \Pr[\mathsf{E}_{j-1,1}]| \leq \mathrm{negl}(\lambda)$.

Proof. We observe that $\mathbf{C}_1^{(i)} \boxdot \mathbf{D}_1^{(i')}$ and $\mathbf{C}_2^{(i)} \boxdot \mathbf{D}_2^{(i')}$ in **Game**$_{j-1,0}$ are the same as those in **Game**$_{j-1,1}$ for all the combinations of $i, i' \in [Q_{\mathsf{ct}}]$. We can change $\mathbf{C}_1^{(i)}$ and $\mathbf{D}_1^{(i')}$ in **Game**$_{j-1,0}$ to those of **Game**$_{j-1,1}$ in a computationally indistinguishable way by a straightforward reduction to the function privacy of BIPFE. In the reduction, the reduction algorithm samples msk except for $\mathsf{BIPFE.msk}_0$ and simulates BIPFE secret keys and ciphertexts by the oracle queries. We can also change $\mathbf{C}_2^{(i)}$ and $\mathbf{D}_2^{(i')}$ using the security of BIPFE again. □

Lemma 5.8 (**Game**$_{j-1,1} \approx_c$ **Game**$_{j-1,2}$)**.** *We have* $|\Pr[\mathsf{E}_{j-1,1}] - \Pr[\mathsf{E}_{j-1,2}]| \leq \mathrm{negl}(\lambda)$.

Proof. We construct an adversary $\mathcal{B}$ against the bilinear KOALA assumption with respect to certain samplers that are specified later assuming an adversary $\mathcal{A}$ that distinguishes the two games. We first fix the challenge queries $\{(\mathbf{x}_1^{(i)}, b_0^{(i)}, b_1^{(i)})\}_{i\in[Q_{\mathsf{ct}}]}$ and $\{\mathbf{x}_2^{(i)}\}_{i\in[Q_{\mathsf{ct}}]}$ and secret key queries $\{F^{(i)}\}_{i\in[Q_{\mathsf{kq}}]}$ that maximizes the distinguishing advantage of $\mathcal{A}$. This is possible because the queries are only dependent on the security parameter and the randomness of $\mathcal{A}$, both of which can be hardwired into $\mathcal{A}$ in the non-uniform setting.[8]

We first consider the sampling algorithm Samp_0 that works as follows. In the following, let $\mathsf{BGG}^+ = (\mathsf{BGG}^+.\mathsf{Setup}, \mathsf{BGG}^+.\mathsf{KeyGen}, \mathsf{BGG}^+.\mathsf{Enc}, \mathsf{BGG}^+.\mathsf{Dec})$ be the kpABE scheme by [BGG+14] that is introduced in Sec. 2.5.

$\mathsf{Samp}_0(1^\lambda, q)$: Given the security parameter 1^λ and the modulus q, it works as follows.

(a) Run $(\mathsf{BGG}^+.\mathsf{mpk}, \mathsf{BGG}^+.\mathsf{msk}) \leftarrow \mathsf{BGG}^+.\mathsf{Setup}(1^\lambda)$ for the circuit class $\mathcal{C}_{2\ell(\lambda),d(\lambda)}$ and modulus q.

(b) Run $\mathsf{BGG}^+.\mathsf{sk}^{(i)} \leftarrow \mathsf{BGG}^+.\mathsf{KeyGen}(\mathsf{BGG}^+.\mathsf{mpk}, \mathsf{BGG}^+.\mathsf{msk}, F^{(i)})$ for $i \in [Q_{\mathsf{kq}}]$.

(c) Output $\mathsf{aux} := (\mathsf{BGG}^+.\mathsf{mpk}, \{\mathsf{BGG}^+.\mathsf{sk}^{(i)}\}_{i\in[Q_{\mathsf{kq}}]})$.

We then define Samp_1 as follows:

[8] This is the only proof in the paper where we need the non-uniform reduction algorithm.

$\mathsf{Samp}_1(\mathsf{aux}, q)$: Given the auxiliary information $\mathsf{aux} = (\mathsf{BGG}^+.\mathsf{mpk}, \{\mathsf{BGG}^+.\mathsf{sk}^{(i)}\}_{i\in[Q_{\mathsf{kq}}]})$ and the modulus q, it works as follows.

(a) Sample $\mathbf{z} \leftarrow \mathbb{Z}_q^n$, $e_0 \leftarrow \chi$, $\mathbf{e} \leftarrow \chi^m$, and $\mathbf{e}_k \leftarrow \widetilde{\chi^m}$ for $k \in [\ell]$.

(b) Compute

$$\psi_{2\ell+1} := \mathbf{z}\mathbf{A} + \mathbf{e} \in \mathbb{Z}_q^m, \quad \psi_{2\ell+2} := \mathbf{z}\mathbf{u}^\top + e_0 + b_0^{(j)}\lceil q/2 \rceil \in \mathbb{Z}_q,$$
$$\text{For all } k \in [2\ell],\ b \in \{0,1\}, \psi_{k,b} := \mathbf{z}(\mathbf{B} - b\mathbf{G}) + \mathbf{e}_k \in \mathbb{Z}_q^m$$

(c) Set $\mathbf{u}^{(i)} := \left(\psi_{2\ell+1}, \psi_{2\ell+2}, \left\{\psi_{k,x_{1,k}^{(j)}}\right\}_{k\in[\ell]}, \left\{\psi_{k,x_{2,k}^{(i)}}\right\}_{k\in[\ell]}\right)$.

(d) Output $\{\mathbf{u}^{(i)}\}_{i\in[Q_{\mathsf{ct}}]}$

We can observe that each $\mathbf{u}^{(i)}$ is distributed as a BGG^+ ciphertext for message $b_0^{(j)}$ and attribute $(\mathbf{x}_1^{(j)}, \mathbf{x}_2^{(i)})$, even though when we consider the joint distribution of $\{\mathbf{u}^{(i)}\}_i$, the set of vectors is mutually correlated. This along with the fact that the j-th ciphertext is the challenge ciphertext (i.e., there is no i and i' such that $F^{(i')}(\mathbf{x}_1^{(j)}, \mathbf{x}_2^i) = 0$) imply that each $\mathbf{u}^{(i)}$ is pseudorandom given aux by $\mathsf{Sel\text{-}INDr}$ security of BGG^+ (as per Definition 2.5). Therefore, assuming the bilinear KOALA assumption with respect to $\mathsf{GroupGen}$, Samp_0, and Samp_1', where Samp_1' is defined as in Lemma 5.5 from Samp_1 above[9], we have that the following distributions are computationally indistinguishable:

$$\left(\begin{array}{c}\mathbb{G}, \mathsf{aux}, \\ \left\{[\gamma^{(i)}]_2, [\gamma^{(i)}\mathbf{u}^{(i)}]_2\right\}_{i\in[Q_{\mathsf{ct}}]}\end{array}\right) \approx_c \left(\begin{array}{c}\mathbb{G}, \mathsf{aux}, \\ \left\{[\gamma^{(i)}]_2, [\mathbf{v}^{(i)}]_2\right\}_{i\in[Q_{\mathsf{ct}}]}\end{array}\right)$$

where $\mathbb{G}$ is chosen by $\mathsf{GroupGen}(1^\lambda)$, $\mathsf{aux} \leftarrow \mathsf{Samp}_0(q)$, $\{\mathbf{u}^{(i)}\}_{i\in[t]} \leftarrow \mathsf{Samp}_1(q, \mathsf{aux})$, $\gamma^{(i)} \leftarrow \mathbb{Z}_q$, and $\mathbf{v}^{(i)} \leftarrow \mathbb{Z}_q^m$ for $i \in [Q_{\mathsf{ct}}]$.

To complete the proof, we construct $\mathcal{B}$ that distinguishes the above distributions given the adversary $\mathcal{A}$. At the beginning, given the input, $\mathcal{B}$ first parses $\mathsf{aux} \to (\mathsf{BGG}^+.\mathsf{mpk}, \{\mathsf{BGG}^+.\mathsf{sk}^{(i)}\}_{i\in[Q_{\mathsf{kq}}]})$. $\mathcal{B}$ then honestly samples $\{\mathsf{BIPFE}.\mathsf{msk}_i\}_{i\in\{1,2\}}$ by itself.

Secret Key Queries: It can answer the secret key queries by $\mathcal{A}$ by $\{\mathsf{BGG}^+.\mathsf{sk}^{(i)}\}_{i\in[Q_{\mathsf{kq}}]}$.

Ciphertext Queries for Slot 1: For answering the ciphertext queries, the simulation for slot 1 is straightforward. Namely, $\mathcal{B}$ samples $\mathbf{s}^{(i)}$, $e_0^{(i)}$, $\mathbf{e}^{(i)}$, $\mathbf{e}_L^{(i)}$, and $\mathbf{e}_R^{(i)}$ for $i \neq j$ and generates the ciphertext $\mathsf{ct}_1^{(i)}$ for all $i \in [Q_{\mathsf{ct}}]$ as specified in $\mathbf{Game}_{j-1}$ using $\mathsf{BGG}^+.\mathsf{mpk}$. Note that $\mathbf{s}^{(j)}$, $e_0^{(j)}$, $\mathbf{e}^{(j)}$, $\mathbf{e}_L^{(j)}$, and $\mathbf{e}_R^{(j)}$ are not necessary for computing $\mathsf{ct}_1^{(j)}$ and are undefined at this point.

Ciphertext Queries for Slot 2: $\mathcal{B}$ also has to answer ciphertext queries for slot 2. To answer the i-th ciphertext query for slot 2, $\mathcal{B}$ first sets terms $\mathbf{d}_{1,2}^{(i)}$ and $\mathbf{d}_{2,3}^{(i)}$ as

$$\left[\mathbf{d}_{1,2}^{(i)}, \mathbf{d}_{2,3}^{(i)}\right]_2 := \left[\gamma^{(i)}, \mathbf{v}^{(i)}\right]_2,$$

where $\mathbf{v}^{(i)}$ is either random or $\mathbf{v}^{(i)} = \gamma^{(i)}\mathbf{u}^{(i)}$. The computation of $[\mathbf{D}^{(i)}]_2$ for the terms other than the above terms is straightforward using $[\gamma^{(i)}]_2$ by implicitly setting $t^{(i)} := \gamma^{(i)}$. It then computes the ciphertext $\mathsf{ct}_2^{(i)}$ as in the honest encryption algorithm.

[9] Thus, matrix $\mathbf{V}$ is defined as Eq.(5.2), where vectors $\{\mathbf{u}_{(i)}\}_i$ in the matrix are the vectors output by the Samp_1 algorithm above.

It can be seen that $\mathcal{B}$ simulates $\mathbf{Game}_{j-1,4}$ if $\mathbf{v}^{(i)}$ is random and $\mathbf{Game}_{j-1,3}$ if $\mathbf{v}^{(i)} = \gamma^{(i)}\mathbf{u}^{(i)}$, where $\mathcal{B}$ implicitly sets

$$\mathbf{s}^{(j)} := \mathbf{z}, \quad e_0^{(j)} := e_0, \quad \mathbf{e}^{(j)} := \mathbf{e}, \quad \left(\mathbf{e}_L^{(j)}, \mathbf{e}_L^{(j)}\right) := (\mathbf{e}_1, \ldots \mathbf{e}_{2\ell}) .$$

Therefore, $\mathcal{B}$ distinguishes the distributions if $\mathcal{A}$ distinguishes the games.

□

Lemma 5.9 ($\mathbf{Game}_{j-1,2} \approx_c \mathbf{Game}_{j-1,3}$)**.** *We have* $|\Pr[\mathsf{E}_{j-1,2}] - \Pr[\mathsf{E}_{j-1,3}]| \leq \mathrm{negl}(\lambda)$.

Proof. This follows by considering the same sequence of games that is used for showing the indistinguishability of $\mathbf{Game}_{j-1,0}$ and $\mathbf{Game}_{j-1,2}$, but in the reverse order and with the difference that $b_0^{(j)}$ is replaced by $b_1^{(j)}$. The indistinguishability between the games follows from the security of BIPFE, LWE, and the bilinear KOALA assumption for the same sampler. □

□

6 Compiling k-ABE to k-PE via Lockable Obfuscation

In this section we describe our compiler to lift k-input ABE to k-input PE. Namely, we construct k-input predicate encryption using k-input ABE and lockable obfuscation. The conversion preserves Ada-IND security. The extension of the conversion that preserves strong security is provided in Section 7.

6.1 Construction

Our construction uses the following building blocks:

1. A secret key encryption scheme $\mathsf{SKE} = (\mathsf{SKE.Setup}, \mathsf{SKE.Enc}, \mathsf{SKE.Dec})$.
2. A Lockable Obfuscator $\mathsf{LO} = (\mathsf{LO.Obf}, \mathsf{LO.Eval})$ with lock space $\mathcal{L} = \{0,1\}^m$ and input space $\mathcal{X} = \{0,1\}^n$.
3. A k-input ABE scheme $\mathsf{kABE} = (\mathsf{kABE.Setup}, \mathsf{kABE.KeyGen}, \mathsf{kABE.Enc}_1, \ldots, \mathsf{kABE.Enc}_k, \mathsf{kABE.Dec})$ in which the message bit is associated with the last slot, $\mathsf{kABE.Enc}_k$. We require $k = O(1)$.

   In the construction below, we require the message space of the SKE scheme to be the same as the lock space $\mathcal{L}$ of the lockable obfuscator scheme LO and the message space of kABE to be the same as the key space of SKE.

We now describe the construction of k-input predicate encryption scheme. Our k-input PE construction has the same attribute space and the function class as the underlying k-input ABE, when we consider the function class of NC_1 circuits or polynomial-size circuits.

$\mathsf{Setup}(1^\lambda)$: On input the security parameter 1^λ, the Setup algorithm does the following:

1. Run $(\mathsf{kABE.msk}, \mathsf{kABE.pp}) \leftarrow \mathsf{kABE.Setup}(1^\lambda)$.
2. Run $\mathsf{SKE.Setup}(1^\lambda)$ k times and obtain secret keys $K_1, K_2, \ldots, K_k$.
3. Output $\mathsf{msk} = (\mathsf{kABE.msk}, K_1, \ldots, K_k)$ and $\mathsf{pp} = \mathsf{kABE.pp}$.

$\mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, F)$: On input the public parameters pp, the master secret key $\mathsf{msk} = (\mathsf{kABE.msk}, K_1, \ldots, K_k)$ and a circuit F, the KeyGen algorithm does the following:

1. Run $\mathsf{kABE.sk}_F \leftarrow \mathsf{kABE.KeyGen}(\mathsf{pp}, \mathsf{kABE.msk}, F)$.
2. Output $\mathsf{sk}_F = \mathsf{kABE.sk}_F$.

$\mathsf{Enc}_1(\mathsf{pp}, \mathsf{msk}, \mathbf{x}_1, m)$: On input the public parameters pp, master secret key $\mathsf{msk} = (\mathsf{kABE.msk}, K_1, \ldots, K_k)$, attribute $\mathbf{x}_1$ for position 1 and message m, the encryption algorithm does the following:

1. Sample $\gamma_1 \leftarrow \mathcal{L}$ and let $\mathsf{ct}_1^* = \mathsf{SKE.Enc}(K_1, \gamma_1)$
2. Compute $\mathsf{ct}_1 = \mathsf{kABE.Enc}_1(\mathsf{pp}, \mathsf{kABE.msk}, \mathbf{x}_1)$.
3. Define a function $f_1[\mathsf{ct}_1, \mathsf{ct}_1^*]$ as in Figure 2.
4. Output $\mathsf{ct}_1' = \mathsf{LO.Obf}(1^\lambda, f_1[\mathsf{ct}_1, \mathsf{ct}_1^*], m, \gamma_1)$.

$\mathsf{Enc}_i(\mathsf{pp}, \mathsf{msk}, \mathbf{x}_i)$ for $2 \le i \le k$: On input the public parameters pp, master secret key $\mathsf{msk} = (\mathsf{kABE.msk}, K_1, \ldots, K_k)$, attribute $\mathbf{x}_i$ for position i, the encryption algorithm does the following:

1. Sample a random value $\gamma_i \leftarrow \mathcal{L}$ and let $\mathsf{ct}_i^* = \mathsf{SKE.Enc}(K_i, \gamma_i)$.
2. Compute $\mathsf{ct}_i = \begin{cases} \mathsf{kABE.Enc}_i(\mathsf{pp}, \mathsf{kABE.msk}, \mathbf{x}_i) & \text{for } 2 \le i < k \\ \mathsf{kABE.Enc}_k(\mathsf{pp}, \mathsf{kABE.msk}, \mathbf{x}_k, K_k) & \text{for } i = k \end{cases}$.
3. Define a function $f_i[\mathsf{ct}_i, \mathsf{ct}_i^*]$ as in Figure 2.
4. Output $\mathsf{ct}_i' = \mathsf{LO.Obf}(1^\lambda, f_i[\mathsf{ct}_i, \mathsf{ct}_i^*], K_{i-1}, \gamma_i)$.

Circuit $f_i[\mathsf{ct}_i, \mathsf{ct}_i^*]$ for $1 \le i \le k$

1. Parse input as $(\mathsf{ct}_1, \ldots, \mathsf{ct}_{i-1}, \tilde{G}_{i+1}, \ldots, \tilde{G}_k, \mathsf{sk}_F)$ where ct_j is regarded as a slot j ciphertext of kABE, $\tilde{G}_j$ is regarded as an obfuscated circuit of LO and sk_F is regarded as a kABE secret key.
2. Compute $K_i' = \begin{cases} \mathsf{LO.Eval}(\tilde{G}_{i+1}, (\mathsf{ct}_1, \ldots, \mathsf{ct}_i, \tilde{G}_{i+2}, \ldots, \tilde{G}_k, \mathsf{sk}_F)) & \text{for } 1 \le i < k \\ \mathsf{kABE.Dec}(\mathsf{pp}, \mathsf{sk}_F, \mathsf{ct}_1, \ldots, \mathsf{ct}_k) & \text{for } i = k \end{cases}$
3. Outputs $\gamma_i' \leftarrow \mathsf{SKE.Dec}(\mathsf{K}_i', \mathsf{ct}_i^*)$.

Figure 2: Circuit Obfuscated by Slot i Encryption for $1 \le i \le k$

$\mathsf{Dec}(\mathsf{sk}_F, \mathsf{ct}_1', \ldots, \mathsf{ct}_k')$: On input the secret key sk_F for function F, and kPE ciphertexts $\mathsf{ct}_1', \ldots, \mathsf{ct}_k'$, do the following:

1. Parse ct_1' as an LO obfuscation.
2. Compute and output $\mathsf{LO.Eval}(\mathsf{ct}_1', (\mathsf{ct}_2', \ldots, \mathsf{ct}_k', \mathsf{sk}_F))$.

Correctness. To establish correctness, we first prove the following statement:

Claim 6.1. *For* $\mathbf{x}_1, \ldots, \mathbf{x}_k$ *such that* $F(\mathbf{x}_1, \ldots, \mathbf{x}_k) = 0$*, and* $\mathsf{ct}_i, \mathsf{ct}_i^*, \mathsf{ct}_i'$*, for* $1 \leq i \leq k$*, computed as per the scheme,*

$$\textit{For } 2 \leq i \leq k,\ \mathsf{LO.Eval}(\mathsf{ct}_i', (\mathsf{ct}_1, \ldots, \mathsf{ct}_{i-1}, \mathsf{ct}_{i+1}', \ldots, \mathsf{ct}_k', \mathsf{sk}_F)) = K_{i-1}.$$

Proof. We can prove this by induction.
Base case: For $i = k$, we show that

$$\mathsf{LO.Eval}(\mathsf{ct}_k', (\mathsf{ct}_1, \ldots, \mathsf{ct}_{k-1}, \mathsf{sk}_F)) = K_{k-1}$$

Proof: Since, $\mathsf{ct}_k' = \mathsf{LO.Obf}(1^\lambda, f_k[\mathsf{ct}_k, \mathsf{ct}_k^*], K_{k-1}, \gamma_k)$, from the functionality of LO, $f_k[\mathsf{ct}_k, \mathsf{ct}_k^*]$ is evaluated on input $(\mathsf{ct}_1, \ldots, \mathsf{ct}_{k-1}, \mathsf{sk}_F)$ in the following steps:

1. $\mathsf{kABE.Dec}(\mathsf{pp}, \mathsf{sk}_F, \mathsf{ct}_1, \ldots, \mathsf{ct}_k) = \mathsf{K}_k$, from the correctness of $\mathsf{kABE.Dec}$
2. Output $\mathsf{SKE.Dec}(K_k, \mathsf{ct}_k^*) = \gamma_k$, from the correctness of $\mathsf{SKE.Dec}$

Since, the output of function $f_k[\mathsf{ct}_k, \mathsf{ct}_k^*]$ matches the lock value in ct_k', $\mathsf{LO.Eval}(\mathsf{ct}_k', (\mathsf{ct}_1, \ldots, \mathsf{ct}_{k-1}, \mathsf{sk}_F)) = K_{k-1}$, from the correctness of LO.

