Multi-Input Attribute Based Encryption and Predicate Encryption

Shweta Agrawal[1], Anshu Yadav[1], and Shota Yamada[2]

[1] IIT Madras, Chennai, India
shweta.a@cse.iitm.ac.in, anshu.yadav06@gmail.com
[2] National Institute of Advanced Industrial Science and Technology (AIST), Tokyo
yamada-shota@aist.go.jp

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. We remark that our constructions provide the first candidates for a nontrivial class of miFE without needing LPN or low depth PRG.

Keywords: Multi-input · Attribute Based Encryption · Predicate Encryption.

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) [18, 49, 36, 29] 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 [48, 32, 18, 36, 37, 46, 47, 22, 6, 38, 39, 51, 28, 52, 13, 16, 29, 30, 20, 18, 49, 36, 29]. 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 [44, 24, 45] and functional encryption [27, 11, 4, 25, 3, 23, 50, 2, 1, 40, 7], this has not been investigated at all in the context of predicate encryption, and only sparingly [19] 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. [19] 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 [28]. 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* [48, 17], which has been studied extensively, in both the single-input [48, 17, 28, 16] and multi-input setting [27, 11, 4, 25, 3, 23, 50, 2, 1, 40, 7]. 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 $m_0 = m_1$[3]. 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.

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 [34, 11]. It was shown by [11, 15] 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) [14]. A long line of exciting works endeavoured to construct compact FE (and hence iO) from standard assumptions [41, 42, 43, 5, 12, 33, 26], 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 [34, 35]. In [34, 35], 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

[3] 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.

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 [34] 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 [35], 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 [11]. Unlike FE, both PE and ABE are *much simpler, "all or nothing"* primitives and permit direct constructions in the single-input setting [28, 16, 29]. 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 [4, 25, 3, 23, 50, 2, 1, 40, 7].
- *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 [34, 35], 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 [10] in the context of succinct *single*-input *ciphertext-policy* ABE, to the seemingly unrelated problem of *two*-input *key-policy* ABE. Similarly to [10], 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 [8].

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 [21] can also be seen from the lens of miABE and obtain the first two-input key-policy ABE from lattices for P. Similarly to [21], 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 [10] and the Brakerski-Vaikuntanathan [21] 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. [19] 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. We remark that our constructions provide the first candidates for a nontrivial class of miFE without needing LPN or low depth PRG.

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) [27], 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 $\{f_i\}_{i\in[\text{poly}]}$. 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. 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 2.

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., [32, 16]), 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 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 [10] 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 [16, 30], 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 [30, 28]. 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 [21, 10, 8] obtain by relying on the *decomposability* of specific ABE schemes [16, 30] – 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 [21, 10, 8]. 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. [16], which is denoted as BGG^+ hereafter. As discussed above, a ciphertext for an attribute $\mathbf{x} \in [2\ell]$[4] 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 the exponent of $\mathbb{G}_2$. In decryption, we compute a pairing of matching components of the two ciphertexts to obtain $(t_1t_2, \{t_1t_2\psi_{i,x_i}\}_{i\in[2\ell]})$ 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 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

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

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 [10] 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) [10] 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 3 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 [10] by Agrawal, Wichs and Yamada [8] 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 [8]. 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 [8], 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 [8], 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 full version of the paper [9, Sec. 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 [29]. 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 [30]. 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) [31, 53]. 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 2ABE secret key $\mathsf{2ABE.sk}_f$. The inner LO instance encodes the circuit for 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 2ABE ciphertext $\mathsf{2ABE.ct}_2$ with message β. The other piece of 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 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 4 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) 2ABE ciphertext when the lock is unlocked, which is then used to perform 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 2ABE. While oversimplified, this intuition may now be formalized into a proof. For more details, please see Section 5.