Inductive Step: We show that for $2 \leq i \leq k-1$, if

$$\mathsf{LO.Eval}(\mathsf{ct}_{i+1}', (\mathsf{ct}_1, \ldots, \mathsf{ct}_i, \mathsf{ct}_{i+2}', \ldots, \mathsf{ct}_k', \mathsf{sk}_F)) = K_i,$$

then

$$\mathsf{LO.Eval}(\mathsf{ct}_i', (\mathsf{ct}_1, \ldots, \mathsf{ct}_{i-1}, \mathsf{ct}_{i+1}', \ldots, \mathsf{ct}_k', \mathsf{sk}_F)) = K_{i-1}.$$

Proof: Recall that $\mathsf{ct}_i' = \mathsf{LO.Obf}(1^\lambda, f_i[\mathsf{ct}_i, \mathsf{ct}_i^*], K_{i-1}, \gamma_i)$. By LO's functionality, $\mathsf{LO.Eval}(\mathsf{ct}_i', (\mathsf{ct}_1, \ldots, \mathsf{ct}_{i-1}, \mathsf{ct}_{i+1}', \ldots, \mathsf{ct}_k', \mathsf{sk}_F))$ first evaluates $f_i[\mathsf{ct}_i, \mathsf{ct}_i^*]$ on input $(\mathsf{ct}_1, \ldots, \mathsf{ct}_{i-1}, \mathsf{ct}_{i+1}', \ldots, \mathsf{ct}_k', \mathsf{sk}_F)$ in the following two steps:

1. $\mathsf{LO.Eval}(\mathsf{ct}_{i+1}', (\mathsf{ct}_1, \ldots, \mathsf{ct}_i, \mathsf{ct}_{i+2}', \ldots, \mathsf{ct}_k', \mathsf{sk}_F)) = K_i$, by the induction hypothesis.
2. Output $\mathsf{SKE.Dec}(K_i, \mathsf{ct}_i^*) = \gamma_i$, from the correctness of SKE.

Since, the function output matches with the lock value,
$\mathsf{LO.Eval}(\mathsf{ct}_i', (\mathsf{ct}_1, \ldots, \mathsf{ct}_{i-1}, \mathsf{ct}_{i+1}', \ldots, \mathsf{ct}_k', \mathsf{sk}_F)) = K_{i-1}$ from the correctness of $\mathsf{LO.Eval}$. □

Finally, we observe that the kPE decryption outputs $\mathsf{LO.Eval}(\mathsf{ct}_1', (\mathsf{ct}_2', \ldots, \mathsf{ct}_k', \mathsf{sk}_F))$, where $\mathsf{ct}_1' = \mathsf{LO.Obf}(1^\lambda, f_1[\mathsf{ct}_1, \mathsf{ct}_1^*], m, \gamma_1)$. Hence from the functionality of LO, firstly the function $f_1[\mathsf{ct}_1, \mathsf{ct}_1^*]$ is evaluated on input $(\mathsf{ct}_2', \ldots, \mathsf{ct}_k', \mathsf{sk}_F)$ in the following steps:

1. Compute $\mathsf{LO.Eval}(\mathsf{ct}_2', (\mathsf{ct}_1, \mathsf{ct}_3', \ldots, \mathsf{ct}_k', \mathsf{sk}_F)) = K_1$.
2. Output $\mathsf{SKE.Dec}(K_1, \mathsf{ct}_1^*) = \gamma_1$ from the correctness of SKE.

Since, $f_1[\mathsf{ct}_1, \mathsf{ct}_1^*]$ evaluates to γ_1, which is the lock value in ct_1', from the correctness of $\mathsf{LO.Eval}$, we get $\mathsf{LO.Eval}(\mathsf{ct}_1', (\mathsf{ct}_2', \ldots, \mathsf{ct}_k', \mathsf{sk}_F)) = m$ as desired.

6.2 Security

We prove that the above construction satisfies Ada-IND security of Definition 3.2 via the following theorem.

Theorem 6.2. *Assume* LO *is a secure lockable obfuscation scheme as per Definition 2.9, that* kABE *is a secure k input* ABE *scheme as per Definition 3.1 and* SKE *is a secure secret key encryption scheme. Then, the* kPE *construction presented above is secure as per Definition 3.2.*

Proof. The proof proceeds via a sequence of following games between the challenger and a PPT adversary $\mathcal{A}$.

Game$_0$: This is the real world.

Game$_1$: In this world, the SKE key K_k encrypted in the kABE ciphertext ct_k is replaced with $\mathbf{0}$.

For $a = 2$ to $k+1$ define:

Game$_{a.0}$: In this world,

1. For $j \in [1, k-(a-1)]$, ct'_j is computed as in the real world.
2. For $j = k-(a-2)$,
   (a) $\mathsf{ct}_j = \begin{cases} \mathsf{kABE.Enc}_j(\mathsf{pp}, \mathsf{kABE.msk}, \mathbf{x}^i_{j,b}) & \text{if } j < k \text{ (i.e. } a > 2) \\ \mathsf{kABE.Enc}_j(\mathsf{pp}, \mathsf{kABE.msk}, \mathbf{x}^i_{j,b}, \mathbf{0}) & \text{if } j = k \text{ (i.e. } a = 2) \end{cases}$
   (b) $\mathsf{ct}^*_j = \mathsf{SKE.Enc}(K_j, \mathbf{0})$
   (c) $\mathsf{ct}'_j = \begin{cases} \mathsf{LO.Obf}(1^\lambda, f_j[\mathsf{ct}_j, \mathsf{ct}^*_j], K_{j-1}, \gamma_j) & \text{if } j > 1 \text{ (i.e. } a < k+1) \\ \mathsf{LO.Obf}(1^\lambda, f_j[\mathsf{ct}_j, \mathsf{ct}^*_j], m^i_b, \gamma_j) & \text{if } j = 1 \text{ (i.e. } a = k+1) \end{cases}$
3. For $j \in [k-(a-3), k]$, ct'_j is generated using LO simulator. In more detail,
   (a) $\mathsf{ct}_j = \begin{cases} \mathsf{kABE.Enc}_j(\mathsf{pp}, \mathsf{kABE.msk}, \mathbf{x}^i_{j,b}) & \text{if } j < k \\ \mathsf{kABE.Enc}_j(\mathsf{pp}, \mathsf{kABE.msk}, \mathbf{x}^i_{j,b}, \mathbf{0}) & \text{if } j = k \end{cases}$
   (b) $\mathsf{ct}^*_j = \mathsf{SKE.Enc}(K_j, \mathbf{0})$
   (c) $\mathsf{ct}'_j = \mathsf{LO.Sim}(1^\lambda, 1^{|f_j[\mathsf{ct}_j, \mathsf{ct}^*_j]|}, 1^{|K_{j-1}|})$

Game$_{a.1}$: This is same as **Game**$_{a.0}$, except the following change: In this world, $\mathsf{ct}'_{k-(a-2)}$ is generated using LO simulator.

Indistinguishability of Hybrids. We now show that the consecutive games are indistinguishable. We let E_x denote the event that the adversary $\mathcal{A}$ outputs correct guess for the challenge bit b at the end of **Game**$_x$.

Claim 6.3. *Assume that* kABE *satisfies* Ada-IND *security (Definition 3.1). Then,* **Game**$_0$ *and* **Game**$_1$ *are computationally indistinguishable. That is,*

$$|\Pr[\mathsf{E}_0] - \Pr[\mathsf{E}_1]| \leq \mathrm{negl}(\lambda).$$

Proof. We show that if $\mathcal{A}$ can distinguish between **Game**$_0$ and **Game**$_1$ with non-negligible probability then there exists an adversary $\mathcal{B}$ who can break Ada-IND security of kABE using $\mathcal{A}$. The reduction is as follows:

1. The kABE challenger samples $(\mathsf{kABE.msk}, \mathsf{kABE.pp}) \leftarrow \mathsf{kABE.Setup}(1^\lambda)$, a bit $b' \leftarrow \{0,1\}$ and sends $\mathsf{kABE.pp}$ to $\mathcal{B}$.

2. Upon receiving the public parameters $\mathsf{kABE.pp}$ from the kABE challenger, $\mathcal{B}$ sets $\mathsf{pp} = \mathsf{kABE.pp}$, samples k SKE keys $K_1, \ldots, K_k$ using $\mathsf{SKE.KeyGen}$ and invokes $\mathcal{A}$ with pp as public parameters and chooses a bit $b \leftarrow \{0,1\}$. $\mathcal{B}$ implicitly sets $\mathsf{msk} = (\mathsf{kABE.msk}, K_1, \ldots, K_k)$.

3. $\mathcal{B}$ then responds to key queries and ciphertext queries from $\mathcal{A}$ as follows:

   Key Queries: $\mathcal{B}$ forwards each key query for a function F to kABE challenger and obtains a secret key $\mathsf{kABE.sk}_F$. $\mathcal{B}$ returns $\mathsf{sk}_F = \mathsf{kABE.sk}_F$ to $\mathcal{A}$.

   Ciphertext Queries: Each ciphertext query from $\mathcal{A}$ is of the form

   $$\begin{cases} (\mathbf{x}^i_{1,0}, \mathbf{x}^i_{1,1}), (m^i_0, m^i_1) & \text{(for slot 1), or} \\ (\mathbf{x}^i_{j,0}, \mathbf{x}^i_{j,1}) & \text{(for slot } 1 < j \leq k) \end{cases}$$

   On receiving a ciphertext query, $\mathcal{B}$ does the following:

   (a) If the query is for slot $1 \leq j \leq k-1$,
   - Samples a random value $\gamma_j \leftarrow \mathcal{L}$ and computes $\mathsf{ct}^*_j = \mathsf{SKE.Enc}(K_j, \gamma_j)$.
   - Sends $\mathbf{x}^i_{j,b}$, as a ciphertext query to the kABE challenger.
   - The kABE challenger returns a ciphertext $\mathsf{ct}_j = \mathsf{kABE.Enc}_j(\mathsf{kABE.pp}, \mathsf{kABE.msk}, \mathbf{x}^i_{j,b})$ for slot j.
   - $\mathcal{B}$ defines the function $f_j[\mathsf{ct}_j, \mathsf{ct}^*_j]$ and returns
     $$\mathsf{ct}'_j = \begin{cases} \mathsf{LO.Obf}(1^\lambda, f_j[\mathsf{ct}_j, \mathsf{ct}^*_j], m^i_b, \gamma_j) & \text{if } j = 1 \\ \mathsf{LO.Obf}(1^\lambda, f_j[\mathsf{ct}_j, \mathsf{ct}^*_j], K_{j-1}, \gamma_j) & \text{otherwise} \end{cases}$$

   (b) If the query is for slot k
   - Samples $\gamma_k \leftarrow \mathcal{L}$ and computes $\mathsf{ct}^*_k = \mathsf{SKE.Enc}(K_k, \gamma_k)$.
   - Sends $(\mathbf{x}^i_{k,b}, (\mu^i_0 = K_k, \mu^i_1 = \mathbf{0}))$ as ciphertext query to the kABE challenger.
   - The kABE challenger computes and returns a ciphertext ct_k for slot k, computed as $\mathsf{ct}_k = \mathsf{kABE.Enc}_k(\mathsf{kABE.pp}, \mathsf{kABE.msk}, \mathbf{x}^i_{k,b}, \mu^i_{b'})$.
   - $\mathcal{B}$ defines function $f_k[\mathsf{ct}_k, \mathsf{ct}^*_k]$ and computes and returns
     $$\mathsf{ct}'_k \quad = \quad \mathsf{LO.Obf}(1^\lambda, f_k[\mathsf{ct}_k, \mathsf{ct}^*_k], K_{k-1}, \gamma_k).$$

4. In the end, $\mathcal{A}$ outputs its guess bit $\hat{b}$. If $b = \hat{b}$, then $\mathcal{B}$ returns $b'' = 1$, else $b'' = 0$ to the kABE challenger.

We can observe that if the bit b' chosen by kABE challenger is 0, then $\mathcal{B}$ simulated $\mathbf{Game}_0$, else $\mathbf{Game}_1$ with $\mathcal{A}$. This gives us the advantage of $\mathcal{B}$, against kABE challenger, as $|\Pr(b'' = 1 | b' = 0) - \Pr(b'' = 1 | b' = 1)| = |\Pr[\mathsf{E}_0] - \Pr[\mathsf{E}_1]|$. Hence, assuming $\mathsf{Ada\text{-}IND}$ security of kABE, we get

$$|\Pr[\mathsf{E}_0] - \Pr[\mathsf{E}_1]| \leq \mathrm{negl}(\lambda).$$

Admissibility of $\mathcal{B}$: Observe that the key queries made by $\mathcal{B}$ to the kABE challenger are the same key queries as made by $\mathcal{A}$ to $\mathcal{B}$. Also the attribute in each ciphertext query by $\mathcal{B}$ to kABE challenger is taken from the corresponding ciphertext query by $\mathcal{A}$. Hence, the admissibility of $\mathcal{A}$ implies admissibility of $\mathcal{B}$. □

Claim 6.4. *Assume that* SKE *is a CPA secure encryption scheme. Then* $\mathbf{Game}_1$ *and* $\mathbf{Game}_{2.0}$ *are computationally indistinguishable. That is,*

$$|\Pr[\mathsf{E}_1] - \Pr[\mathsf{E}_{2.0}]| \leq \mathrm{negl}(\lambda).$$

We show that if $\mathcal{A}$ can distinguish between $\mathbf{Game}_1$ and $\mathbf{Game}_{2.0}$ with non-negligible probability, then there exists an adversary $\mathcal{B}$ who can break CPA security of SKE using $\mathcal{A}$. The reduction is as follows:

1. The SKE challenger samples $K \leftarrow \mathsf{SKE.Setup}(1^\lambda)$ and a bit $b' \leftarrow \{0,1\}$ and invokes $\mathcal{B}$.
2. Upon being challenged by SKE challenger, $\mathcal{B}$ does the following:
   (a) Samples $(\mathsf{kABE.pp}, \mathsf{kABE.msk}) \leftarrow \mathsf{kABE.Setup}(1^\lambda)$ and SKE keys $K_1, \ldots, K_{k-1}$. Sets $\mathsf{pp} = \mathsf{kABE.pp}$, $\mathsf{msk} = (\mathsf{kABE.msk}, K_1, \ldots, K_{k-1}, K_k)$, where $\mathcal{B}$ implicitly sets K_k to be the secret key K sampled by the SKE challenger.
   (b) Samples a bit b and invokes $\mathcal{A}$ with pp.
   (c) For each key query for any function F from $\mathcal{A}$, $\mathcal{B}$ returns $\mathsf{sk}_F \leftarrow \mathsf{kABE.KeyGen}(\mathsf{pp}, \mathsf{kABE.msk}, F)$.
   (d) To answer each ciphertext query which is of the form

$$\begin{cases} (\mathbf{x}^i_{1,0}, \mathbf{x}^i_{1,1}), (m^i_0, m^i_1) & \text{(for slot 1), or} \\ (\mathbf{x}^i_{j,0}, \mathbf{x}^i_{j,1}) & \text{(for slot } 1 < j \leq k\text{),} \end{cases}$$

   $\mathcal{B}$ does the following:
      i. If the query is for slot $j < k$, $\mathcal{B}$ samples γ_j and computes ct_j and ct^*_j on its own as $\mathsf{ct}_j = \mathsf{kABE.Enc}_j(\mathsf{pp}, \mathsf{kABE.msk}, \mathbf{x}^i_{j,b})$, $\mathsf{ct}^*_j = \mathsf{SKE.Enc}(K_j, \gamma_j)$. Defines $f_j[\mathsf{ct}_j, \mathsf{ct}^*_j]$ and returns

$$\mathsf{ct}'_j = \begin{cases} \mathsf{LO.Obf}(1^\lambda, f_1[\mathsf{ct}_1, \mathsf{ct}^*_1], m^i_b, \gamma_1) & \text{if } j = 1 \\ \mathsf{LO.Obf}(1^\lambda, f_j[\mathsf{ct}_j, \mathsf{ct}^*_j], K_{j-1}, \gamma_j), & \text{otherwise} \end{cases}$$

      ii. If the query is for slot k
         A. $\mathcal{B}$ computes $\mathsf{ct}_k = \mathsf{kABE.Enc}_k(\mathsf{pp}, \mathsf{kABE.msk}, \mathbf{x}^i_{k,b}, \mathbf{0})$
         B. Samples $\gamma_k \leftarrow \mathcal{L}$ and sends $\mu^i_0 = \gamma_k$ and $\mu^i_1 = \mathbf{0}$ as challenge messages to the SKE challenger.
         C. SKE challenger returns $\mathsf{ct}^*_k = \mathsf{SKE.Enc}(K_k, \mu^i_{b'})$.
         D. $\mathcal{B}$ defines function $f_k[\mathsf{ct}_k, \mathsf{ct}^*_k]$ and returns

$$\mathsf{ct}'_k = \mathsf{LO.Obf}(1^\lambda, f_k[\mathsf{ct}_k, \mathsf{ct}^*_k], K_{k-1}, \gamma_k)$$

         to $\mathcal{A}$.
   (e) In the end, $\mathcal{A}$ outputs a bit $\hat{b}$. If $\hat{b} = b$, then $\mathcal{B}$ returns $b'' = 1$, else $b'' = 0$ to the SKE challenger.

We can observe that if $b' = 0$ then $\mathcal{B}$ simulated $\mathbf{Game}_1$, else $\mathbf{Game}_{2.0}$. Hence, if $\mathcal{A}$ distinguishes between $\mathbf{Game}_1$ and $\mathbf{Game}_{2.0}$, with non negligible probability then $\mathcal{B}$ also wins against the SKE challenger. Assuming CPA security of SKE, we get

$$|\Pr[\mathsf{E}_1] - \Pr[\mathsf{E}_{2.0}]| \leq \mathrm{negl}(\lambda).$$

□

Claim 6.5. *Assume that* LO *is a secure lockable obfuscation scheme (Definition 2.9). Then for* $2 \leq a \leq k+1$, $\mathbf{Game}_{a.0}$ *and* $\mathbf{Game}_{a.1}$ *are computationally indistinguishable. That is,*

$$|\Pr[\mathsf{E}_{a.0}] - \Pr[\mathsf{E}_{a.1}]| \leq \mathrm{negl}(\lambda).$$

Proof. Recall that in both the hybrids,

- For $j \in [1, k-(a-1)]$, ct'_j is computed as in the real world.
- For $j \in [k-(a-3), k]$, ct'_j is generated using LO simulator.
- For $j = k-(a-2)$, ct_j and ct^*_j are computed as:

$$\mathsf{ct}_j = \begin{cases} \mathsf{kABE.Enc}_j(\mathsf{pp}, \mathsf{kABE.msk}, \mathbf{x}^i_{j,b}), & \text{if } j < k, (\text{i.e. } a > 2) \\ \mathsf{kABE.Enc}_j(\mathsf{pp}, \mathsf{kABE.msk}, \mathbf{x}^i_{j,b}, \mathbf{0}), & \text{if } j = k, (\text{i.e. } a = 2), \end{cases}$$

$$\mathsf{ct}^*_j = \mathsf{SKE.Enc}(K_j, \mathbf{0})$$

The only difference between the two hybrids is in the generation of $\mathsf{ct}'_{k-(a-2)}$ as following. Let $j = k-(a-2)$.

In $\mathbf{Game}_{a.0}$,

$$\mathsf{ct}'_j = \begin{cases} \mathsf{LO.Obf}(1^\lambda, f_j[\mathsf{ct}_j, \mathsf{ct}^*_j], m^i_b, \gamma_j), & \text{if } j = 1, (\text{i.e. } a = k+1) \\ \mathsf{LO.Obf}(1^\lambda, f_j[\mathsf{ct}_j, \mathsf{ct}^*_j], K_{j-1}, \gamma_j), & \text{otherwise} \end{cases}$$

In $\mathbf{Game}_{a.1}$,

$$\mathsf{ct}'_j = \begin{cases} \mathsf{LO.Sim}(1^\lambda, 1^{|f_j[\mathsf{ct}_j, \mathsf{ct}^*_j]|}, 1^{|m^i_b|}), & \text{if } j = 1, (\text{i.e. } a = k+1) \\ \mathsf{LO.Sim}(1^\lambda, 1^{|f_j[\mathsf{ct}_j, \mathsf{ct}^*_j]|}, 1^{|K_{j-1}|}), & \text{otherwise} \end{cases}$$

We show that if $\mathcal{A}$ can distinguish between $\mathbf{Game}_{a.0}$ and $\mathbf{Game}_{a.1}$ then there exists an adversary $\mathcal{B}$ who can distinguish between LO obfuscated programs and simulated programs, thus breaking the security of LO. The reduction is as follows:

1. The LO challenger samples $b' \leftarrow \{0,1\}$ and starts the game with $\mathcal{B}$. Upon being challenged by the LO challenger, $\mathcal{B}$ does the following:
   (a) Samples public parameters and master secret key for kPE as $(\mathsf{pp}, \mathsf{msk} = (\mathsf{kABE.msk}, K_1, \ldots, K_k)) \leftarrow \mathsf{Setup}(1^\lambda)$ and invokes $\mathcal{A}$ with pp. $\mathcal{B}$ also samples a bit b.
   (b) $\mathcal{A}$ issues polynomially many key queries and ciphertext queries, to which $\mathcal{B}$ responds as following.
      i. For each key query for a function F from $\mathcal{A}$, $\mathcal{B}$ returns $\mathsf{sk}_F \leftarrow \mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, F)$ to $\mathcal{A}$.

ii. To answer each ciphertext query which is of the form

$$\begin{cases} (\mathbf{x}^i_{1,0}, \mathbf{x}^i_{1,1}), (m^i_0, m^i_1) & \text{(for slot 1), or} \\ (\mathbf{x}^i_{j,0}, \mathbf{x}^i_{j,1}) & \text{(for slot } j > 1\text{),} \end{cases}$$

$\mathcal{B}$ does the following:

A. If the query is for slot $j \in [1, k-(a-1)]$,
- Samples $\gamma_j \leftarrow \mathcal{L}$.
- Computes ct_j and ct^*_j using msk and $\mathbf{x}^i_{j,b}$ as attribute.
- Defines function $f_j[\mathsf{ct}_j, \mathsf{ct}^*_j]$ and returns

$$\mathsf{ct}'_j = \begin{cases} \mathsf{LO.Obf}(1^\lambda, f_1[\mathsf{ct}_1, \mathsf{ct}^*_1], m^i_b, \gamma_1), & \text{if } j = 1 \\ \mathsf{LO.Obf}(1^\lambda, f_j[\mathsf{ct}_j, \mathsf{ct}^*_j], K_{j-1}, \gamma_j), & \text{otherwise} \end{cases}$$

B. If the query is for slot $j \in [k-(a-3), k]$
- Computes

$$\mathsf{ct}_j = \begin{cases} \mathsf{kABE.Enc}_j(\mathsf{pp}, \mathsf{kABE.msk}, \mathbf{x}^i_{j,b}) & \text{if } j < k \\ \mathsf{kABE.Enc}_j(\mathsf{pp}, \mathsf{kABE.msk}, \mathbf{x}^i_{j,b}, \mathbf{0}) & \text{if } j = k \end{cases},$$

  $\mathsf{ct}^*_j = \mathsf{SKE.Enc}(K_j, \mathbf{0})$ and defines $f_j[\mathsf{ct}_j, \mathsf{ct}^*_j]$.
- Returns $\mathsf{ct}'_j = \mathsf{LO.Sim}(1^\lambda, 1^{|f_j[\mathsf{ct}_j, \mathsf{ct}^*_j]|}, 1^{|K_{j-1}|})$.

C. If the query is for slot $j = k-(a-2)$
- Computes

$$\mathsf{ct}_j = \begin{cases} \mathsf{kABE.Enc}_j(\mathsf{pp}, \mathsf{kABE.msk}, \mathbf{x}^i_{j,b}) & \text{if } j < k \\ \mathsf{kABE.Enc}_j(\mathsf{pp}, \mathsf{kABE.msk}, \mathbf{x}^i_{j,b}, \mathbf{0}) & \text{if } j = k \end{cases},$$

  $\mathsf{ct}^*_j = \mathsf{SKE.Enc}(K_j, \mathbf{0})$ and defines $f_j[\mathsf{ct}_j, \mathsf{ct}^*_j]$.
- If $j = 1$ (i.e. $a = k+1$), sends $f_j[\mathsf{ct}_j, \mathsf{ct}^*_j], m^i_b$, else sends $f_j[\mathsf{ct}_j, \mathsf{ct}^*_j], K_{j-1}$ to the LO challenger and receives either an LO obfuscated or a simulated program ct'_j from the LO challenger.
- Returns ct'_j to $\mathcal{A}$.

(c) In the end, $\mathcal{A}$ outputs a bit $\hat{b}$. If $\hat{b} = b$, then $\mathcal{B}$ returns $b'' = 1$, else $b'' = 0$, to LO challenger.

We can observe that if the LO challenger returned obfuscated programs, then $\mathcal{B}$ simulated $\mathbf{Game}_{a.0}$, else if LO challenger returned simulated programs, then $\mathcal{B}$ simulated $\mathbf{Game}_{a.1}$ with $\mathcal{A}$. Hence, if $\mathcal{A}$ distinguishes between the two games, then so does $\mathcal{B}$ between obfuscated and simulated programs. Assuming LO is secure, we get

$$|\Pr[\mathsf{E}_{a.0}] - \Pr[\mathsf{E}_{a.1}]| \leq \mathrm{negl}(\lambda).$$

□

Claim 6.6. *Assume that* SKE *is a CPA secure encryption scheme. Then for* $2 \leq a \leq k$, $\mathbf{Game}_{a.1}$ *and* $\mathbf{Game}_{a+1.0}$ *are computationally indistinguishable . That is,*

$$|\Pr[\mathsf{E}_{a.1}] - \Pr[\mathsf{E}_{a+1.0}]| \leq \mathrm{negl}(\lambda).$$

Proof. The only difference between the two hybrids is in the computation of $\mathsf{ct}^*_{k-(a-1)}$. In $\mathbf{Game}_{a.1}$, $\mathsf{ct}^*_{k-(a-1)} = \mathsf{SKE.Enc}(K_{k-(a-1)}, \gamma_{k-(a-1)})$, while in $\mathbf{Game}_{a+1.0}$, $\mathsf{ct}^*_{k-(a-1)} = \mathsf{SKE.Enc}(K_{k-(a-1)}, \mathbf{0})$. Hence the indistinguishability of the two hybrids follows from the CPA security of SKE. The reduction is similar to that in the proof of indistinguishability between $\mathbf{Game}_1$ and $\mathbf{Game}_{2.0}$. □

Claim 6.7. $\Pr[\mathsf{E}_{k+1.1}] - \frac{1}{2} = 0$

Proof. In $\mathbf{Game}_{k+1.1}$, all the LO, circuits returned as kPE ciphertexts, are simulated using LO simulator as $\mathsf{ct}'_1 = \mathsf{LO.Sim}(1^\lambda, 1^{|f_1[\mathsf{ct}_1, \mathsf{ct}^*_1]|}, 1^{|m^i_b|})$ and $\mathsf{ct}'_j = \mathsf{LO.Sim}(1^\lambda, 1^{|f_j[\mathsf{ct}_j, \mathsf{ct}^*_j]|}, 1^{|K_{j-1}|})$ for $2 \leq j \leq k$. Hence, they depend only on the lengths of functions $f_j[\mathsf{ct}_j, \mathsf{ct}^*_j]$, length of message and length of SKE keys. Length of $f_j[\mathsf{ct}_j, \mathsf{ct}^*_j]$ further depends only on the length of attributes and messages. Since, these lengths are fixed for the scheme, $\{\mathsf{ct}'_j\}_{j\in[k]}$ completely hide the bit b. Hence, $\mathcal{A}$ can do nothing better than a pure guess for bit b in $\mathbf{Game}_{k+1.1}$. □

Applications. The conversion above can be applied to all the multi-input ABE schemes in this paper. Here, we focus on the applications to the candidate two input ABE scheme from lattices in Sec. 9 and the candidate three input ABE scheme in Sec. 8. The other schemes will be discussed in Sec. 7 because they satisfy strong (very selective) security and thus we can apply the conversion in Sec. 7. A nice property of the PE scheme obtained from the two input ABE scheme in Sec. 9 is that it can handle any polynomial-size circuits. Besides, we can expect that it is post-quantum secure, because it does not use pairings and only uses lattice tools. By applying the conversion to the three input ABE scheme in Sec. 8, we can obtain a three-input PE scheme that can handle NC_1 circuits.

7 Two-Input PE with Stronger Security

In this section we describe our compiler to lift 2-input ABE to 2-input PE that preserves strong security. The conversion uses lockable obfuscation similarly to Sec. 6. Unlike the conversion in Sec. 6, we do not know how to extend it to general arity k and it is set to be $k = 2$ here. The construction uses the following building blocks:

1. Two instances of 2-input ABE scheme. In one instance the message is associated with encryption for position 2, while in the other instance, the message is associated with the encryption for position 1. We represent the two instances as $\mathsf{2ABE} = (\mathsf{2ABE.Setup}, \mathsf{2ABE.KeyGen}, \mathsf{2ABE.Enc}_1, \mathsf{2ABE.Enc}_2, \mathsf{2ABE.Dec})$ and $\mathsf{2ABE}' = (\mathsf{2ABE}'.\mathsf{Setup}, \mathsf{2ABE}'.\mathsf{KeyGen}, \mathsf{2ABE}'.\mathsf{Enc}_1, \mathsf{2ABE}'.\mathsf{Enc}_2, \mathsf{2ABE}'.\mathsf{Dec})$.
2. A Lockable Obfuscator $\mathsf{Obf} = (\mathsf{LO.Obf}, \mathsf{LO.Eval})$.

7.1 Construction

Our two-input PE construction has the same attribute space and the function class as the underlying two-input ABE, when we consider the function class of NC_1 circuits or polynomial-size circuits.

$\mathsf{Setup}(1^\lambda)$: On input 1^λ, the Setup algorithm does the following:

1. Run $(\mathsf{2ABE.msk}, \mathsf{2ABE.pp}) \leftarrow \mathsf{2ABE.Setup}(1^\lambda)$ and $(\mathsf{2ABE}'.\mathsf{msk}, \mathsf{2ABE}'.\mathsf{pp}) \leftarrow \mathsf{2ABE}'.\mathsf{Setup}(1^\lambda)$.
2. Output $\mathsf{msk} = (\mathsf{2ABE.msk}, \mathsf{2ABE}'.\mathsf{msk})$ and $\mathsf{pp} = (\mathsf{2ABE.pp}, \mathsf{2ABE}'.\mathsf{pp})$.

$\mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, F)$: On input the public parameters pp, the master secret key msk and a circuit F, the keygen algorithm does the following:

1. Parse msk as $(\mathsf{2ABE.msk}, \mathsf{2ABE}'.\mathsf{msk})$ and $\mathsf{pp} = (\mathsf{2ABE.pp}, \mathsf{2ABE}'.\mathsf{pp})$.

2. Run $\mathsf{2ABE.sk}_F \leftarrow \mathsf{2ABE.KeyGen}(\mathsf{2ABE.pp}, \mathsf{2ABE.msk}, F)$ and
$\mathsf{2ABE'.sk}_F \leftarrow \mathsf{2ABE'.KeyGen}(\mathsf{2ABE'.pp}, \mathsf{2ABE'.msk}, F)$.
3. Output $\mathsf{sk}_F = (\mathsf{2ABE.sk}_F, \mathsf{2ABE'.sk}_F)$.

$\mathsf{Enc}_1(\mathsf{pp}, \mathsf{msk}, \mathbf{x}_1, m)$: On input the public parameters, pp, master secret key msk, attribute $\mathbf{x}_1$ for position 1 and message m, the encryption algorithm does the following:

1. Parses msk as $(\mathsf{2ABE.msk}, \mathsf{2ABE'.msk})$ and pp as $(\mathsf{2ABE.pp}, \mathsf{2ABE'.pp})$.
2. Computes $\mathsf{ct}_1 = \mathsf{2ABE.Enc}_1(\mathsf{2ABE.pp}, \mathsf{2ABE.msk}, \mathbf{x}_1)$.
3. Sample $\alpha \leftarrow \mathcal{M}$ and compute $\mathsf{ct}'_1 = \mathsf{2ABE'.Enc}_1(\mathsf{2ABE'.pp}, \mathsf{2ABE'.msk}, \mathbf{x}_1, \alpha)$.
4. Define a function $f_1[\mathsf{ct}_1, \mathsf{ct}'_1]$, with $\mathsf{ct}_1, \mathsf{ct}'_1$ being hardwired (Figure 3).
5. Output $\mathsf{ct}''_1 = \mathsf{LO.Obf}(1^\lambda, f_1[\mathsf{ct}_1, \mathsf{ct}'_1], m, \alpha)$.

Circuit $f_1[\mathsf{ct}_1, \mathsf{ct}'_1]$

1. Parse input as $(\tilde{G}, \mathsf{sk}, \mathsf{sk}')$ where $\tilde{G}$ is regarded as an obfuscated circuit of LO, and sk and sk' are regarded as secret keys of 2ABE and 2ABE′ respectively.
2. Compute $r \leftarrow \mathsf{LO.Eval}(\tilde{G}, (\mathsf{ct}_1, \mathsf{sk}))$.
3. Output $\alpha' = \mathsf{2ABE'.Dec}(\mathsf{2ABE'.pp}, \mathsf{sk}', \mathsf{ct}'_1, r)$.

Figure 3: Circuit Obfuscated by Slot 1 Encryption

$\mathsf{Enc}_2(\mathsf{pp}, \mathsf{msk}, \mathbf{x}_2)$:

1. Parse msk as $(\mathsf{2ABE.msk}, \mathsf{2ABE'.msk})$ and pp as $(\mathsf{2ABE.pp}, \mathsf{2ABE'.pp})$.
2. Sample $\beta \leftarrow \mathcal{M}$.
3. Compute $\mathsf{ct}_2 = \mathsf{2ABE.Enc}_2(\mathsf{2ABE.pp}, \mathsf{2ABE.msk}, \mathbf{x}_2, \beta)$.
4. Compute $\mathsf{ct}'_2 = \mathsf{2ABE.Enc}'_2(\mathsf{2ABE'.pp}, \mathsf{2ABE'.msk}, \mathbf{x}_2)$.
5. Define a function $f_2[\mathsf{ct}_2]$, with ct_2 being hardwired, as in Figure 4.
6. Output $\mathsf{ct}''_2 = \mathsf{LO.Obf}(1^\lambda, f_2[\mathsf{ct}_2], \mathsf{ct}'_2, \beta)$.

Circuit $f_2[\mathsf{ct}_2]$

1. Parse input as $(\mathsf{ct}_1, \mathsf{sk})$ where ct_1 is regarded as a ciphertext of 2ABE for the first slot and sk is regarded a secret key of 2ABE.
2. Output $\beta' \leftarrow \mathsf{2ABE.Dec}(\mathsf{2ABE.pp}, \mathsf{sk}, \mathsf{ct}_1, \mathsf{ct}_2)$.

Figure 4: Circuit Obfuscated by Slot 2 Encryption

$\mathsf{Dec}(\mathsf{sk}_F, \mathsf{ct}''_1, \mathsf{ct}''_2)$: On input the secret key sk_F for function F, and 2PE ciphertexts ct''_1 and ct''_2, do the following:

1. Parse sk_F as $(\mathsf{2ABE.sk}_F, \mathsf{2ABE'.sk}_F)$.
2. Output $\mathsf{LO.Eval}(\mathsf{ct}''_1, (\mathsf{ct}''_2, \mathsf{2ABE.sk}_F, \mathsf{2ABE'.sk}_F)$.

Correctness. Recall that $\mathsf{ct}''_1 = \mathsf{LO.Obf}(1^\lambda, f_1[\mathsf{ct}_1, \mathsf{ct}'_1], m, \alpha)$. We claim that

$$f_1[\mathsf{ct}_1, \mathsf{ct}'_1](\mathsf{ct}''_2, \mathsf{2ABE.sk}_F, \mathsf{2ABE}'.\mathsf{sk}_F) = \alpha.$$

This may be argued via the following steps:

1. Recall that $\mathsf{ct}''_2 = \mathsf{LO.Obf}(1^\lambda, f_2[\mathsf{ct}_2], \mathsf{ct}'_2, \beta)$ and $f_2[\mathsf{ct}_2](\mathsf{ct}_1, \mathsf{2ABE.sk}_F) = \mathsf{2ABE.Dec}(\mathsf{2ABE.pp}, \mathsf{2ABE.sk}_F, \mathsf{ct}_1, \mathsf{ct}_2) = \beta$. The second equality follows by correctness of 2ABE and the fact that ct_1 and ct_2 encrypt β under attributes $\mathbf{x}_1, \mathbf{x}_2$. Since ct''_2 has lock value β and message value ct'_2, we have by correctness of LO that $\mathsf{LO.Eval}(\mathsf{ct}''_2, (\mathsf{ct}_1, \mathsf{2ABE.sk}_F)) = \mathsf{ct}'_2$.

2. Next, following the description of $f_1[\mathsf{ct}_1, \mathsf{ct}'_1]$ (Figure 3), we evaluate $\mathsf{2ABE}'.\mathsf{Dec}(\mathsf{2ABE}'.\mathsf{sk}_F, \mathsf{ct}'_1, \mathsf{ct}'_2)$ and recover α by correctness of 2ABE′ decryption and the construction of ct'_1 and ct'_2 as encryptions of α under attributes $\mathbf{x}_1, \mathbf{x}_2$.

Thus, we get that $f_1[\mathsf{ct}_1, \mathsf{ct}'_1](\mathsf{ct}''_2, \mathsf{2ABE.sk}_F, \mathsf{2ABE}'.\mathsf{sk}_F) = \alpha$. Now, by correctness of LO, we have that $\mathsf{LO.Eval}(\mathsf{ct}''_1, (\mathsf{ct}''_2, \mathsf{2ABE.sk}_F, \mathsf{2ABE}'.\mathsf{sk}_F)) = m$ as desired. This concludes the proof.

7.2 Security

We prove security via the following theorem.

Theorem 7.1. *Assume* LO *is a secure lockable obfuscation scheme as per Definition 2.9 and that* 2ABE *and* 2ABE′ *are secure two input* ABE *schemes satisfying strong security as in Definition 3.3 (resp., strong very selective security as in Definition 3.5). Then, the* 2PE *construction presented above satisfies strong security as per Definition 3.4 (resp., strong very selective security as in Definition 3.5).*

Proof. This proof is more complex than that of Theorem 6.2, because the adversary can make queries for decrypting keys, in which case contents of obfuscated circuits can be revealed. However, as we argue, this leakage does not compromise the security of messages that must remain hidden, because for their corresponding 2ABE ciphertexts, the protecting obfuscators will remain "locked". Moreover, the "unlocked" LO output 2ABE ciphertexts ct'_2 which cannot be used to decrypt slot 1 ciphertexts by admissibility of the adversary, and hence do not compromise security of the hidden instances. This is in contrast to the previous scheme, where a global secret SKE key K was being output after successful inner 2ABE decryption.