Constructing 3ABE from Pairings and Lattices. Finally, we discuss our candidate construction for three input ABE scheme based on techniques developed by Brakerski and Vaikuntanathan [21] in conjunction with our 2ABE construction in Section 3. The work of Brakerski and Vaikuntanathan [21] 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 [10] use pairings to generate shared randomness, [21] 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 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 [19], 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 [21]. 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 [21] 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 [21] 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.[5] 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 [21]. 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 [21] 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.

[5] 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.

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 [10] 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 [21], which we expect to be as secure as the original scheme. Please see Section 6 for more details.

2 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 1 (Ada-IND security for k-ABE). *For a* k-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* 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* $p = p(\lambda)$*. 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 2 (Ada-IND security for k-PE.). *For an k-PE scheme $\mathsf{k\text{-}PE} = \{\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}_1, \ldots, \mathsf{Enc}_k, \mathsf{Dec}\}$ 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*

$$\begin{cases} ((\alpha^i_{j,0}, \alpha^i_{j,1}), (b^i_0, b^i_1)) & \text{if } j = 1 \\ (\alpha^i_{j,0}, \alpha^i_{j,1}) & \text{if } j \neq 1 \end{cases}$$

   to the challenger, where $(\alpha^i_{j,0}, \alpha^i_{j,1})$ is a pair of attributes and (b^i_0, b^i_1) is the pair of messages. Then, the challenger computes and returns to $\mathcal{A}$

$$\mathsf{ct}^i_{j,\beta} = \begin{cases} \mathsf{Enc}_j(\mathsf{pp}, \mathsf{msk}, \alpha^i_{j,\beta}, b^i_\beta) & \text{if } j = 1 \\ \mathsf{Enc}_j(\mathsf{pp}, \mathsf{msk}, \alpha^i_{j,\beta}) & \text{if } j \neq 1 \end{cases}$$

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 $f_i(\alpha^{i_1}_{1,\beta}, \ldots, \alpha^{i_k}_{k,\beta}) = 0$ 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 $\mathsf{Adv}^{\mathsf{Ada\text{-}IND}}_{\mathsf{k\text{-}PE},\mathcal{A}}(1^\lambda) = \mathrm{negl}(\lambda)$.*

2.1 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 (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 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 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 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 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.*

2.2 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.

3 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 assuming LWE in bilinear generic group model. Since the intuition was described in Section 1, we proceed directly with the construction. We refer to the full version of the paper [9] for backgrounds on lattices and pairings respectively and for description of the kpABE scheme by Boneh et al. [16] on which our construction is based.

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 for the kpABE scheme of Boneh et al. (pl. see [9, Section 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 [9, 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 $\psi_{2\ell+1} := \mathbf{s}\mathbf{A} + \mathbf{e}$ and $\psi_{2\ell+2} := \mathbf{s}\mathbf{u}^\top + e_0$.
5. Set $\mu = \lceil \frac{q}{2} \rceil b$; $\mathbf{c} = (1, \{\psi_i\}_{i\in[\ell]}, \{\psi_{i,b}\}_{i\in[\ell+1,2\ell],b\in\{0,1\}}, \psi_{2\ell+1}, \psi_{2\ell+2} + \mu)$.
6. 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: Due to space constraints, we argue correctness in the full version [9, Sec. 4].

Proof of Security: We prove the security via the following theorem.

Theorem 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 (please see full version [9] for definition of* Ada-INDr *security).*

The proof is provided in the full version of the paper [9, Sec. 4].

4 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 Sec. 5.

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 1.
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 1.
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 $\gamma_i' \leftarrow \mathsf{SKE.Dec}(\mathsf{K}_i', \mathsf{ct}_i^*)$.