We focus on the case of strong security below. The case of strong very selective security is similar and simpler. The proof proceeds via a sequence of games between the challenger and a PPT adversary $\mathcal{A}$.

Game$_0$: This is the real world.

Game$_1$: This world differs from the previous in the way slot 2 ciphertext queries are answered. Let us recall that each ciphertext query is of the form

$$\begin{cases} (\mathbf{x}^i_{1,0}, \mathbf{x}^i_{1,1}), (m^i_0, m^i_1) & \text{(for slot 1), or} \\ (\mathbf{x}^i_{2,0}, \mathbf{x}^i_{2,1}) & \text{(for slot 2).} \end{cases}$$

Let $\mathcal{S}$ be the set of all those slot 2 ciphertext queries in which $\mathbf{x}^i_{2,0} \neq \mathbf{x}^i_{2,1}$. Then in this world, for queries in $\mathcal{S}$, we replace the value β encrypted in 2ABE ciphertext, ct_2 with $\mathbf{0}$.

Game$_2$: This world differs from the previous in the following ways. In this world, in response to queries in set $\mathcal{S}$, ct''_2 is simulated using the LO simulator.

Game$_3$: This world differs from the previous in the following ways. Let $\mathcal{S}'$ be the set of slot 1 ciphertext queries satisfying one of the following two conditions: (i) $\mathbf{x}^i_{1,0} \neq \mathbf{x}^i_{1,1}$ (ii) $(\mathbf{x}^i_{1,0} = \mathbf{x}^i_{1,1})$ and $(m^i_0 \neq m^i_1)$. In this world, ct'_1 encrypts $\mathbf{0}$ instead of α for all queries in $\mathcal{S}'$.

Game$_4$: This world differs from the previous in the following ways. In this world, in response to queries in set $\mathcal{S}'$, ct''_1 is simulated using the LO simulator.

Indistinguishability of Hybrids. We now show that the consecutive hybrids are indistinguishable. We let E_x denote the event that the adversary $\mathcal{A}$ outputs correct guess for the challenge bit b at the end of **Game**$_\mathrm{x}$.

Claim 7.2. *Assume that* 2ABE *satisfies strong* Ada-IND *security (Definition 3.3). Then,* **Game**$_0$ *and* **Game**$_1$ *are computationally indistinguishable. That is,*

$$|\Pr[\mathsf{E}_0] - \Pr[\mathsf{E}_1]| \leq \mathrm{negl}(\lambda).$$

Proof. We show that if $\mathcal{A}$ distinguishes between **Game**$_0$ and **Game**$_1$ with non-negligible probability then there exists an adversary $\mathcal{B}$ who can break strong Ada-IND security of 2ABE using $\mathcal{A}$. The reduction is as follows:

1. The 2ABE challenger samples $(\mathsf{2ABE.msk}, \mathsf{2ABE.pp}) \leftarrow \mathsf{2ABE.Setup}(1^\lambda)$ and $b' \leftarrow \{0,1\}$ and sends $\mathsf{2ABE.pp}$ to $\mathcal{B}$.

2. Upon receiving the public parameters $\mathsf{2ABE.pp}$ from 2ABE challenger, $\mathcal{B}$ does the following.

   (a) Samples $(\mathsf{2ABE'.msk}, \mathsf{2ABE'.pp}) \leftarrow \mathsf{2ABE'.Setup}(1^\lambda)$ and sets $\mathsf{pp} = (\mathsf{2ABE.pp}, \mathsf{2ABE'.pp})$. It implicitly sets the master secret key as $(\mathsf{msk} = (\mathsf{2ABE.msk}, \mathsf{2ABE.msk'})$.

   (b) Invokes $\mathcal{A}$ with pp as public parameters and chooses a bit $b \leftarrow \{0,1\}$.

3. $\mathcal{B}$ then responds to key queries and challenge queries from $\mathcal{A}$ as follows:

   Key Queries:

   (a) For each key query for a function F, from $\mathcal{A}$, $\mathcal{B}$ makes a key query for F to 2ABE challenger and receives $\mathsf{2ABE.sk}_F$ from the challenger.

   (b) Computes $\mathsf{2ABE'.sk}_F \leftarrow \mathsf{2ABE'.KeyGen}(\mathsf{2ABE'.pp}, \mathsf{2ABE'.msk}, F)$.

   (c) Sets $\mathsf{sk}_F = (\mathsf{2ABE.sk}_F, \mathsf{2ABE'.sk}_F)$ and returns it to $\mathcal{A}$.

   Ciphertext Queries: Each ciphertext query from $\mathcal{A}$ is of the form

   $$\begin{cases} (\mathbf{x}^i_{1,0}, \mathbf{x}^i_{1,1}), (m^i_0, m^i_1) & \text{(for slot 1), or} \\ (\mathbf{x}^i_{2,0}, \mathbf{x}^i_{2,1}) & \text{(for slot 2),} \end{cases}$$

   Upon receiving such a query, $\mathcal{B}$ does the following:

   (a) For slot 1 queries:

      i. Samples $\alpha \leftarrow \mathcal{M}$.

      ii. Sends $\mathbf{x}^i_{1,b}$ to 2ABE challenger as slot 1 ciphertext query, to which the 2ABE challenger replies with ct_1.

iii. Computes $\mathsf{ct}'_1 = \mathsf{2ABE}'.\mathsf{Enc}_1(\mathsf{2ABE}'.\mathsf{pp}, \mathsf{2ABE}'.\mathsf{msk}, \mathbf{x}^i_{1b}, \alpha)$

iv. Defines $f_1[\mathsf{ct}_1, \mathsf{ct}'_1]$ and computes

$$\mathsf{ct}''_1 = \mathsf{LO.Obf}(1^\lambda, f_1[\mathsf{ct}_1, \mathsf{ct}'_1], m^i_b, \alpha).$$

v. Returns ct''_1 to $\mathcal{A}$.

(b) For slot 2 queries:

i. Computes $\mathsf{ct}'_2 = \mathsf{2ABE}'.\mathsf{Enc}_2(\mathsf{2ABE}'.\mathsf{pp}, \mathsf{2ABE}'.\mathsf{msk}, \mathbf{x}^i_{2,b})$

ii. Samples $\beta \leftarrow \mathcal{M}$.

iii. If $\mathbf{x}^i_{2,0} \neq \mathbf{x}^i_{2,1}$, then sets $\mu^i_0 = \beta, \mu^i_1 = \mathbf{0}$, else sets $\mu^i_0 = \beta, \mu^i_1 = \beta$.

iv. Sends $\mathbf{x}^i_{2,b}, (\mu^i_0, \mu^i_1)$ as ciphertext query for slot 2 to the 2ABE challenger.

v. The 2ABE challenger computes and returns slot 2 ciphertext as

$$\mathsf{ct}_2 = \mathsf{2ABE.Enc}_2(\mathsf{2ABE.pp}, \mathsf{2ABE.msk}, \mathbf{x}^i_{2,b}, \mu^i_{b'}),$$

where b' is the coin chosen by the challenger, which is fixed throughout the game.

vi. $\mathcal{B}$ defines $f_2[\mathsf{ct}_2]$ and computes

$$\mathsf{ct}''_2 = \mathsf{LO.Obf}(1^\lambda, f_2[\mathsf{ct}_2], \mathsf{ct}'_2, \beta).$$

vii. Returns ct''_2 to $\mathcal{A}$.

4. In the end, $\mathcal{A}$ outputs a bit $\hat{b}$. If $\hat{b} = b$, then $\mathcal{B}$ returns $b'' = 1$, else $b'' = 0$, to the 2ABE challenger.

We can observe that if $b' = 0$, then $\mathcal{B}$ simulated $\mathbf{Game}_0$, else $\mathbf{Game}_1$. Hence, if $\mathcal{A}$ can distinguish between the two hybrids, then $\mathcal{B}$ can win against 2ABE challenger.

Assuming strong Ada-IND security of 2ABE, we get

$$|\Pr[\mathsf{E}_0] - \Pr[\mathsf{E}_1]| \leq \mathrm{negl}(\lambda).$$

Admissibility of $\mathcal{B}$: We show that if $\mathcal{A}$ is admissible for strong 2PE security then $\mathcal{B}$ is also admissible for strong 2ABE security. Observe that the key queries issued by $\mathcal{B}$ to the 2ABE challenger are the same key queries as issued by $\mathcal{A}$ to $\mathcal{B}$. Consider the challenge queries issued by $\mathcal{B}$. If for some function F for which key query has been made, $F(\mathbf{x}^{j_1}_{1,b}, \mathbf{x}^{j_2}_{2,b}) = 1$ then we need to ensure that $\mu^{j_2}_0 = \mu^{j_2}_1$ (since 2ABE encrypts message in slot 2, the message equality condition is required for query index j_2, i.e. corresponding to slot 2). Since, the ciphertext queries with the same attributes are issued by $\mathcal{A}$ to $\mathcal{B}$, admissibility of $\mathcal{A}$ demands that $\mathbf{x}^{j_2}_{2,0} = \mathbf{x}^{j_2}_{2,1}$. But, when $\mathbf{x}^{j_2}_{2,0} = \mathbf{x}^{j_2}_{2,1}$, $\mathcal{B}$ takes $\mu^{j_2}_0 = \mu^{j_2}_1 = \beta$, as desired. □

Claim 7.3. *Assume that* LO *is a secure lockable obfuscation scheme (Definition 2.9). Then* $\mathbf{Game}_1$ *and* $\mathbf{Game}_2$ *are computationally indistinguishable. That is,*

$$|\Pr[\mathsf{E}_1] - \Pr[\mathsf{E}_2]| \leq \mathrm{negl}(\lambda).$$

Proof. We show that if $\mathcal{A}$ can distinguish between $\mathbf{Game}_1$ and $\mathbf{Game}_2$ with non-negligible probability then there exists an adversary $\mathcal{B}$ who can distinguish between LO obfuscated programs and simulated programs using $\mathcal{A}$, thus breaking the security of LO. The reduction is as follows:

1. Upon being challenged by LO challenger, $\mathcal{B}$ does the following:

(a) Samples public parameters and master secret for 2PE as $(\mathsf{pp}, \mathsf{msk} = (\mathsf{2ABE.msk}, \mathsf{2ABE'.msk})) \leftarrow \mathsf{Setup}(1^\lambda)$ and invokes $\mathcal{A}$ with pp. $\mathcal{B}$ also samples a bit b.

(b) $\mathcal{A}$ issues polynomially many key queries and ciphertext queries to which $\mathcal{B}$ responds as follows:

Key Queries:
For each key query for a function F from $\mathcal{A}$, $\mathcal{B}$ returns $\mathsf{sk}_F \leftarrow \mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, F)$ to $\mathcal{A}$.
Ciphertext Queries: To answer each ciphertext query, which is of the form

$$\begin{cases} (\mathbf{x}^i_{1,0}, \mathbf{x}^i_{1,1}), (m^i_0, m^i_1) & \text{(for slot 1), or} \\ (\mathbf{x}^i_{2,0}, \mathbf{x}^i_{2,1}) & \text{(for slot 2),} \end{cases}$$

$\mathcal{B}$ does the following:

i. For slot 1 queries:
   A. Samples $\alpha \leftarrow \mathcal{M}$.
   B. Computes

$$\mathsf{ct}_1 = \mathsf{2ABE.Enc}_1(\mathsf{2ABE.pp}, \mathsf{2ABE.msk}, \mathbf{x}^i_{1,b})$$

   and

$$\mathsf{ct}'_1 = \mathsf{2ABE'.Enc}_1(\mathsf{2ABE'.pp}, \mathsf{2ABE'.msk}, \mathbf{x}^i_{1,b}, \alpha)$$

   .
   C. Defines $f_1[\mathsf{ct}_1, \mathsf{ct}'_1]$ and computes

$$\mathsf{ct}''_1 = \mathsf{LO.Obf}(1^\lambda, f_1[\mathsf{ct}_1, \mathsf{ct}'_1], m^i_b, \alpha).$$

   D. Sends ct''_1 to $\mathcal{A}$.

ii. For slot 2 queries:
   A. Computes $\mathsf{ct}'_2 = \mathsf{2ABE'.Enc}_2(\mathsf{2ABE'.pp}, \mathsf{2ABE'.msk}, \mathbf{x}^i_{2,b})$.
   B. If $\mathbf{x}^i_{2,0} = \mathbf{x}^i_{2,1}$,
      - Samples $\beta \leftarrow \mathcal{M}$
      - Computes $\mathsf{ct}_2 = \mathsf{2ABE.Enc}_2(\mathsf{2ABE.pp}, \mathsf{2ABE.msk}, \mathbf{x}^i_{2,b}, \beta)$.
      - Defines $f_2[\mathsf{ct}_2]$ and computes

$$\mathsf{ct}''_2 = \mathsf{LO.Obf}(1^\lambda, f_2[\mathsf{ct}_2], \mathsf{ct}'_2, \beta).$$

   C. Else if $\mathbf{x}^i_{2,0} \neq \mathbf{x}^i_{2,1}$,
      - Computes $\mathsf{ct}_2 = \mathsf{2ABE.Enc}_2(\mathsf{2ABE.pp}, \mathsf{2ABE.msk}, \mathbf{x}^i_{2,b}, \mathbf{0})$, defines $f_2[\mathsf{ct}_2]$ and sends it along with ct'_2 to the LO challenger.
      - The LO challenger returns either an obfuscated circuit or a simulated circuit which $\mathcal{B}$ sets as ct''_2.
   D. Sends ct''_2 to $\mathcal{A}$.

(c) In the end, $\mathcal{A}$ outputs a bit $\hat{b}$. If $\hat{b} = b$, then $\mathcal{B}$ returns $b'' = 1$, else $b'' = 0$, to LO challenger.

We can observe that if the LO challenger returned obfuscated programs, then $\mathcal{B}$ simulated $\mathbf{Game}_1$, else if LO challenger returned simulated programs, then $\mathcal{B}$ simulated $\mathbf{Game}_2$. Hence, if $\mathcal{A}$ distinguishes between the two games, then so does $\mathcal{B}$ between obfuscated and simulated programs. Assuming LO is secure, we get

$$|\Pr[\mathsf{E}_1] - \Pr[\mathsf{E}_2]| \leq \mathrm{negl}(\lambda).$$

□

Claim 7.4. *Assume that* 2ABE′ *satisfies strong* Ada-IND *security (Definition 3.3). Then,* $\mathbf{Game}_2$ *and* $\mathbf{Game}_3$ *are computationally indistinguishable. That is,*

$$|\Pr[\mathsf{E}_2] - \Pr[\mathsf{E}_3]| \leq \mathrm{negl}(\lambda).$$

Proof. We show that if $\mathcal{A}$ can distinguish between $\mathbf{Game}_2$ and $\mathbf{Game}_3$ with non-negligible probability then there exists an adversary $\mathcal{B}$ who can break strong Ada-IND security of 2ABE′ using $\mathcal{A}$. The reduction is as follows:

1. The 2ABE′ challenger samples $(\mathsf{2ABE'.msk}, \mathsf{2ABE'.pp}) \leftarrow \mathsf{2ABE'.Setup}(1^\lambda)$ and $b' \leftarrow \{0,1\}$ and sends $\mathsf{2ABE'.pp}$ to $\mathcal{B}$.
2. Upon receiving the public parameters $\mathsf{2ABE'.pp}$ from 2ABE′ challenger, $\mathcal{B}$ does the following.
   (a) Samples $(\mathsf{2ABE.msk}, \mathsf{2ABE.pp}) \leftarrow \mathsf{2ABE.Setup}(1^\lambda)$ and sets $\mathsf{pp} = (\mathsf{2ABE.pp}, \mathsf{2ABE'.pp})$. $\mathcal{B}$ implicitly sets $\mathsf{msk} = (\mathsf{2ABE.msk}, \mathsf{2ABE'.msk})$.
   (b) Invokes $\mathcal{A}$ with pp as public parameters and chooses a bit $b \leftarrow \{0,1\}$.
3. $\mathcal{B}$ then responds to key queries and ciphertext queries from $\mathcal{A}$ as follows:

   Key Queries:

   (a) Upon receiving a key query for function F from $\mathcal{A}$, $\mathcal{B}$ makes a key query for F to 2ABE′ challenger and receives $\mathsf{2ABE'.sk}_F$ from 2ABE′ challenger.
   (b) Computes $\mathsf{2ABE.sk}_F \leftarrow \mathsf{2ABE.KeyGen}(\mathsf{2ABE.pp}, \mathsf{2ABE.msk}, F)$.
   (c) Sets $\mathsf{sk}_F = (\mathsf{2ABE.sk}_F, \mathsf{2ABE'.sk}_F)$ and returns it to $\mathcal{A}$.

   Ciphertext Queries: Each ciphertext query i from $\mathcal{A}$ is of the form

   $$\begin{cases} (\mathbf{x}^i_{1,0}, \mathbf{x}^i_{1,1}), (m^i_0, m^i_1) & \text{(for slot 1), or} \\ (\mathbf{x}^i_{2,0}, \mathbf{x}^i_{2,1}) & \text{(for slot 2),} \end{cases}$$

   Upon receiving such a query, $\mathcal{B}$ does the following:

   (a) For slot 1 queries:
      i. Computes $\mathsf{ct}_1 = \mathsf{2ABE.Enc}_1(\mathsf{2ABE.pp}, \mathsf{2ABE.msk}, \mathbf{x}^i_{1,b})$
      ii. Samples $\alpha \leftarrow \mathcal{M}$.
      iii. If the $\mathsf{query}_i \in \mathcal{S}'$, i.e.
      $(\mathbf{x}^i_{1,0} \neq \mathbf{x}^i_{1,1})$ OR $(\mathbf{x}^i_{1,0} = \mathbf{x}^i_{1,1})$ and $(m^i_0 \neq m^i_1))$
         - Sets $\mu^i_0 = \alpha$ and $\mu^i_1 = \mathbf{0}$ and sends ciphertext query $(\mathbf{x}^i_{1,b}, (\mu^i_0, \mu^i_1))$ to the 2ABE′ challenger.

- The $\mathsf{2ABE}'$ challenger computes and returns slot 1 ciphertext as

$$\mathsf{ct}'_1 = \mathsf{2ABE}'.\mathsf{Enc}_1(\mathsf{2ABE}'.\mathsf{pp}, \mathsf{2ABE}'.\mathsf{msk}, \mathbf{x}^i_{1,b}, \mu^i_{b'}).$$

iv. Else, if the $\mathsf{query}_i \notin \mathcal{S}'$, i.e. $(\mathbf{x}^i_{1,0} = \mathbf{x}^i_{1,1})$ and $(m^i_0 = m^i_1)$
- Sets $\mu^i_0 = \alpha$ and $\mu^i_1 = \alpha$ and sends ciphertext query $(\mathbf{x}^i_{1,b}, (\mu^i_0, \mu^i_1))$ to the $\mathsf{2ABE}'$ challenger.
- The $\mathsf{2ABE}'$ challenger computes and returns slot 1 ciphertext as

$$\mathsf{ct}'_1 = \mathsf{2ABE}'.\mathsf{Enc}_1(\mathsf{2ABE}'.\mathsf{pp}, \mathsf{2ABE}'.\mathsf{msk}, \mathbf{x}^i_{1,b}, \mu^i_{b'}),$$

where b' is the coin chosen by the challenger, which is fixed throughout the game.

v. $\mathcal{B}$ defines $f_1[\mathsf{ct}_1, \mathsf{ct}'_1]$ and computes

$$\mathsf{ct}''_1 = \mathsf{LO.Obf}(1^\lambda, f_1[\mathsf{ct}_1, \mathsf{ct}'_1], m^i_b, \alpha).$$

vi. Sends ct''_1 to $\mathcal{A}$.