Fig. 1. 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. Here, we briefly discuss the correctness of the scheme. For the full proof, we refer to the full version of the paper [9, Sec. 6]. If we run $\mathsf{LO.Eval}(\mathsf{ct}_1', (\mathsf{ct}_2', \ldots, \mathsf{ct}_k', \mathsf{sk}_F))$, we end up with running the inner most obfuscation that obfuscates $f_k[\mathsf{ct}_k, \mathsf{ct}_k^*]$ on input $(\mathsf{ct}_1, \ldots, \mathsf{ct}_{k-1}, \mathsf{sk}_F)$. Within the circuit, K_k is retrieved by the kABE decryption and thus it recovers the lock value γ_k, which unlocks the obfuscation. The circuit outputs K_{k-1} and this is then input to the second-innermost obfuscated circuit, which outputs K_{k-2} because of the similar reason. This process continues until the outermost circuit is unlocked and outputs the hardwired message m.

Security. We can prove that the above PE construction satisfies Ada-IND security if so does the underlying kABE. We refer to the full version of the paper [9, Sec. 6] for the full proof. Here, we provide an intuition. First, we replace K_k encrypted in $\mathsf{kABE.Enc}_k$ with $\mathbf{0}$. This is possible by using the security of the underlying kABE. Then, we can use the security of SKE to change all ct_k^* hardwired in the slot k ciphertexts, since K_k is not used except for the encrypting the lock value γ_k due to the change introduced in the previous step. This allows us to change the circuit ct'_k into a simulated one rather than honestly obfuscated one, since the lock value γ_k is erased in the previous step. This in particular erases K_{k-1}, which allows to invoke the security of SKE to erase the lock value γ_{k-1}. This process continues until we can convert ct'_1 into a simulated circuit. At this point, every ciphertext is a simulated circuit and does not convey any information of attribute or message, as desired.

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 provided in the full version of the paper [9, Sec. 9] and the candidate three input ABE scheme in Sec. 6. The other schemes will be discussed in Sec. 5 because they satisfy strong (very selective) security and thus we can apply the conversion in Sec. 5. A nice property of the PE scheme obtained from the two input ABE scheme in [9, 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. 6, we can obtain a three-input PE scheme that can handle NC_1 circuits.

5 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. 4. Unlike the conversion in Sec. 4, we do not know how to extend it to general arity k and it is set to be $k = 2$ here. It 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 $\mathsf{Obf} = (\mathsf{LO.Obf}, \mathsf{LO.Eval})$.

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

$\mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, F)$: 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 $(\mathsf{2ABE.msk}, \mathsf{2ABE'.msk})$ and $\mathsf{pp} = (\mathsf{2ABE.pp}, \mathsf{2ABE'.pp})$.
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 2).
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 $\mathsf{2ABE}$ and $\mathsf{2ABE'}$ respectively.
2. Compute $r \leftarrow \mathsf{LO.Eval}(\tilde{G}, (\mathsf{ct}_1, \mathsf{sk}))$.
3. Output $\alpha' = \mathsf{2ABE'.Dec}(\mathsf{2ABE'.pp}, \mathsf{sk}', \mathsf{ct}'_1, r)$.

Fig. 2. 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. Compute $\mathsf{ct}_2 = \mathsf{2ABE.Enc}_2(\mathsf{2ABE.pp}, \mathsf{2ABE.msk}, \mathbf{x}_2, \beta)$, where $\beta \leftarrow \mathcal{M}$.
3. Compute $\mathsf{ct}'_2 = \mathsf{2ABE.Enc}'_2(\mathsf{2ABE'.pp}, \mathsf{2ABE'.msk}, \mathbf{x}_2)$.
4. Define a function $f_2[\mathsf{ct}_2]$, with ct_2 being hardwired, as in Figure 3.
5. 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 $\mathsf{2ABE}$ for the first slot and sk is regarded a secret key of $\mathsf{2ABE}$.
2. Output $\beta' \leftarrow \mathsf{2ABE.Dec}(\mathsf{2ABE.pp}, \mathsf{sk}, \mathsf{ct}_1, \mathsf{ct}_2)$.

Fig. 3. 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

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

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

Security. We prove security via the following theorem.