(b) For slot 2 queries:

i. Sends ciphertext query $\mathbf{x}^i_{2,b}$ for slot 2 to the $\mathsf{2ABE}'$ challenger. The $\mathsf{2ABE}'$ challenger computes and returns slot 2 ciphertext as

$$\mathsf{ct}'_2 = \mathsf{2ABE}'.\mathsf{Enc}_2(\mathsf{2ABE}'.\mathsf{pp}, \mathsf{2ABE}'.\mathsf{msk}, \mathbf{x}^i_{2,b}).$$

ii. If the $\mathsf{query}_i \in \mathcal{S}$, i.e. $(\mathbf{x}^i_{2,0} \neq \mathbf{x}^i_{2,1})$
- Computes $\mathsf{ct}_2 = \mathsf{2ABE.Enc}_2(\mathsf{2ABE.pp}, \mathsf{2ABE.msk}, \mathbf{x}^i_{2,b}, \mathbf{0})$.
- Defines $f_2[\mathsf{ct}_2]$ and simulates $\mathsf{ct}''_2 = \mathsf{LO.Sim}(1^\lambda, 1^{|f_2[\mathsf{ct}_2]|}, 1^{|\mathsf{ct}'_2|})$.

iii. If the $\mathsf{query}_i \notin \mathcal{S}$, i.e. $(\mathbf{x}^i_{2,0} = \mathbf{x}^i_{2,1})$
- Samples $\beta \leftarrow \mathcal{M}$ and computes $\mathsf{ct}_2 = \mathsf{2ABE.Enc}_2(\mathsf{2ABE.pp}, \mathsf{2ABE.msk}, \mathbf{x}^i_{2,b}, \beta)$.
- Defines $f_2[\mathsf{ct}_2]$ and computes $\mathsf{ct}''_2 = \mathsf{LO.Obf}(1^\lambda, f_2[\mathsf{ct}_2], \mathsf{ct}'_2, \beta)$.

iv. Sends ct''_2 to $\mathcal{A}$.

4. In the end, $\mathcal{A}$ outputs a bit $\hat{b}$. If $\hat{b} = b$, then $\mathcal{B}$ returns $b'' = 1$, else $b'' = 0$, to the $\mathsf{2ABE}'$ challenger.

We can observe that if $b' = 0$, then $\mathcal{B}$ simulated $\mathbf{Game}_2$, else $\mathbf{Game}_3$. Hence, if $\mathcal{A}$ distinguishes between the two hybrids, then $\mathcal{B}$ wins against $\mathsf{2ABE}'$ challenger.

Assuming strong $\mathsf{Ada\text{-}IND}$ security of $\mathsf{2ABE}'$, we get

$$|\Pr[\mathsf{E}_2] - \Pr[\mathsf{E}_3]| \leq \mathrm{negl}(\lambda).$$

Admissibility of $\mathcal{B}$: We show that if $\mathcal{A}$ is admissible for strong $\mathsf{2PE}$ security then $\mathcal{B}$ is also admissible for strong $\mathsf{2ABE}'$ security. Observe that the key queries issued by $\mathcal{B}$ are the same key queries as issued by $\mathcal{A}$. Now consider the challenge queries issued by $\mathcal{B}$. If there is a function F for which key has been queried, such that $F(\mathbf{x}^{j_1}_{1,b}, \mathbf{x}^{j_2}_{2,b}) = 1$ then we need to ensure that $\mu^{j_1}_0 = \mu^{j_1}_1$ (since, $\mathsf{2ABE}'$ encrypts message in slot 1, the message equality condition is required for query index j_1). Since, the challenge queries for the same attributes are issued by $\mathcal{A}$ to $\mathcal{B}$, admissibility of $\mathcal{A}$ demands that $\mathbf{x}^{j_1}_{1,0} = \mathbf{x}^{j_1}_{1,1}$ and $m^{j_1}_0 = m^{j_1}_1$. But, in this case, $\mathcal{B}$ takes $\mu^{j_1}_0 = \mu^{j_1}_1 = \alpha$, as desired.

□

Claim 7.5. *Assume that* LO *is a secure lockable obfuscation scheme (Definition 2.9). Then* $\mathbf{Game}_3$ *and* $\mathbf{Game}_4$ *are computationally indistinguishable. That is,*

$$|\Pr[\mathsf{E}_3] - \Pr[\mathsf{E}_4]| \leq \mathrm{negl}(\lambda).$$

Proof. We show that if $\mathcal{A}$ can distinguish between $\mathbf{Game}_3$ and $\mathbf{Game}_4$ with non-negligible probability then there exists an adversary $\mathcal{B}$ who can distinguish between LO obfuscated programs and simulated programs using $\mathcal{A}$, thus breaking the security of LO. The reduction is as follows:

1. Upon being challenged by LO challenger, $\mathcal{B}$ does the following:

   (a) Samples public parameters and master secret for 2PE as $(\mathsf{pp}, \mathsf{msk}{=}(\mathsf{2ABE.msk}, \mathsf{2ABE'.msk})) \leftarrow \mathsf{Setup}(1^\lambda)$ and invokes $\mathcal{A}$ with pp. $\mathcal{B}$ also samples a bit b.

   (b) $\mathcal{A}$ issues polynomially many key queries and ciphertext queries to which $\mathcal{B}$ responds as follows:

   Key Queries:
   For each key query for a function F from $\mathcal{A}$, $\mathcal{B}$ returns $\mathsf{sk}_F \leftarrow \mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, F)$ to $\mathcal{A}$.

   Ciphertext Queries: To answer each ciphertext query, which is of the form

   $$\begin{cases} (\mathbf{x}^i_{1,0}, \mathbf{x}^i_{1,1}), (m^i_0, m^i_1) & \text{(for slot 1), or} \\ (\mathbf{x}^i_{2,0}, \mathbf{x}^i_{2,1}) & \text{(for slot 2),} \end{cases}$$

   $\mathcal{B}$ does the following:

   i. For slot 1 queries:
      A. Computes $\mathsf{ct}_1 = \mathsf{2ABE.Enc}_1(\mathsf{2ABE.pp}, \mathsf{2ABE.msk}, \mathbf{x}^i_{1,b})$.
      B. If $(\mathbf{x}^i_{1,0} = \mathbf{x}^i_{1,1})$ and $(m^i_0 = m^i_1)$,
         - Samples $\alpha \leftarrow \mathcal{M}$.
         - Computes $\mathsf{ct}'_1 = \mathsf{2ABE'.Enc}_1(\mathsf{2ABE'.pp}, \mathsf{2ABE'.msk}, \mathbf{x}^i_{1,b}, \alpha)$.
         - Computes
         $$\mathsf{ct}''_1 = \mathsf{LO.Obf}(1^\lambda, f_1[\mathsf{ct}_1, \mathsf{ct}'_1], m^i_b, \alpha).$$
      C. Else,
         - Computes $\mathsf{ct}'_1 = \mathsf{2ABE'.Enc}_1(\mathsf{2ABE'.pp}, \mathsf{2ABE'.msk}, \mathbf{x}^i_{1,b}, \mathbf{0})$ and defines $f_1[\mathsf{ct}_1, \mathsf{ct}'_1]$.
         - Sends $f_1[\mathsf{ct}_1, \mathsf{ct}'_1], m^i_b$ to LO challenger.
         - The LO challenger returns either an obfuscated or a simulated program, which $\mathcal{B}$ sets as ct''_1.
      D. $\mathcal{B}$ sends ct''_1 to $\mathcal{A}$.
   ii. For slot 2 queries:
      A. Computes $\mathsf{ct}'_2 = \mathsf{2ABE'.Enc}_2(\mathsf{2ABE'.pp}, \mathsf{2ABE'.msk}, \mathbf{x}^i_{2,b})$.
      B. If $\mathbf{x}^i_{2,0} = \mathbf{x}^i_{2,1}$,
         - Samples $\beta \leftarrow \mathcal{M}$
         - Computes $\mathsf{ct}_2 = \mathsf{2ABE.Enc}_2(\mathsf{2ABE.pp}, \mathsf{2ABE.msk}, \mathbf{x}^i_{2,b}, \beta)$.
         - Defines $f_2[\mathsf{ct}_2]$ and computes
         $$\mathsf{ct}''_2 = \mathsf{LO.Obf}(1^\lambda, f_2[\mathsf{ct}_2], \mathsf{ct}'_2, \beta).$$

C. Else if $\mathbf{x}^i_{2,0} \neq \mathbf{x}^i_{2,1}$,
- Computes $\mathsf{ct}_2 = \mathsf{2ABE.Enc}_2(\mathsf{2ABE.pp}, \mathsf{2ABE.msk}, \mathbf{x}^i_{2,b}, \mathbf{0})$.
- Defines $f_2[\mathsf{ct}_2]$ and simulates

$$\mathsf{ct}''_2 = \mathsf{LO.Sim}(1^\lambda, 1^{|f_2[\mathsf{ct}_2]|}, 1^{|\mathsf{ct}'_2|}).$$

D. Sends ct''_2 to $\mathcal{A}$.

(c) In the end, $\mathcal{A}$ outputs a bit $\hat{b}$. If $\hat{b} = b$, then $\mathcal{B}$ returns $b'' = 1$, else $b'' = 0$, to LO challenger.

If the LO challenger returned obfuscated programs, then $\mathcal{B}$ simulated $\mathbf{Game}_3$, else if LO challenger returned simulated programs, then $\mathcal{B}$ simulated $\mathbf{Game}_4$. Hence, if $\mathcal{A}$ distinguishes between the two games, then so does $\mathcal{B}$ between obfuscated and simulated programs. Assuming LO is secure, we get

$$|\Pr[\mathsf{E}_3] - \Pr[\mathsf{E}_4]| \leq \mathrm{negl}(\lambda).$$

□

Claim 7.6. $\Pr[\mathsf{E}_4] - \frac{1}{2} = 0$

Proof. We argue that the adversary cannot obtain any information of b in $\mathbf{Game}_4$. We first observe that the only possible way for the adversary to learn information of b is to make challenge ciphertext queries, since decrypting ciphertexts do not convey any information of b. However, responses to challenge ciphertext queries does not convey any information of b either as we see below. In $\mathbf{Game}_4$, a challenge ciphertext for slot 1 is computed as $\mathsf{ct}''_1 = \mathsf{LO.Sim}(1^\lambda, 1^{|f_1[\mathsf{ct}_1, \mathsf{ct}'_1]|}, 1^{|m^i_b|})$. The distribution of ct''_1 depends only on the lengths of function $f_1[\mathsf{ct}_1, \mathsf{ct}'_1]$ and message m^i_b. We observe that $|f_1[\mathsf{ct}_1, \mathsf{ct}'_1]|$ further depends only on the lengths of ct_1 and ct'_1, which in turn depends only on the lengths of the underlying message and attribute. Similarly, challenge ciphertext for slot 2 is computed as $\mathsf{ct}''_2 = \mathsf{LO.Sim}(1^\lambda, 1^{|f_2[\mathsf{ct}_2]|}, 1^{|\mathsf{ct}'_2|})$. We can see that ct''_2 does not convey any information of b because of the same reason as above.

□

□

Applications. By applying the above conversion to two input ABE scheme with strong security in Sec. 4, we obtain a candidate construction of two input PE scheme with strong security. A caveat here is that the resulting scheme cannot necessarily be proven secure under LWE in the bilinear generic group model as one might expect. The problem here is that our conversion uses the decryption algorithm of the underlying two input ABE scheme in a non-black box way, which especially uses the code of the group operations. To claim the security of the resulting scheme, we heuristically assume that the two-input ABE scheme in Sec. 4 is strongly secure even in the standard model if we implement the bilinear generic group model with concrete well-chosen bilinear group and then apply the above conversion. We note that this kind of heuristic instantiation is widely used in the context of cryptographic hash functions and bilinear maps. We also mention that we can apply the above conversion to the two input ABE scheme in Sec. 5. Since the scheme is proven secure in the standard model, the construction does not suffer from the above problem.

8 Three-Input ABE from Pairings and Lattices

In this section, we provide a candidate construction for 3ABE using the structure of [BV22] and [AY20] as discussed in Section 1. Leveraging ideas from the Brakerski-Vaikuntanathan construction [BV22], we also obtain a candidate for 2ABE for P – we provide this construction in Section 9. Our 3ABE scheme supports NC_1 circuits. More formally, it supports attribute space $A_\lambda = \{0,1\}^{\ell(\lambda)}$ and any circuit class $\mathcal{F} = \{\mathcal{F}_\lambda\}_\lambda$ that is subclass of $\{\mathcal{C}_{3\ell(\lambda),d(\lambda)}\}_\lambda$ with arbitrary $\ell(\lambda) \leq \mathrm{poly}(\lambda)$ and $d(\lambda) = O(\log \lambda)$, where $\mathcal{C}_{3\ell(\lambda),d(\lambda)}$ is a set of circuits with input length $3\ell(\lambda)$ and depth at most $d(\lambda)$.

8.1 Construction

The construction is defined as follows:

$\mathsf{Setup}(1^\lambda)$: On input 1^λ, the setup algorithm defines the parameters $n = n(\lambda)$, $m = m(\lambda)$, $k = k(\lambda)$, noise distribution χ, $\hat{\chi}$ over $\mathbb{Z}$, $\tau_0 = \tau_0(\lambda)$, $\tau = \tau(\lambda)$, $\tau_0' = \tau_0'(\lambda)$, $\tau_t = \tau_t(\lambda)$ and $B = B(\lambda)$ as specified in Sec. 8.2. It samples a group description $\mathbb{G} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, [1]_1, [1]_2)$. It then sets $L := (5\ell+1)m + 1$ and proceeds as follows.

1. Samples BGG^+ scheme:
   (a) Samples $(\mathbf{A}, \mathbf{A}_{\tau_0}^{-1}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$ such that $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$.
   (b) Samples random matrix $\mathbf{B} = (\mathbf{B}_1, \ldots, \mathbf{B}_{3\ell}) \leftarrow (\mathbb{Z}_q^{n\times m})^{3\ell}$ and a random vector $\mathbf{u} \leftarrow \mathbb{Z}_q^n$.
2. Samples $w_0 \leftarrow \mathbb{Z}_q^*$, $\mathbf{W} \leftarrow (\mathbb{Z}_q^*)^{k\times L}$.
3. Samples BV scheme:
   (a) Samples $\mathbf{C}$ along with its trapdoor $\mathbf{C}_{\tau_0'}^{-1}$ as
   $(\mathbf{C}, \mathbf{C}_{\tau_0'}^{-1}) \leftarrow \mathsf{TrapGen}(1^{2(\ell+1)n}, 1^k, q)$, where
   $\mathbf{C}^\top = (\mathbf{C}_{2\ell+1,0} \| \mathbf{C}_{2\ell+1,1} \| \ldots \| \mathbf{C}_{3\ell,0} \| \mathbf{C}_{3\ell,1} \| \mathbf{C}_{3\ell+1} \| \mathbf{C}_{3\ell+2}) \in (\mathbb{Z}_q^{k\times n})^{2(\ell+1)}$.
4. Outputs
$$\mathsf{pp} = (\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{u}), \ \mathsf{msk} = \left(\mathbf{A}_{\tau_0}^{-1}, \mathbf{C}_{\tau_0'}^{-1}, w_0, \mathbf{W} \right).$$

$\mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, F)$: On input the public parameters pp, master secret key msk and a circuit F, compute BGG^+ function key for circuit F as follows:

1. Compute $\mathbf{H}_F = \mathsf{EvalF}(\mathbf{B}, F)$ and $\mathbf{B}_F = \mathbf{B}\mathbf{H}_F$.
2. Compute $[\mathbf{A}\|\mathbf{B}_F]_\tau^{-1}$ from $\mathbf{A}_{\tau_0}^{-1}$ and sample $\mathbf{r} \in \mathbb{Z}^{2m}$ as $\mathbf{r}^\top \leftarrow [\mathbf{A}\|\mathbf{B}_F]_\tau^{-1}(\mathbf{u}^\top)$.
3. Output the secret key $\mathsf{sk}_F := \mathbf{r}$.

$\mathsf{Enc}_1(\mathsf{pp}, \mathsf{msk}, \mathbf{x}_1, \mu)$: On input the public parameters pp, master secret key msk, attribute vector $\mathbf{x}_1$, message bit μ, encryption for slot 1 is defined as follows:

1. Set $\mathbf{m} = \lceil \frac{q}{K} \rceil \mu(1, \ldots, 1) \in \mathbb{Z}_q^k$. We define $K = 2\tau_t\sqrt{n}k$. .
2. Samples LWE secret $\mathbf{S} \leftarrow \mathbb{Z}_q^{k\times n}$ and error terms $\mathbf{e}_0 \leftarrow \chi^k$, $\mathbf{E} \leftarrow \chi^{k\times m}$, $\mathbf{E}_{i,x_{1,i}} \leftarrow \hat{\chi}^{k\times m}$, for $i \in [\ell]$, and $\mathbf{E}_{i,b} \leftarrow \hat{\chi}^{k\times m}$, for $i \in [\ell+1, 3\ell]$ and $b \in \{0,1\}$.
3. For $i \in [\ell]$, computes
$$\psi_{i,x_{1,i}} := \mathbf{S}(\mathbf{B}_i - x_{1,i}\mathbf{G}) + \mathbf{E}_{i,x_{1,i}} \in \mathbb{Z}_q^{k\times m}.$$

4. For $i \in [\ell+1, 3\ell]$, $b \in \{0,1\}$, computes

$$\psi_{i,b} := \mathbf{S}(\mathbf{B}_i - b\mathbf{G}) + \mathbf{E}_{i,b} \in \mathbb{Z}_q^{k\times m}.$$

5. Computes $\psi_{3\ell+1} := \mathbf{SA} + \mathbf{E} \in \mathbb{Z}_q^{k\times m}$ and $\psi_{3\ell+2}^\top := \mathbf{Su}^\top + \mathbf{e}_0^\top \in \mathbb{Z}_q^{k\times 1}$.
6. Sample $\widehat{\mathbf{S}}_{3\ell+1} \leftarrow \mathbb{Z}_q^{n\times m}$, $\widehat{\mathbf{s}}_{3\ell+2} \leftarrow \mathbb{Z}_q^n$, $\{\widehat{\mathbf{S}}_{2\ell+i,b}\}_{i\in[\ell],b\in\{0,1\}} \leftarrow (\mathbb{Z}_q^{n\times m})^{2\ell}$, $\widehat{\mathbf{E}} \leftarrow \chi^{k\times m}$, $\widehat{\mathbf{e}}_0 \leftarrow \chi^k$, $\widehat{\mathbf{E}}_{2\ell+i,b} \leftarrow \hat{\chi}^{k\times m}$ for $i \in [\ell], b \in \{0,1\}$.
7. Compute all possible "BV encodings" for slot 3 attribute $\mathbf{x}_3$ and construct $\widehat{\mathbf{C}}_1$ as follows:

$$\widehat{\mathbf{C}}_1 = (\{\psi_{i,x_{1i}}\}_{i\in[\ell]}, \{\psi_{i,b}\}_{\substack{i\in[\ell+1,2\ell],\\ b\in\{0,1\}}}, \{\mathbf{C}_{i,b}\widehat{\mathbf{S}}_{i,b} + \widehat{\mathbf{E}}_{i,b} + \psi_{i,b}\}_{\substack{i\in[2\ell+1,3\ell],\\ b\in\{0,1\}}},$$
$$\mathbf{C}_{3\ell+1}\widehat{\mathbf{S}}_{3\ell+1} + \widehat{\mathbf{E}} + \psi_{3\ell+1}, \mathbf{C}_{3\ell+2}\widehat{\mathbf{s}}_{3\ell+2}^\top + \widehat{\mathbf{e}}_0^\top + \psi_{3\ell+2}^\top + \mathbf{m}^\top) \in \mathbb{Z}_q^{k\times L}$$

   Here, we assume that the entries of $\widehat{\mathbf{C}}_1$ are vectorized in some fixed order.
8. Sample $t_{\mathbf{x}_1} \leftarrow \mathbb{Z}_q^*$ and
9. Output $\mathsf{ct}_1 = ([t_{\mathbf{x}_1} w_0]_1, [t_{\mathbf{x}_1}\widehat{\mathbf{C}}_1 \odot \mathbf{W}]_1$.

$\mathsf{Enc}_2(\mathsf{pp}, \mathsf{msk}, \mathbf{x}_2)$: On input the public parameters pp, master secret key msk, attribute vector $\mathbf{x}_2$, encryption for slot 2 is defined as follows:

1. Set $\widehat{\mathbf{C}}_2 = (\mathbf{1}_{k\times\ell m}, \{\hat{\psi}_{\ell+i,x_{2,i}}\}_{i\in[\ell]}, \mathbf{1}_{k\times 2\ell m}, \mathbf{1}_{k\times m}, \mathbf{1}_{k\times 1})$, where

$$\hat{\psi}_{\ell+i,b} := \begin{cases} \mathbf{1}_m \in \mathbb{Z}_q^m & \text{if } b = x_{2,i} \\ \mathbf{0}_m \in \mathbb{Z}_q^m & \text{if } b \neq x_{2,i} \end{cases} \quad \text{for } i \in [\ell] \text{ and } b \in \{0,1\}.$$

2. Sample $t_{\mathbf{x}_2} \leftarrow \mathbb{Z}_q^*$ and output $\mathsf{ct}_2 = ([t_{\mathbf{x}_2}/w_0]_2, [t_{\mathbf{x}_2}\widehat{\mathbf{C}}_2 \oslash \mathbf{W}]_2$.

$\mathsf{Enc}_3(\mathsf{pp}, \mathsf{msk}, \mathbf{x}_3)$: Given input the public parameters pp, master secret key msk, attribute vector $\mathbf{x}_3$, encryption for slot 3 is defined as follows:

1. Compute $[(\mathbf{C}_{2\ell+1,\mathbf{x}_{3,1}} \| \ldots \| \mathbf{C}_{3\ell,\mathbf{x}_{3,\ell}} \| \mathbf{C}_{3\ell+1} \| \mathbf{C}_{3\ell+2})^\top]_{\tau_t}^{-1}$ from $\mathbf{C}_{\tau_0'}^{-1}$ and sample short vector $\mathbf{t}_{\mathbf{x}_3}$ such that
   $\mathbf{t}_{\mathbf{x}_3}\mathbf{C}_{2\ell+i,\mathbf{x}_{3,i}} = \mathbf{0}$ for all $i \in [\ell]$, $\mathbf{t}_{\mathbf{x}_3}\mathbf{C}_{3\ell+1} = \mathbf{t}_{\mathbf{x}_3}\mathbf{C}_{3\ell+2} = \mathbf{0}$, as
   $\mathbf{t}_{\mathbf{x}_3}^\top \leftarrow [(\mathbf{C}_{2\ell+1,\mathbf{x}_{3,1}} \| \ldots \| \mathbf{C}_{3\ell,\mathbf{x}_{3,\ell}} \| \mathbf{C}_{3\ell+1} \| \mathbf{C}_{3\ell+2})^\top]_{\tau_t}^{-1}(\mathbf{0}^\top)$.
2. Output $\mathsf{ct}_3 = \mathbf{t}_{\mathbf{x}_3}$.

$\mathsf{Dec}(\mathsf{pp}, \mathsf{sk}_F, \mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3)$: On input the public parameters pp, the secret key sk_F for circuit F and ciphertexts ct_1, ct_2 and ct_3 corresponding to the three attributes $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$, the decryption algorithm proceeds as follows:

1. Takes the coordinate-wise pairing between ciphertexts for slot 1 and slot 2:
   Computes $[v_0]_T = [t_{\mathbf{x}_1} t_{\mathbf{x}_2}]_T$ and $[\mathbf{V}]_T = [t_{\mathbf{x}_1} t_{\mathbf{x}_2}\widehat{\mathbf{C}}_1 \odot \widehat{\mathbf{C}}_2]_T$ as $e(\mathsf{ct}_1, \mathsf{ct}_2)$.

2. Expands obtained matrix:
Let $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$. Expands $[\mathbf{V}]_T$ to obtain:
$[\mathbf{V}_i]_T = [t_{\mathbf{x}_1} t_{\mathbf{x}_2} \psi_{i,x_i}]_T$ for $i \in [\ell]$, $[\mathbf{V}_{i,b}]_T = [t_{\mathbf{x}_1} t_{\mathbf{x}_2} \psi'_{i,b}]_T$, where
$$\psi'_{i,b} = \begin{cases} \psi_{i,x_i} & \text{if } b = x_i \\ \mathbf{0} & \text{if } b = 1 - x_i \end{cases}, \text{ for } i \in [\ell+1, 2\ell], b \in \{0,1\}.$$

$$[\mathbf{V}_{i,b}]_T = [t_{\mathbf{x}_1} t_{\mathbf{x}_2} (\psi_{i,b} + \mathbf{C}_{i,b} \widehat{\mathbf{S}}_{i,b} + \widehat{\mathbf{E}}_{i,b})]_T \text{ for } i \in [2\ell+1, 3\ell], b \in \{0,1\},$$
$$[\mathbf{V}_{3\ell+1}]_T = [t_{\mathbf{x}_1} t_{\mathbf{x}_2} (\mathbf{C}_{3\ell+1} \widehat{\mathbf{S}}_{3\ell+1} + \widehat{\mathbf{E}} + \psi_{3\ell+1})]_T,$$
$$[\mathbf{v}_{3\ell+2}^\top]_T = [t_{\mathbf{x}_1} t_{\mathbf{x}_2} (\mathbf{C}_{3\ell+2} \widehat{\mathbf{s}}_{3\ell+2}^\top + \widehat{\mathbf{e}}_0^\top + \psi_{3\ell+2}^\top + \mathbf{m}^\top)]_T.$$

3. Recovers BGG^+ ciphertext components for third slot:
Let us denote $\mathbf{V}_{i,x_i}$ as $\mathbf{V}_i$ for $i \in [2\ell+1, 3\ell]$.
Computes $[\mathbf{t}_{\mathbf{x}_3} \mathbf{V}_i]_T = [t_{\mathbf{x}_1} t_{\mathbf{x}_2} \mathbf{t}_{\mathbf{x}_3} (\psi_{i,x_i} + \widehat{\mathbf{E}}_{i,x_i})]_T$ for $i \in [2\ell+1, 3\ell]$,
$[\mathbf{t}_{\mathbf{x}_3} \mathbf{V}_{3\ell+1}]_T = [t_{\mathbf{x}_1} t_{\mathbf{x}_2} \mathbf{t}_{\mathbf{x}_3} (\psi_{3\ell+1} + \widehat{\mathbf{E}})]_T$ and $[\mathbf{t}_{\mathbf{x}_3} \mathbf{v}_{3\ell+2}^\top]_T = [t_{\mathbf{x}_1} t_{\mathbf{x}_2} \mathbf{t}_{\mathbf{x}_3} (\psi_{3\ell+2}^\top + \mathbf{m}^\top + \widehat{\mathbf{e}}_0^\top)]_T$.
(because $\mathbf{t}_{\mathbf{x}_3} \mathbf{C}_{i,x_i} = \mathbf{0}$ for $i \in [2\ell+1, 3\ell]$, $\mathbf{t}_{\mathbf{x}_3} \mathbf{C}_{3\ell+1} = \mathbf{t}_{\mathbf{x}_3} \mathbf{C}_{3\ell+2} = \mathbf{0}$)

4. Computes function to be applied on BGG^+ ciphertexts:
Computes $\widehat{\mathbf{H}}_{F,\mathbf{x}} = \mathsf{EvalFX}(F, \mathbf{x}, \mathbf{B})$.

5. Performs BGG^+ decryption in the exponent:
   (a) Let us denote $\mathbf{V}_{i,x_i}$ as $\mathbf{V}_i$ for $i \in [\ell+1, 2\ell]$.
   (b) Computes $[\mathbf{t}_{\mathbf{x}_3} \mathbf{V}_i]_T$ for $i \in [2\ell]$.
   (c) Forms $[\mathbf{t}_{\mathbf{x}_3} \mathbf{V}_{\mathbf{x}}]_T = [\mathbf{t}_{\mathbf{x}_3} \mathbf{V}_1 \| \dots \| \mathbf{t}_{\mathbf{x}_3} \mathbf{V}_{3\ell}]_T$, $\mathbf{r} = (\mathbf{r}_1 \in \mathbb{Z}_q^m, \mathbf{r}_2 \in \mathbb{Z}_q^m)$.
   (d) Then computes
   $$[v']_T := \left[\left(\mathbf{t}_{\mathbf{x}_3} \mathbf{v}_{3\ell+2}^\top - \left(\mathbf{t}_{\mathbf{x}_3} \mathbf{V}_{3\ell+1} \mathbf{r}_1^\top + \mathbf{t}_{\mathbf{x}_3} \mathbf{V}_{\mathbf{x}} \widehat{\mathbf{H}}_{F,\mathbf{x}} \mathbf{r}_2^\top \right) \right) \right]_T$$

6. Recover exponent via brute force if $F(\mathbf{x}) = 0$:
After simplification, for $F(\mathbf{x}) = 0$, we get $v' = t_{\mathbf{x}_1} t_{\mathbf{x}_2} (\mathbf{t}_{\mathbf{x}_3} \mathbf{m}^\top + e')$, where e' is the combined error. Find $\eta \in [-B, B] \cup [-B - \lceil q/2 \rceil, B - \lceil q/K \rceil] \cup [-B + \lceil q/K \rceil, B + \lceil q/2 \rceil]$ such that $[v_0]_T^\eta = [v']_T$ by brute-force search. If there is no such η, output $\perp$. In the correctness, we show that η can be found in polynomial steps. To speed up the operation, one can employ the baby-step giant-step algorithm.

7. Output 0 if $\eta \in [-B, B]$ and 1, otherwise.

8.2 Parameters and Correctness

We choose the parameters for the 3-ABE scheme as follows:

$$m = n^{1.1} \log q, \qquad k = \theta(n\ell \log q), \qquad q = 2^{\Theta(\lambda)}$$
$$\tau_0 = n \log q \log m, \qquad \tau = m^{3.1} \ell \cdot 2^{O(d)} \qquad \tau'_0 = \omega(\sqrt{2n(\ell+1) \log q \log k})$$
$$\chi = \mathsf{SampZ}(3\sqrt{n}), \qquad \hat{\chi} = \mathsf{SampZ}(6\sqrt{n} m^2), \qquad B = \ell m^5 n^3 k \tau \tau_t \cdot 2^{O(d)}.$$

We can set τ_t to be arbitrary polynomial such that $\tau_t > \tau'_0$. The parameter n may be chosen as $n = \lambda^c$ for some constant $c > 1$.

Correctness To see correctness, we first make following observations:

1. Let $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$, then

$$\widehat{\mathbf{C}}_1 \odot \widehat{\mathbf{C}}_2 = (\{\psi_{i,x_i}\}_{i\in[\ell]}, \{\psi'_{i,b}\}_{i\in[\ell+1,2\ell],b\in\{0,1\}}, \{\mathbf{C}_{i,b}\widehat{\mathbf{S}}_{i,b} + \widehat{\mathbf{E}}_{i,b} + \psi_{i,b}\}_{\substack{i\in[2\ell+1,3\ell],\\ b\in\{0,1\}}},$$
$$\mathbf{C}_{3\ell+1}\widehat{\mathbf{S}}_{3\ell+1} + \widehat{\mathbf{E}} + \psi_{3\ell+1}, \mathbf{C}_{3\ell+2}\widehat{\mathbf{s}}^\top_{3\ell+2} + \widehat{\mathbf{e}}^\top_0 + \psi^\top_{3\ell+2} + \mathbf{m}^\top),$$

where

$$\psi'_{i,b} = \begin{cases} \psi_{i,x_i} & \text{if } b = x_i \\ \mathbf{0} & \text{if } b = 1 - x_i \end{cases}, \text{ for } i \in [\ell+1, 2\ell], b \in \{0,1\}.$$

Hence, on expanding $\mathbf{V}$, the decryptor obtains

$$\begin{aligned}
&[\mathbf{V}_i]_T = [t_{\mathbf{x}_1} t_{\mathbf{x}_2} \psi_{i,x_i}]_T \quad \text{for } i \in [2\ell], \\
&[\mathbf{V}_{i,b}]_T = [t_{\mathbf{x}_1} t_{\mathbf{x}_2} (\mathbf{C}_{i,b}\widehat{\mathbf{S}}_{i,b} + \widehat{\mathbf{E}}_{i,b} + \psi_{i,b})]_T, \quad \text{for } i \in [2\ell+1, 3\ell], b \in \{0,1\}, \\
&[\mathbf{V}_{3\ell+1}]_T = [t_{\mathbf{x}_1} t_{\mathbf{x}_2} (\mathbf{C}_{3\ell+1}\widehat{\mathbf{S}}_{3\ell+1} + \widehat{\mathbf{E}} + \psi_{3\ell+1})]_T, \\
&[\mathbf{v}^\top_{3\ell+2}]_T = [t_{\mathbf{x}_1} t_{\mathbf{x}_2} (\mathbf{C}_{3\ell+2}\widehat{\mathbf{s}}^\top_{3\ell+2} + \widehat{\mathbf{e}}^\top_0 + \psi^\top_{3\ell+2} + \mathbf{m}^\top)]
\end{aligned}$$

Here, recall that we represent $\mathbf{V}_{i,x_i}$ by $\mathbf{V}_i$, for $i \in [\ell+1, 2\ell]$.

2. Recovering $\{\psi_{2\ell+i,x_{3,i}}\}_{i\in[\ell]}$, $\psi_{3\ell+1}$ and $\psi_{3\ell+2}$:
For $i \in [\ell]$,

$$\begin{aligned}
\mathbf{t}_{\mathbf{x}_3}\mathbf{V}_{2\ell+i,x_{3,i}} &= t_{\mathbf{x}_1} t_{\mathbf{x}_2} (\mathbf{t}_{\mathbf{x}_3} (\psi_{2\ell+i,x_{3,i}} + \widehat{\mathbf{E}}_{2\ell+i,x_{3,i}}) + \mathbf{t}_{\mathbf{x}_3}\mathbf{C}_{2\ell+i,x_{3,i}}\widehat{\mathbf{S}}_{2\ell+i,x_{3,i}}) \\
&= t_{\mathbf{x}_1} t_{\mathbf{x}_2} \mathbf{t}_{\mathbf{x}_3} (\psi_{2\ell+i,x_{3,i}} + \widehat{\mathbf{E}}_{2\ell+i,x_{3,i}}) \text{ (because } \mathbf{t}_{\mathbf{x}_3}\mathbf{C}_{2\ell+i,x_{3,i}} = 0).
\end{aligned}$$

$$\begin{aligned}
\mathbf{t}_{\mathbf{x}_3}\mathbf{V}_{3\ell+1} &= t_{\mathbf{x}_1} t_{\mathbf{x}_2} (\mathbf{t}_{\mathbf{x}_3} (\psi_{3\ell+1} + \widehat{\mathbf{E}}) + \mathbf{t}_{\mathbf{x}_3}\mathbf{C}_{3\ell+1}\widehat{\mathbf{S}}_{3\ell+1}) \\
&= t_{\mathbf{x}_1} t_{\mathbf{x}_2} \mathbf{t}_{\mathbf{x}_3} (\psi_{3\ell+1} + \widehat{\mathbf{E}}) \text{ (because } \mathbf{t}_{\mathbf{x}_3}\mathbf{C}_{3\ell+1} = 0) \\
&= t_{\mathbf{x}_1} t_{\mathbf{x}_2} \mathbf{t}_{\mathbf{x}_3} (\mathbf{SA} + \mathbf{E} + \widehat{\mathbf{E}}).
\end{aligned}$$

$$\begin{aligned}
\mathbf{t}_{\mathbf{x}_3}\mathbf{v}^\top_{3\ell+2} &= t_{\mathbf{x}_1} t_{\mathbf{x}_2} (\mathbf{t}_{\mathbf{x}_3} (\psi^\top_{3\ell+2} + \mathbf{m}^\top + \widehat{\mathbf{e}}^\top_0) + \mathbf{t}_{\mathbf{x}_3}\mathbf{C}_{3\ell+2}\widehat{\mathbf{s}}^\top_{3\ell+2}) \\
&= t_{\mathbf{x}_1} t_{\mathbf{x}_2} \mathbf{t}_{\mathbf{x}_3} (\psi^\top_{3\ell+2} + \mathbf{m}^\top + \widehat{\mathbf{e}}^\top_0) \text{ (because } \mathbf{t}_{\mathbf{x}_3}\mathbf{C}_{3\ell+2} = 0) \\
&= t_{\mathbf{x}_1} t_{\mathbf{x}_2} \mathbf{t}_{\mathbf{x}_3} (\mathbf{Su}^\top + \mathbf{m}^\top + (\mathbf{e}^\top_0 + \widehat{\mathbf{e}}^\top_0)).
\end{aligned}$$

Representing $\mathbf{t}_{\mathbf{x}_3}\mathbf{V}_{i,x_i}$ by $\mathbf{t}_{\mathbf{x}_3}\mathbf{V}_i$ for $i \in [2\ell+1, 3\ell]$ gives us, for $i \in [2\ell+1, 3\ell]$,

$$\begin{aligned}
\mathbf{t}_{\mathbf{x}_3}\mathbf{V}_i &= \mathbf{t}_{\mathbf{x}_3}\mathbf{V}_{i,x_i} \\
&= t_{\mathbf{x}_1} t_{\mathbf{x}_2} \mathbf{t}_{\mathbf{x}_3} (\psi_{i,x_i} + \widehat{\mathbf{E}}_{i,x_i}) \\
&= t_{\mathbf{x}_1} t_{\mathbf{x}_2} \mathbf{t}_{\mathbf{x}_3} (\mathbf{S}(\mathbf{B}_i - x_i\mathbf{G}) + \mathbf{E}_{i,x_i} + \widehat{\mathbf{E}}_{i,x_i}).
\end{aligned}$$

3. Next, observe that:

$$\begin{aligned}
\mathbf{t}_{\mathbf{x}_3}\mathbf{V}_{\mathbf{x}} &= \mathbf{t}_{\mathbf{x}_3}(\mathbf{V}_1,\ldots,\mathbf{V}_{2\ell},\mathbf{V}_{2\ell+1},\ldots,\mathbf{V}_{3\ell}) \\
&= t_{\mathbf{x}_1}t_{\mathbf{x}_2}\mathbf{t}_{\mathbf{x}_3}(\mathbf{S}(\mathbf{B}_1 - x_1\mathbf{G}) + \mathbf{E}_{1,x_1},\ldots,\mathbf{S}(\mathbf{B}_{2\ell} - x_{2\ell}\mathbf{G}) + \mathbf{E}_{2\ell,x_{2\ell}}, \\
&\quad \mathbf{S}(\mathbf{B}_{2\ell+1} - x_{2\ell+1}\mathbf{G}) + \mathbf{E}_{2\ell+1,x_{2\ell+1}} + \widehat{\mathbf{E}}_{2\ell+1,x_{2\ell+1}},\ldots,\mathbf{S}(\mathbf{B}_{3\ell} - x_{3\ell}\mathbf{G}) + \mathbf{E}_{3\ell,x_{3\ell}} + \widehat{\mathbf{E}}_{3\ell,x_{3\ell}}) \\
&= t_{\mathbf{x}_1}t_{\mathbf{x}_2}\mathbf{t}_{\mathbf{x}_3}\mathbf{S}((\mathbf{B}_1,\ldots,\mathbf{B}_{3\ell}) - (x_1\mathbf{G},\ldots,x_{3\ell}\mathbf{G})) \\
&\quad + t_{\mathbf{x}_1}t_{\mathbf{x}_2}\mathbf{t}_{\mathbf{x}_3}((\mathbf{E}_{1,x_1},\ldots,\mathbf{E}_{3\ell,x_{3\ell}}) + (\mathbf{0}_{k\times 2\ell m},\widehat{\mathbf{E}}_{2\ell+1,x_{2\ell+1}},\ldots,\widehat{\mathbf{E}}_{3\ell,x_{3\ell}})) \\
&= t_{\mathbf{x}_1}t_{\mathbf{x}_2}\mathbf{t}_{\mathbf{x}_3}\mathbf{S}(\mathbf{B} - \mathbf{x}\otimes\mathbf{G}) + t_{\mathbf{x}_1}t_{\mathbf{x}_2}\mathbf{t}_{\mathbf{x}_3}(\mathbf{E}_{\mathbf{x}} + \widehat{\mathbf{E}}_{\mathbf{x}_3}), \\
&\quad (\text{ where } \mathbf{E}_{\mathbf{x}} = (\mathbf{E}_{1,x_1},\ldots,\mathbf{E}_{3\ell,x_{3\ell}}) \text{ and } \widehat{\mathbf{E}}_{\mathbf{x}_3} = (\mathbf{0}_{k\times 2\ell m},\widehat{\mathbf{E}}_{2\ell+1,x_{2\ell+1}},\ldots,\widehat{\mathbf{E}}_{3\ell,x_{3\ell}})).
\end{aligned}$$