Theorem 2. *Assume* LO *is a secure lockable obfuscation scheme as per Definition 2.9 in [9] and that* 2ABE *and* $\mathsf{2ABE'}$ *are secure two input* ABE *schemes satisfying strong security as in Definition 3 (resp., strong very selective security as in Definition 5). Then, the* 2PE *construction presented above satisfies strong security as per Definition 4 (resp., strong very selective security as in Definition 5).*

Due to space constraints, the proof is provided in the full version [9, Sec. 7].

Applications. By applying the above conversion to two input ABE scheme with strong security in Sec. 3, 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. 3 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 standard model provided in the full version [9, Sec. 5]. Since the scheme is proven secure in the standard model, the construction does not suffer from the above problem.

6 Three-Input ABE from Pairings and Lattices

In this section, we provide a candidate construction for 3ABE using the structure of [21] and [10] as discussed in Section 1. Leveraging ideas from the Brakerski-Vaikuntanathan construction [21], we also obtain a candidate for 2ABE for P – due to space constraints, we provide this construction in the full version [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 $\ell(\lambda) \le \mathrm{poly}(\lambda)$ 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 $d(\lambda)$.

$\mathsf{Setup}(1^\lambda)$: 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 later. 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 $i \in [\ell]$, $\mathbf{E}_{i,b} \leftarrow \hat{\chi}^{k\times m}$, for $i \in [\ell+1, 3\ell]$, $b \in \{0,1\}$.
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 $\psi_{3\ell+1} := \mathbf{S}\mathbf{A} + \mathbf{E} \in \mathbb{Z}_q^{k\times m}$ and $\psi_{3\ell+2}^\top := \mathbf{S}\mathbf{u}^\top + \mathbf{e}_0^\top \in \mathbb{Z}_q^{k\times 1}$.
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:
$\widehat{\mathbf{C}}_1 = (\{\psi_{i,x_{1i}}\}_{i\in[\ell]}, \{\psi_{i,b}\}_{i\in[\ell+1,2\ell],\, b\in\{0,1\}}, \{\mathbf{C}_{i,b}\widehat{\mathbf{S}}_{i,b} + \widehat{\mathbf{E}}_{i,b} + \psi_{i,b}\}_{i\in[2\ell+1,3\ell],\, b\in\{0,1\}},$
$\mathbf{C}_{3\ell+1}\widehat{\mathbf{S}}_{3\ell+1} + \widehat{\mathbf{E}} + \psi_{3\ell+1}, \mathbf{C}_{3\ell+2}\widehat{\mathbf{s}}_{3\ell+2}^{\top} + \widehat{\mathbf{e}}_0^{\top} + \psi_{3\ell+2}^{\top} + \mathbf{m}^{\top}) \in \mathbb{Z}_q^{k\times L}$.
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 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

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

$[\mathbf{V}_{i,b}]_T = [t_{\mathbf{x}_1} t_{\mathbf{x}_2}(\psi_{i,b} + \mathbf{C}_{i,b}\widehat{\mathbf{S}}_{i,b} + \widehat{\mathbf{E}}_{i,b})]_T$ 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 $[\mathbf{t}_{\mathbf{x}_3}\mathbf{V}_{\mathbf{x}}]_T = [\mathbf{t}_{\mathbf{x}_3}\mathbf{V}_1 \| \ldots \| \mathbf{t}_{\mathbf{x}_3}\mathbf{V}_{3\ell}]_T$, $\mathbf{r} = (\mathbf{r}_1 \in \mathbb{Z}_q^m, \mathbf{r}_2 \in \mathbb{Z}_q^m)$.
   (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.

Parameters and Correctness: We choose the parameters for the 3-ABE scheme as follows (pl. refer to the full version [9] for definition of SampZ):

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

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 $c > 1$.

We argue correctness in the full version of the paper [9, Sec. 8].

Acknowledgement The third author was partially supported by JST AIP Acceleration Research JPMJCR22U5 and JSPS KAKENHI Grant Number 19H01109, Japan.

References

1. M. Abdalla, F. Benhamouda, and R. Gay. From single-input to multi-client inner-product functional encryption. In *ASIACRYPT*, 2019.
2. M. Abdalla, F. Benhamouda, M. Kohlweiss, and H. Waldner. Decentralizing inner-product functional encryption. In *PKC*, 2019.
3. M. Abdalla, D. Catalano, D. Fiore, R. Gay, and B. Ursu. Multi-input functional encryption for inner products: Function-hiding realizations and constructions without pairings. In *CRYPTO*, 2018.
4. M. Abdalla, R. Gay, M. Raykova, and H. Wee. Multi-input inner-product functional encryption from pairings. In *EUROCRYPT*, 2017.

5. S. Agrawal. Indistinguishability obfuscation without multilinear maps: New techniques for bootstrapping and instantiation. In *Eurocrypt*, 2019.
6. S. Agrawal, D. M. Freeman, and V. Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In *Asiacrypt*, 2011.
7. S. Agrawal, R. Goyal, and J. Tomida. Multi-input quadratic functional encryption from pairings. In *CRYPTO*, 2021.
8. S. Agrawal, D. Wichs, and S. Yamada. Optimal broadcast encryption from lwe and pairings in the standard model. In *TCC*, 2020.
9. S. Agrawal, A. Yadav, and S. Yamada. Multi-input attribute based encryption and predicate encryption. IACR Cryptology ePrint Archive, 2022, 2022.
10. S. Agrawal and S. Yamada. Optimal broadcast encryption from pairings and lwe. In *EUROCRYPT*, 2020.
11. P. Ananth and A. Jain. Indistinguishability obfuscation from compact functional encryption. In *CRYPTO*, 2015.
12. P. Ananth, A. Jain, H. Lin, C. Matt, and A. Sahai. Indistinguishability obfuscation without multilinear maps: iO from LWE, bilinear maps, and weak pseudorandomness. In *CRYPTO*, 2019.
13. N. Attrapadung. Dual system encryption via doubly selective security: Framework, fully secure functional encryption for regular languages, and more. In *Eurocrypt*, 2014.
14. 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.
15. N. Bitansky and V. Vaikuntanathan. Indistinguishability obfuscation from functional encryption. *FOCS*, 2015.
16. D. Boneh, C. Gentry, S. Gorbunov, S. Halevi, V. Nikolaenko, G. Segev, V. Vaikuntanathan, and D. Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In *EUROCRYPT*, 2014.
17. D. Boneh, A. Sahai, and B. Waters. Functional encryption: Definitions and challenges. In *TCC*, 2011.
18. D. Boneh and B. Waters. Conjunctive, subset, and range queries on encrypted data. In *TCC*, 2007.
19. Z. Brakerski, A. Jain, I. Komargodski, A. Passelègue, and D. Wichs. Non-trivial witness encryption and null-io from standard assumptions. In *SCN*, 2018.
20. Z. Brakerski and V. Vaikuntanathan. Circuit-abe from lwe: Unbounded attributes and semi-adaptive security. In M. Robshaw and J. Katz, editors, *Advances in Cryptology – CRYPTO*, 2016.
21. Z. Brakerski and V. Vaikuntanathan. Lattice-inspired broadcast encryption and succinct ciphertext policy abe. In *ITCS*, 2022.
22. J. Chen and H. Wee. Semi-adaptive attribute-based encryption and improved delegation for boolean formula. In *SCN*, 2014.
23. J. Chotard, E. Dufour Sans, R. Gay, D. H. Phan, and D. Pointcheval. Decentralized multi-client functional encryption for inner product. In *ASIACRYPT*, 2018.
24. M. Clear and C. McGoldrick. Multi-identity and multi-key leveled FHE from learning with errors. In *CRYPTO*, 2015.
25. P. Datta, T. Okamoto, and J. Tomida. Full-hiding (unbounded) multi-input inner product functional encryption from the k-Linear assumption. In *PKC*, 2018.
26. R. Gay, A. Jain, H. Lin, and A. Sahai. Indistinguishability obfuscation from simple-to-state hard problems: New assumptions, new techniques, and simplification. In *EUROCRYPT*, 2021.
27. S. Goldwasser, S. D. Gordon, V. Goyal, A. Jain, J. Katz, F.-H. Liu, A. Sahai, E. Shi, and H.-S. Zhou. Multi-input functional encryption. In *EUROCRYPT*, 2014.