4. Performing BGG^+ evaluation and decryption in the exponent yields:

$$\begin{aligned}
[v']_T &= [(\mathbf{t}_{\mathbf{x}_3}\mathbf{v}_{3\ell+2}^\top - (\mathbf{t}_{\mathbf{x}_3}\mathbf{V}_{3\ell+1}\mathbf{r}_1^\top + \mathbf{t}_{\mathbf{x}_3}\mathbf{V}_{\mathbf{x}}\widehat{\mathbf{H}}_{F,\mathbf{x}}\mathbf{r}_2^\top))]_T \\
&= [t_{\mathbf{x}_1}t_{\mathbf{x}_2}\mathbf{t}_{\mathbf{x}_3}(\mathbf{S}\mathbf{u}^\top + \mathbf{m}^\top + \mathbf{e}_0^\top + \widehat{\mathbf{e}}_0^\top) - t_{\mathbf{x}_1}t_{\mathbf{x}_2}\mathbf{t}_{\mathbf{x}_3}(\mathbf{S}\mathbf{A} + \mathbf{E} + \widehat{\mathbf{E}})\mathbf{r}_1^\top \\
&\quad - t_{\mathbf{x}_1}t_{\mathbf{x}_2}\mathbf{t}_{\mathbf{x}_3}(\mathbf{S}(\mathbf{B} - \mathbf{x}\otimes\mathbf{G}) + \mathbf{E}_{\mathbf{x}} + \widehat{\mathbf{E}}_{\mathbf{x}_3})\widehat{\mathbf{H}}_{F,\mathbf{x}}\mathbf{r}_2^\top]_T \\
&= [t_{\mathbf{x}_1}t_{\mathbf{x}_2}\mathbf{t}_{\mathbf{x}_3}(\mathbf{S}\mathbf{u}^\top + \mathbf{m}^\top - \mathbf{S}(\mathbf{A}\mathbf{r}_1^\top + (\mathbf{B}\mathbf{H}_F - F(\mathbf{x})\mathbf{G})\mathbf{r}_2^\top) \\
&\quad + t_{\mathbf{x}_1}t_{\mathbf{x}_2}\mathbf{t}_{\mathbf{x}_3}(\mathbf{e}_0^\top + \widehat{\mathbf{e}}_0^\top - (\mathbf{E} + \widehat{\mathbf{E}})\mathbf{r}_1^\top - (\mathbf{E}_{\mathbf{x}} + \widehat{\mathbf{E}}_{\mathbf{x}_3})\widehat{\mathbf{H}}_{F,\mathbf{x}}\mathbf{r}_2^\top)]_T \\
&\quad (\because (\mathbf{B} - \mathbf{x}\otimes\mathbf{G})\widehat{\mathbf{H}}_{F,\mathbf{x}} = \mathbf{B}\mathbf{H}_F - F(\mathbf{x})\mathbf{G} \text{ (Lemma 2.13)})
\end{aligned}$$

Replacing $\mathbf{B}\mathbf{H}_F$ by $\mathbf{B}_F$, $(\mathbf{r}_1,\mathbf{r}_2)$ by $\mathbf{r}$, $\mathbf{t}_{\mathbf{x}_3}(\mathbf{e}_0^\top + \widehat{\mathbf{e}}_0^\top - (\mathbf{E} + \widehat{\mathbf{E}})\mathbf{r}_1^\top - (\mathbf{E}_{\mathbf{x}} + \widehat{\mathbf{E}}_{\mathbf{x}_3})\widehat{\mathbf{H}}_{F,\mathbf{x}}\mathbf{r}_2^\top)$ by e' and for $F(\mathbf{x}) = 0$, we get:

$$\begin{aligned}
[v']_T &= [t_{\mathbf{x}_1}t_{\mathbf{x}_2}(\mathbf{t}_{\mathbf{x}_3}(\mathbf{S}\mathbf{u}^\top + \mathbf{m}^\top - \mathbf{S}(\mathbf{A}\|\mathbf{B}_F)\mathbf{r}^\top) + e')]_T \\
&= [t_{\mathbf{x}_1}t_{\mathbf{x}_2}(\mathbf{t}_{\mathbf{x}_3}(\mathbf{S}\mathbf{u}^\top + \mathbf{m}^\top - \mathbf{S}\mathbf{u}^\top) + e')]_T \quad (\because (\mathbf{A}\|\mathbf{B}_F)\mathbf{r}^\top = \mathbf{u}^\top) \\
&= [t_{\mathbf{x}_1}t_{\mathbf{x}_2}(\mathbf{t}_{\mathbf{x}_3}\mathbf{m}^\top + e')]_T = [v_0]_T^{(\mathbf{t}_{\mathbf{x}_3}\mathbf{m}^\top + e')}
\end{aligned}$$

Thus, by brute force search we get $\eta = \mathbf{t}_{\mathbf{x}_3}\mathbf{m}^\top + e'$.

5. Bounding error e':
Recall that we set $\chi = \mathsf{SampZ}(3\sqrt{n})$, $\hat{\chi} = \mathsf{SampZ}(6\sqrt{n}m^2)$. By the definition of SampZ, we have $\|\mathbf{e}_0\|_\infty, \|\widehat{\mathbf{e}}_0\|_\infty \le 3n$, $\|\mathbf{E}\|_\infty, \|\widehat{\mathbf{E}}\|_\infty \le 3n$. and $\|\mathbf{E}_{i,b}\|_\infty, \|\widehat{\mathbf{E}}_{i,b}\|_\infty \le 6nm^2$ for $i \in [3\ell]$ and $b \in \{0,1\}$, $\|\mathbf{r}\|_\infty \le \sqrt{n}\tau$, $\|\mathbf{t}_{\mathbf{x}_3}\|_\infty \le \sqrt{n}\tau_t$, and $\|\widehat{\mathbf{H}}_{F,\mathbf{x}}\|_\infty \le m\cdot 2^{O(d)}$, where the last inequality follows from Lemma 2.13. Thus, we have

$$\begin{aligned}
e' &= \mathbf{t}_{\mathbf{x}_3}(\mathbf{e}_0^\top + \widehat{\mathbf{e}}_0^\top - (\mathbf{E} + \widehat{\mathbf{E}})\mathbf{r}_1^\top - (\mathbf{E}_{\mathbf{x}} + \widehat{\mathbf{E}}_{\mathbf{x}_3})\widehat{\mathbf{H}}_{F,\mathbf{x}}\mathbf{r}_2^\top) \\
&\le k\sqrt{n}\tau_t(6n + 6n^{1.5}m\tau + 24\ell m^5 n^{1.5}\tau\cdot 2^{O(d)}) \\
&= O(\ell m^5 n^2 k\tau\tau_t\cdot 2^{O(d)}) \le B
\end{aligned}$$

by our choice of B.

6. Bounding $\mathbf{t}_{\mathbf{x}_3}\mathbf{m}^\top$:
When message bit $b = 0$, then $\mathbf{t}_{\mathbf{x}_3}\mathbf{m}^\top$=0. For $b = 1$, $\lceil q/K\rceil \le |\mathbf{t}_{\mathbf{x}_3}\mathbf{m}^\top| \le \tau_t\sqrt{n}k\cdot\lceil q/K\rceil$ (where $K = 2\tau_t\sqrt{n}\cdot k$), unless $\mathbf{t}_{\mathbf{x}_3}\mathbf{m}^\top = 0$. Thus, for $b = 0$, $\eta \in [-B, B]$ and for ($b = 1$, and $B < \lceil\frac{q}{2K}\rceil$), $\eta \in [-B - \lceil q/2\rceil, -\lceil q/K\rceil + B] \cup [\lceil q/K\rceil - B, B + \lceil q/2\rceil]$, unless

$\mathbf{t}_{\mathbf{x}_3}\mathbf{m}^\top = 0$. Observe that $|\mathbf{t}_{\mathbf{x}_3}\mathbf{m}^\top| \in \{0, \lceil \frac{q}{K} \rceil, 2 \cdot \lceil \frac{q}{K} \rceil, \ldots, \tau_t\sqrt{n}k \cdot \lceil \frac{q}{K} \rceil\}$. Thus, η can take only $2B + 4B\tau_t\sqrt{n}k$ different values. Since, B, k, τ_t are polynomially bounded, η can be found by brute force search in polynomial steps.

The probability that $\mathbf{t}_{\mathbf{x}_3}\mathbf{m}^\top = 0$ is non-negligible but bounded away from 1 and hence this may be amplified as discussed below.

Amplifying Correctness. Above, we set $\mathbf{m} = \mu \cdot (q/K)(1, \ldots, 1)$. Note that for correctness, we require that $\mathbf{t}_{\mathbf{x}_3}\mathbf{m}^\top \neq 0$. However, since we are constrained to sample $\mathbf{t}_{\mathbf{x}_3}$ polynomially bounded (since the message must be recovered from the exponent), the probability that $\mathbf{t}_{\mathbf{x}_3}\mathbf{m}^\top = 0$ is non-negligible, leading to error in correctness. A simple method to amplify correctness is to simply run the scheme in parallel λ times, and output 0 only if all instances output 0. This (standard) trick allows to make the correctness error exponentially small, although with the disadvantage of reducing efficiency. We note that we can do better by replacing the vector $\mathbf{u}$ with λ vectors $\mathbf{u}_1, \ldots, \mathbf{u}_\lambda$ and providing λ secret keys $\mathbf{r}_1, \ldots, \mathbf{r}_\lambda$ corresponding to each one. Similarly, we can provide λ encodings of the message bit, decrypt each one of them and output 0 only if all instances output 0. However, we choose not to clutter the (already complex) formal description with this added complication for ease of exposition.

8.3 Discussion of Security

Compared to [BV22], we require slightly different security requirements for the encodings (even though neither works formalize this). First, we need the encoding to retain security even if some of the masks are stripped off, as long as only one encoding for the same position is revealed. We expect this to be secure since these stripped off encodings are fresh BGG^+ encodings and should be secure by BGG^+ security. Second, in their case, only a single BGG^+ secret key is generated per each instance of BGG^+, which is sampled afresh for each ciphertext, while in our case, we use the same BGG^+ instance throughout the system and generate multiple secret keys for it. On the other hand, in our case, the encodings all live in the exponent, unlike their case, where they live "downstairs". Hence, the attacker gets restricted to only linear attacks by GGM whereas the attacker has more freedom in their construction.

9 Two-Input ABE for Polynomial Circuits using BV22

In this section, we construct candidate two input ABE scheme using the structure of [BV22] scheme. Unlike other schemes in the paper, the construction below does not employ pairings and thus is expected to be post-quantum secure. Besides, it can support polynomial-size circuits with any depth. Formally, it supports attribute space $A_\lambda = \{0,1\}^{\ell(\lambda)}$ and any circuit class $\mathcal{F} = \{\mathcal{F}_\lambda\}_\lambda$ that is subclass of $\{\mathcal{C}_{2\ell(\lambda),d(\lambda)}\}_\lambda$ with arbitrary $\ell(\lambda) \leq \mathrm{poly}(\lambda)$ and $d(\lambda) \leq \mathrm{poly}(\lambda)$, where $\mathcal{C}_{2\ell(\lambda),d(\lambda)}$ is a set of circuits with input length $2\ell(\lambda)$ and depth at most $d(\lambda)$.

9.1 Construction

The construction is defined as follows:

$\mathsf{Setup}(1^\lambda)$: On input 1^λ, the setup algorithm defines the parameters $n = n(\lambda)$, $m = m(\lambda)$, $k = k(\lambda)$, noise distribution χ, $\hat{\chi}$ over $\mathbb{Z}$, $\tau_0 = \tau_0(\lambda)$, $\tau = \tau(\lambda)$, $\tau_0' = \tau_0'(\lambda)$, $\tau_t = \tau_t(\lambda)$ and $B = B(\lambda)$ as specified in Sec. 8.2. Let ℓ be the length of the attributes and d be the maximum depth of circuits. Then the setup algorithm does the following.

1. Samples BGG^+ master secret and public keys:
   (a) Samples $(\mathbf{A}, \mathbf{A}_{\tau_0}^{-1}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$ such that $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$.
   (b) Samples random matrices $\mathbf{B} = (\mathbf{B}_1, \ldots, \mathbf{B}_{2\ell}) \leftarrow (\mathbb{Z}_q^{n\times m})^{2\ell}$ and a random vector $\mathbf{u} \leftarrow \mathbb{Z}_q^n$.
2. Samples $(\mathbf{C}, \mathbf{C}_{\tau_0'}^{-1}) \leftarrow \mathsf{TrapGen}(1^{2n(\ell+1)}, 1^k, q)$, where
$$\mathbf{C}^\top = (\mathbf{C}_{\ell+1,0} \| \mathbf{C}_{\ell+1,1} \| \ldots \| \mathbf{C}_{2\ell,0} \| \mathbf{C}_{2\ell,1} \| \mathbf{C}_{2\ell+1} \| \mathbf{C}_{2\ell+2}) \in (\mathbb{Z}_q^{k\times n})^{2\ell+2}.$$
3. Outputs
$$\mathsf{pp} = (\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{u}),\ \mathsf{msk} = (\mathbf{A}_{\tau_0}^{-1}, \mathbf{C}_{\tau_0'}^{-1}).$$

$\mathsf{Enc}_1(\mathsf{pp}, \mathsf{msk}, \mathbf{x}_1, \mu)$: On input the public parameters pp, master secret msk, attribute vector $\mathbf{x}_1$ and message bit μ, encryption for slot 1 does the following:

1. Sets $\mathbf{m} = \lceil \frac{q}{K} \rceil \mu(1, \ldots, 1) \in \mathbb{Z}_q^k$. We define $K = 2\tau_t\sqrt{n}k$.
2. Samples a random secret matrix $\mathbf{S} \leftarrow \mathbb{Z}_q^{k\times n}$ and error vectors/matrices as $\mathbf{e}_0 \leftarrow \chi^k$, $\mathbf{E} \leftarrow \chi^{k\times m}$, $\mathbf{E}_{i,x_{1,i}} \leftarrow \hat{\chi}^{k\times m}$ for $i \in [\ell]$, and $\mathbf{E}_{i,b} \leftarrow \hat{\chi}^{k\times m}$ for $i \in [\ell+1, 2\ell]$ and $b \in \{0,1\}$.
3. Computes
$$\psi_{i,x_{1,i}} = \mathbf{S}(\mathbf{B}_i - x_{1,i}\mathbf{G}) + \mathbf{E}_{i,x_{1,i}} \text{ for } i \in [\ell],$$
$$\psi_{i,b} = \mathbf{S}(\mathbf{B}_i - b\mathbf{G}) + \mathbf{E}_{i,b} \text{ for } i \in [\ell+1, 2\ell], b \in \{0,1\},$$
$$\psi_{2\ell+1} = \mathbf{SA} + \mathbf{E},\ \psi_{2\ell+2}^\top = \mathbf{Su}^\top + \mathbf{e}_0^\top.$$
4. For $i \in [\ell+1, 2\ell], b \in \{0,1\}$, samples
$$\widehat{\mathbf{S}}_{i,b} \leftarrow \mathbb{Z}_q^{n\times m},\ \widehat{\mathbf{S}}_{2\ell+1} \leftarrow \mathbb{Z}_q^{n\times m},\ \widehat{\mathbf{s}}_{2\ell+2} \leftarrow \mathbb{Z}_q^n,$$
$$\widehat{\mathbf{E}}_{i,b} \leftarrow \hat{\chi}^{k\times m},\ \widehat{\mathbf{E}} \leftarrow \chi^{k\times m},\ \widehat{\mathbf{e}}_0 \leftarrow \chi^k.$$
5. Computes $\widehat{\psi}_{i,b} = \mathbf{C}_{i,b}\widehat{\mathbf{S}}_{i,b} + \widehat{\mathbf{E}}_{i,b} + \psi_{i,b}$ for $i \in [\ell+1, 2\ell], b \in \{0,1\}$,
$$\widehat{\psi}_{2\ell+1} = \mathbf{C}_{2\ell+1}\widehat{\mathbf{S}}_{2\ell+1} + \widehat{\mathbf{E}} + \psi_{2\ell+1},\ \ \widehat{\psi}_{2\ell+2}^\top = \mathbf{C}_{2\ell+2}\widehat{\mathbf{s}}_{2\ell+2}^\top + \widehat{\mathbf{e}}_0^\top + \psi_{2\ell+2}^\top + \mathbf{m}^\top.$$
6. Outputs $\mathsf{ct}_1 = (\{\psi_{i,x_{1,i}}\}_{i\in[\ell]}, \{\widehat{\psi}_{i,b}\}_{i\in[\ell+1,2\ell],b\in\{0,1\}}, \widehat{\psi}_{2\ell+1}, \widehat{\psi}_{2\ell+2})$.

$\mathsf{Enc}_2(\mathsf{pp}, \mathsf{msk}, \mathbf{x}_2)$: On input the public parameters pp, master secret key msk and an attribute vector $\mathbf{x}_2$, encryption for slot 2 is defined as follows:

1. Computes $[(\mathbf{C}_{2\ell+1} \| \mathbf{C}_{2\ell+2} \| \mathbf{C}_{\ell+1,x_{2,1}} \| \ldots \| \mathbf{C}_{2\ell,x_{2,\ell}})^\top]_{\tau_t}^{-1}$ from $\mathbf{C}_{\tau_0'}^{-1}$ and samples a short vector $\mathbf{t}_{\mathbf{x}_2}$ such that
$$\mathbf{t}_{\mathbf{x}_2}(\mathbf{C}_{2\ell+1} \| \mathbf{C}_{2\ell+2} \| \mathbf{C}_{\ell+1,x_{2,1}} \| \ldots \| \mathbf{C}_{2\ell,x_{2,\ell}}) = \mathbf{0} \mod q \text{ as}$$
$$\mathbf{t}_{\mathbf{x}_2}^\top \leftarrow [(\mathbf{C}_{2\ell+1} \| \mathbf{C}_{2\ell+2} \| \mathbf{C}_{\ell+1,x_{2,1}} \| \ldots \| \mathbf{C}_{2\ell,x_{2,\ell}})^\top]_{\tau_t}^{-1}(\mathbf{0}).$$
2. Returns $\mathsf{ct}_2 = \mathbf{t}_{\mathbf{x}_2}$.

$\mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, F)$: On input the public parameters pp, master secret key msk and a function F, the keygen algorithm does the following.

1. Generates BGG^+ function key:
   (a) Computes $\mathbf{H}_F = \mathsf{EvalF}(\mathbf{B}, F)$ and $\mathbf{B}_F = \mathbf{B}\mathbf{H}_F$.
   (b) Computes $[\mathbf{A}\|\mathbf{B}_F]_\tau^{-1}$ from $\mathbf{A}_{\tau_0}^{-1}$ and samples $\mathbf{r} \in \mathbb{Z}^{2m}$ as $\mathbf{r}^\top \leftarrow [\mathbf{A}\|\mathbf{B}_F]_\tau^{-1}(\mathbf{u}^\top)$.
2. Returns $\mathsf{sk}_F = \mathbf{r}$.