28. S. Gorbunov, V. Vaikuntanathan, and H. Wee. Attribute based encryption for circuits. In *STOC*, 2013.
29. S. Gorbunov, V. Vaikuntanathan, and H. Wee. Predicate encryption for circuits from lwe. In *Crypto*, 2015.
30. S. Gorbunov and D. Vinayagamurthy. Riding on asymmetry: Efficient ABE for branching programs. In *ASIACRYPT*, 2015.
31. R. Goyal, V. Koppula, and B. Waters. Lockable obfuscation. In *FOCS*, 2017.
32. V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM CCS*, 2006.
33. A. Jain, H. Lin, C. Matt, and A. Sahai. How to leverage hardness of constant-degree expanding polynomials over r to build io. In *EUROCRYPT*, 2019.
34. A. Jain, H. Lin, and A. Sahai. Indistinguishability obfuscation from well-founded assumptions. In *STOC*, 2021.
35. A. Jain, H. Lin, and A. Sahai. Indistinguishability obfuscation from lpn over large fields, dlin, and constant depth prgs. In *EUROCRYPT*, 2022.
36. J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT*, 2008.
37. A. B. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *EUROCRYPT*, 2010.
38. A. B. Lewko and B. Waters. Unbounded HIBE and attribute-based encryption. In *Eurocrypt*, pages 547–567, 2011.
39. A. B. Lewko and B. Waters. New proof methods for attribute-based encryption: Achieving full security through selective techniques. In *Crypto*, 2012.
40. B. Libert and R. Titiu. Multi-client functional encryption for linear functions in the standard model from LWE. In *ASIACRYPT*, 2019.
41. H. Lin. Indistinguishability obfuscation from constant-degree graded encoding schemes. In *EUROCRYPT*, 2016.
42. H. Lin. Indistinguishability obfuscation from sxdh on 5-linear maps and locality-5 PRGs. In *Crypto*, 2017.
43. H. Lin and V. Vaikuntanathan. Indistinguishability obfuscation from ddh-like assumptions on constant-degree graded encodings. In *FOCS*, 2016.
44. A. López-Alt, E. Tromer, and V. Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *STOC*, 2012.
45. P. Mukherjee and D. Wichs. Two round multiparty computation via multi-key fhe. In *EUROCRYPT*, 2016.
46. T. Okamoto and K. Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In *CRYPTO*, pages 191–208, 2010.
47. T. Okamoto and K. Takashima. Adaptively attribute-hiding (hierarchical) inner product encryption. In *EUROCRYPT*, 2012.
48. A. Sahai and B. Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, 2005.
49. E. Shi, J. Bethencourt, T. H. Chan, D. Song, and A. Perrig. Multi-dimensional range query over encrypted data. In *SP*, 2007.
50. J. Tomida. Tightly secure inner product functional encryption: Multi-input and function-hiding constructions. In *ASIACRYPT*, 2019.
51. B. Waters. Functional encryption for regular languages. In *Crypto*, 2012.
52. H. Wee. Dual system encryption via predicate encodings. In *TCC*, 2014.
53. D. Wichs and G. Zirdelis. Obfuscating compute-and-compare programs under LWE. In *FOCS*, 2017.