$\mathsf{Dec}(\mathsf{pp}, \mathsf{sk}_F, \mathsf{ct}_1, \mathsf{ct}_2)$: On input the public parameters pp, key $\mathsf{sk}_F = \mathbf{r}$, and slot 1 and slot 2 ciphertexts ct_1, ct_2, the decryption algorithm does the following.

1. Parses the public parameters pp as

$$(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{u})$$

   and the ciphertexts for slot 1 and slot 2 as

$$\mathsf{ct}_1 = (\{\psi_{i,x_{1,i}}\}_{i\in[\ell]}, \{\widehat{\psi}_{i,b}\}_{i\in[\ell+1,2\ell],b\in\{0,1\}}, \widehat{\psi}_{2\ell+1}, \widehat{\psi}_{2\ell+2}),\ \mathsf{ct}_2 = \mathbf{t}_{\mathbf{x}_2}.$$

2. Computes

$$\mathbf{t}_{\mathbf{x}_2}\mathbf{V} = \mathbf{t}_{\mathbf{x}_2}(\psi_{1,x_{1,1}}\| \ldots \|\psi_{\ell,x_{1,\ell}}\|\widehat{\psi}_{\ell+1,x_{2,1}}\| \ldots \|\widehat{\psi}_{2\ell,x_{2,\ell}}\|\widehat{\psi}_{2\ell+1}\|\widehat{\psi}_{2\ell+2}^\top).$$

3. Expands $\mathbf{t}_{\mathbf{x}_2}\mathbf{V}$ to obtain

$$\mathbf{t}_{\mathbf{x}_2}\mathbf{V}_{i,x_{1,i}} = \mathbf{t}_{\mathbf{x}_2}\psi_{i,x_{1,i}} \ \text{ for } i \in [\ell], \qquad \mathbf{t}_{\mathbf{x}_2}\mathbf{V}_{\ell+i,x_{2,i}} = \mathbf{t}_{\mathbf{x}_2}\widehat{\psi}_{\ell+i,x_{2,i}} \ \text{ for } i \in [\ell],$$
$$\mathbf{t}_{\mathbf{x}_2}\mathbf{V}_{2\ell+1} = \mathbf{t}_{\mathbf{x}_2}\widehat{\psi}_{2\ell+1}, \qquad \mathbf{t}_{\mathbf{x}_2}\mathbf{v}_{2\ell+2}^\top = \mathbf{t}_{\mathbf{x}_2}\widehat{\psi}_{2\ell+2}^\top.$$

4. Forms $\mathbf{r} = (\mathbf{r}_1 \in \mathbb{Z}_q^m, \mathbf{r}_2 \in \mathbb{Z}_q^m)$ and $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$.
   Let,
$$(\mathbf{V}_{1,x_1}\| \ldots \|\mathbf{V}_{2\ell,x_{2\ell}}) = \mathbf{V}_{\mathbf{x}}.$$
5. Computes $\widehat{\mathbf{H}}_{F,\mathbf{x}} = \mathsf{EvalFX}(F, \mathbf{x}, \mathbf{B})$.
6. Computes
$$v = (\mathbf{t}_{\mathbf{x}_2}\mathbf{v}_{2\ell+2}^\top - \big(\mathbf{t}_{\mathbf{x}_2}\mathbf{V}_{2\ell+1}\mathbf{r}_1^\top + \mathbf{t}_{\mathbf{x}_2}\mathbf{V}_{\mathbf{x}}\widehat{\mathbf{H}}_{F,\mathbf{x}}\mathbf{r}_2^\top\big)).$$
7. Outputs 0 if $v \in [-B, B]$ and 1, otherwise.

Correctness: To see correctness, we first make following observations:

1. Let $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$.

   On expanding $\mathbf{t}_{\mathbf{x}_2}\mathbf{V}$, the decryptor obtains, for $i \in [\ell]$,

$$
\begin{aligned}
\mathbf{t}_{\mathbf{x}_2}\mathbf{V}_{i,x_{1,i}} &= \mathbf{t}_{\mathbf{x}_2}\psi_{i,x_{1,i}} \\
&= \mathbf{t}_{\mathbf{x}_2}\mathbf{S}(\mathbf{B}_i - x_{1,i}\mathbf{G}) + \mathbf{t}_{\mathbf{x}_2}\mathbf{E}_{i,x_{1,i}}.
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{t}_{\mathbf{x}_2}\mathbf{V}_{\ell+i,x_{2,i}} &= \mathbf{t}_{\mathbf{x}_2}\widehat{\psi}_{\ell+i,x_{2,i}} \\
&= \mathbf{t}_{\mathbf{x}_2}(\mathbf{C}_{\ell+i,\mathbf{x}_{2,i}}\widehat{\mathbf{S}}_{\ell+i,x_{2,i}} + \widehat{\mathbf{E}}_{\ell+i,x_{2,i}} + \psi_{\ell+i,\mathbf{x}_{2,i}}) \\
&= \mathbf{t}_{\mathbf{x}_2}\mathbf{S}(\mathbf{B}_{\ell+i} - x_{2,i}\mathbf{G}) + \mathbf{t}_{\mathbf{x}_2}(\mathbf{E}_{\ell+i,x_{2,i}} + \widehat{\mathbf{E}}_{\ell+i,x_{2,i}}).
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{t}_{\mathbf{x}_2}\mathbf{V}_{2\ell+1} &= \mathbf{t}_{\mathbf{x}_2}\widehat{\psi}_{2\ell+1} \\
&= \mathbf{t}_{\mathbf{x}_2}(\mathbf{C}_{2\ell+1}\widehat{\mathbf{S}}_{2\ell+1} + \widehat{\mathbf{E}} + \psi_{2\ell+1}) \\
&= \mathbf{t}_{\mathbf{x}_2}\mathbf{S}\mathbf{A} + \mathbf{t}_{\mathbf{x}_2}(\mathbf{E} + \widehat{\mathbf{E}}).
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{t}_{\mathbf{x}_2}\mathbf{v}_{2\ell+2}^\top &= \mathbf{t}_{\mathbf{x}_2}\widehat{\psi}_{2\ell+2}^\top \\
&= \mathbf{t}_{\mathbf{x}_2}(\mathbf{C}_{2\ell+2}\widehat{\mathbf{s}}_{2\ell+2}^\top + \widehat{\mathbf{e}}_0^\top + \psi_{2\ell+2}^\top + \mathbf{m}^\top) \\
&= \mathbf{t}_{\mathbf{x}_2}(\mathbf{S}\mathbf{u}^\top + \mathbf{m}^\top) + \mathbf{t}_{\mathbf{x}_2}(\mathbf{e}_0^\top + \widehat{\mathbf{e}}_0^\top).
\end{aligned}
$$

2. Next, observe that:

$$
\begin{aligned}
\mathbf{t}_{\mathbf{x}_2}\mathbf{V}_{\mathbf{x}} &= \mathbf{t}_{\mathbf{x}_2}(\mathbf{V}_1, \ldots, \mathbf{V}_\ell, \mathbf{V}_{\ell+1}, \ldots, \mathbf{V}_{2\ell}) \\
&= \mathbf{t}_{\mathbf{x}_2}\mathbf{S}(\mathbf{B} - \mathbf{x} \otimes \mathbf{G}) + \mathbf{t}_{\mathbf{x}_2}(\mathbf{E}_{\mathbf{x}} + \widehat{\mathbf{E}}_{\mathbf{x}_2}), \\
&\quad \text{where } \mathbf{E}_{\mathbf{x}} = (\mathbf{E}_{1,x_1}, \ldots, \mathbf{E}_{2\ell,x_{2\ell}}) \text{ and } \widehat{\mathbf{E}}_{\mathbf{x}_2} = (\mathbf{0}_{k\times\ell m}, \widehat{\mathbf{E}}_{\ell+1,x_{\ell+1}}, \ldots, \widehat{\mathbf{E}}_{2\ell,x_{2\ell}}).
\end{aligned}
$$

3. Finally, we get,

$$
\begin{aligned}
v &= \mathbf{t}_{\mathbf{x}_2}\mathbf{v}_{2\ell+2}^\top - (\mathbf{t}_{\mathbf{x}_2}\mathbf{V}_{2\ell+1}\mathbf{r}_1^\top + \mathbf{t}_{\mathbf{x}_2}\mathbf{V}_{\mathbf{x}}\widehat{\mathbf{H}}_{F,\mathbf{x}}\mathbf{r}_2^\top) \\
&= \mathbf{t}_{\mathbf{x}_2}(\mathbf{S}\mathbf{u}^\top + \mathbf{m}^\top - \mathbf{S}(\mathbf{A}\mathbf{r}_1^\top + (\mathbf{B}\mathbf{H}_F - F(\mathbf{x})\mathbf{G})\mathbf{r}_2^\top)) \\
&+ \mathbf{t}_{\mathbf{x}_2}(\mathbf{e}_0^\top + \widehat{\mathbf{e}}_0^\top - (\mathbf{E} + \widehat{\mathbf{E}})\mathbf{r}_1^\top - (\mathbf{E}_{\mathbf{x}} + \widehat{\mathbf{E}}_{\mathbf{x}_2})\widehat{\mathbf{H}}_{F,\mathbf{x}}\mathbf{r}_2^\top) \\
&= \mathbf{t}_{\mathbf{x}_2}(\mathbf{S}\mathbf{u}^\top + \mathbf{m}^\top - \mathbf{S}(\mathbf{A}\|\mathbf{B}_F)\mathbf{r}^\top) + e' \\
&= \mathbf{t}_{\mathbf{x}_2}(\mathbf{S}\mathbf{u}^\top + \mathbf{m}^\top - \mathbf{S}\mathbf{u}^\top) + e' \\
&= \mathbf{t}_{\mathbf{x}_2}\mathbf{m}^\top + e' \\
&\text{where } e' = \mathbf{t}_{\mathbf{x}_2}(\mathbf{e}_0^\top + \widehat{\mathbf{e}}_0^\top - (\mathbf{E} + \widehat{\mathbf{E}})\mathbf{r}_1^\top - (\mathbf{E}_{\mathbf{x}} + \widehat{\mathbf{E}}_{\mathbf{x}_2})\widehat{\mathbf{H}}_{F,\mathbf{x}}\mathbf{r}_2^\top).
\end{aligned}
$$

4. As discussed in section 8.2, the error e' is bounded by $O(\ell m^5 n^2 k\tau\tau_t 2^{O(d)})$, and $\mathbf{t}_{\mathbf{x}}\mathbf{m}^\top = 0$ when $b = 0$ and $\mathbf{t}_{\mathbf{x}}\mathbf{m}^\top \leq \tau_t\sqrt{n}k \cdot \lceil \frac{q}{2\tau_t\sqrt{nk}} \rceil$ for $b = 1$. Thus, for $b = 0$, $v \in [-B, B]$ and for ($b = 1$, and $B < \frac{q}{4k\sqrt{n}\tau_t}$), $v \notin [-B, B]$ unless $\mathbf{t}_{\mathbf{x}}\mathbf{m}^\top = 0$. The probability that $\mathbf{t}_{\mathbf{x}}\mathbf{m}^\top = 0$ is non-negligible but bounded away from 1 and hence this may be amplified as discussed in section 8.2.

Acknowledgement This work was partly supported by the DST "Swarnajayanti" fellowship, Cybersecurity Center of Excellence, IIT Madras, National Blockchain Project, European Union Horizon 2020 Research and Innovation Program Grant 780701, BPIFrance in the context of the national project RISQ (P141580), and the ANR AMIRAL project (ANR-21-ASTR-0016). The third author was partially supported by JST AIP Acceleration Research JPMJCR22U5 and JSPS KAKENHI Grant Number 19H01109, Japan.

References

[ABB10a] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In *EUROCRYPT*, 2010.

[ABB10b] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE. In *CRYPTO*, 2010.

[ABG19] Michel Abdalla, Fabrice Benhamouda, and Romain Gay. From single-input to multi-client inner-product functional encryption. In *ASIACRYPT*, 2019.

[ABKW19] Michel Abdalla, Fabrice Benhamouda, Markulf Kohlweiss, and Hendrik Waldner. Decentralizing inner-product functional encryption. In *PKC*, 2019.

[ACF+18] Michel Abdalla, Dario Catalano, Dario Fiore, Romain Gay, and Bogdan Ursu. Multi-input functional encryption for inner products: Function-hiding realizations and constructions without pairings. In *CRYPTO*, 2018.

[AFV11] Shweta Agrawal, David Mandell Freeman, and Vinod Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In *Asiacrypt*, 2011.

[Agr19] Shweta Agrawal. Indistinguishability obfuscation without multilinear maps: New techniques for bootstrapping and instantiation. In *Eurocrypt*, 2019.

[AGRW17] Michel Abdalla, Romain Gay, Mariana Raykova, and Hoeteck Wee. Multi-input inner-product functional encryption from pairings. In *EUROCRYPT*, 2017.

[AGT21] Shweta Agrawal, Rishab Goyal, and Junichi Tomida. Multi-input quadratic functional encryption from pairings. In *CRYPTO*, 2021.

[AJ15] Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In *CRYPTO*, 2015.

[AJL+19] Prabhanjan Ananth, Aayush Jain, Huijia Lin, Christian Matt, and Amit Sahai. Indistinguishability obfuscation without multilinear maps: iO from LWE, bilinear maps, and weak pseudorandomness. In *CRYPTO*, 2019.

[Att14] Nuttapong Attrapadung. Dual system encryption via doubly selective security: Framework, fully secure functional encryption for regular languages, and more. In *Eurocrypt*, 2014.

[AWY20] Shweta Agrawal, Daniel Wichs, and Shota Yamada. Optimal broadcast encryption from lwe and pairings in the standard model. In *TCC*, 2020.

[AY20] Shweta Agrawal and Shota Yamada. Optimal broadcast encryption from pairings and lwe. In *EUROCRYPT*, 2020.

[BCFG17] Carmen Elisabetta Zaira Baltico, Dario Catalano, Dario Fiore, and Romain Gay. Practical functional encryption for quadratic functions with applications to predicate encryption. In *CRYPTO*, 2017.

[BFF+14] Gilles Barthe, Edvard Fagerholm, Dario Fiore, John C. Mitchell, Andre Scedrov, and Benedikt Schmidt. Automated analysis of cryptographic assumptions in generic group models. In *CRYPTO*, 2014.

[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 *EUROCRYPT*, 2014.

[BGI+01] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. In *CRYPTO*, 2001.

[BJK15] Allison Bishop, Abhishek Jain, and Lucas Kowalczyk. Function-hiding inner product encryption. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT*, 2015.

[BJK+18] Zvika Brakerski, Aayush Jain, Ilan Komargodski, Alain Passelègue, and Daniel Wichs. Non-trivial witness encryption and null-io from standard assumptions. In *SCN*, 2018.

[BLP+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In *STOC*, 2013.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *TCC*, 2011.

[BV15] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. *FOCS*, 2015.

[BV16] Zvika Brakerski and Vinod Vaikuntanathan. Circuit-abe from lwe: Unbounded attributes and semi-adaptive security. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO*, 2016.

[BV22] Zvika Brakerski and Vinod Vaikuntanathan. Lattice-inspired broadcast encryption and succinct ciphertext policy abe. In *ITCS*, 2022.

[BW07] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In *TCC*, 2007.

[BW19] Ward Beullens and Hoeteck Wee. Obfuscating simple functionalities from knowledge assumptions. In *IACR International Workshop on Public Key Cryptography*, 2019.

[CDG+18] Jérémy Chotard, Edouard Dufour Sans, Romain Gay, Duong Hieu Phan, and David Pointcheval. Decentralized multi-client functional encryption for inner product. In *ASIACRYPT*, 2018.

[CHKP10] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In *EUROCRYPT*, 2010.

[CM15] Michael Clear and Ciaran McGoldrick. Multi-identity and multi-key leveled FHE from learning with errors. In *CRYPTO*, 2015.

[CW14] Jie Chen and Hoeteck Wee. Semi-adaptive attribute-based encryption and improved delegation for boolean formula. In *SCN*, 2014.

[DDM16] Pratish Datta, Ratna Dutta, and Sourav Mukhopadhyay. Functional encryption for inner product with full function privacy. In *PKC*, 2016.

[DOT18] Pratish Datta, Tatsuaki Okamoto, and Junichi Tomida. Full-hiding (unbounded) multi-input inner product functional encryption from the k-Linear assumption. In *PKC*, 2018.

[GGG+14] Shafi Goldwasser, S. Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. In *EUROCRYPT*, 2014.

[GJLS21] Romain Gay, Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from simple-to-state hard problems: New assumptions, new techniques, and simplification. In *EUROCRYPT*, 2021.

[GKW17] Rishab Goyal, Venkata Koppula, and Brent Waters. Lockable obfuscation. In *FOCS*, 2017.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM CCS*, 2006.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, 2008.

[GV15] Sergey Gorbunov and Dhinakaran Vinayagamurthy. Riding on asymmetry: Efficient ABE for branching programs. In *ASIACRYPT*, 2015.

[GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute based encryption for circuits. In *STOC*, 2013.

[GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from lwe. In *Crypto*, 2015.

[JLMS19] Aayush Jain, Huijia Lin, Christian Matt, and Amit Sahai. How to leverage hardness of constant-degree expanding polynomials over r to build io. In *EUROCRYPT*, 2019.

[JLS21] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In *STOC*, 2021.

[JLS22] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from lpn over large fields, dlin, and constant depth prgs. In *EUROCRYPT*, 2022.

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT*, 2008.

[LATV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *STOC*, 2012.

[Lin16] Huijia Lin. Indistinguishability obfuscation from constant-degree graded encoding schemes. In *EUROCRYPT*, 2016.

[Lin17] Huijia Lin. Indistinguishability obfuscation from sxdh on 5-linear maps and locality-5 PRGs. In *Crypto*, 2017.

[LOS+10] Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *EUROCRYPT*, 2010.

[LT19] Benoît Libert and Radu Titiu. Multi-client functional encryption for linear functions in the standard model from LWE. In *ASIACRYPT*, 2019.

[LV16] Huijia Lin and Vinod Vaikuntanathan. Indistinguishability obfuscation from ddh-like assumptions on constant-degree graded encodings. In *FOCS*, 2016.

[LW11] Allison B. Lewko and Brent Waters. Unbounded HIBE and attribute-based encryption. In *Eurocrypt*, pages 547–567, 2011.

[LW12] Allison B. Lewko and Brent Waters. New proof methods for attribute-based encryption: Achieving full security through selective techniques. In *Crypto*, 2012.

[Mau05] Ueli Maurer. Abstract models of computation in cryptography. In *IMACC*, 2005.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *Eurocrypt*, 2012.

[MW16] Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key fhe. In *EUROCRYPT*, 2016.

[OT10] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In *CRYPTO*, pages 191–208, 2010.

[OT12] Tatsuaki Okamoto and Katsuyuki Takashima. Adaptively attribute-hiding (hierarchical) inner product encryption. In *EUROCRYPT*, 2012.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J.ACM*, 56(6), 2009.

[SBC+07] Elaine Shi, John Bethencourt, TH Hubert Chan, Dawn Song, and Adrian Perrig. Multi-dimensional range query over encrypted data. In *SP*, 2007.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In *Eurocrypt*, 1997.

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, 2005.

[Tom19] Junichi Tomida. Tightly secure inner product functional encryption: Multi-input and function-hiding constructions. In *ASIACRYPT*, 2019.

[Tsa19] Rotem Tsabary. Fully secure attribute-based encryption for t-CNF from LWE. In *CRYPTO*, 2019.

[Wat12] Brent Waters. Functional encryption for regular languages. In *Crypto*, 2012.

[Wee14] Hoeteck Wee. Dual system encryption via predicate encodings. In *TCC*, 2014.

[WZ17] Daniel Wichs and Giorgos Zirdelis. Obfuscating compute-and-compare programs under LWE. In *FOCS*, 2017.