




Succinct and Adaptively Secure ABE for ABP from k -Lin

Huijia Lin^(✉) and Ji Luo^(✉) 

University of Washington, Seattle, USA
{rachel,luoji}@cs.washington.edu

Abstract. We present *succinct* and *adaptively secure* attribute-based encryption (ABE) schemes for *arithmetic branching programs*, based on k -Lin in pairing groups. Our key-policy ABE scheme has ciphertexts of *constant size*, independent of the length of the attributes, and our ciphertext-policy ABE scheme has secret keys of *constant size*. Our schemes improve upon the recent succinct ABE schemes in [Tomida and Attrapadung, ePrint '20], which only handles Boolean formulae. All other prior succinct ABE schemes either achieve only selective security or rely on q -type assumptions.

Our schemes are obtained through a general and modular approach that combines a public-key inner product functional encryption satisfying a new security notion called gradual simulation security and an information-theoretic randomized encoding scheme called arithmetic key garbling scheme.

1 Introduction

Attribute-based encryption (ABE) [21] is an advanced form of public-key encryption for enforcing fine-grained access control. In the key-policy version, an authority generates a pair of master public and secret keys mpk, msk . Given mpk , everyone can encrypt a message m with an attribute x to get a ciphertext $\text{ct}_x(m)$. Using the master secret key msk , the authority can issue a secret key sk_y tied to a policy y . Decrypting a ciphertext $\text{ct}_x(m)$ using sk_y recovers the encrypted message m if the attribute x satisfies the policy y . Otherwise, no information about m is revealed. The security requirement of ABE mandates collusion resistance—no information of m should be revealed, even when multiple secret keys are issued, as long as none of them individually decrypts the ciphertext (i.e., the attribute satisfies none of the associated policies).

Over the past decade, a plethora of ABE schemes have been proposed for different expressive classes of policies, achieving different trade-offs between efficiency, security, and assumptions. Meanwhile, ABE has found numerous cryptographic and security applications. A primary desiderata of ABE schemes is efficiency, in particular, having fast encryption algorithms and small ciphertexts. It turns out that the size of ABE ciphertexts can be *independent* of the length of the attribute x , and dependent only on the length of the message m and security parameter—we say such ciphertexts are *succinct* or have *constant size* (in

attribute length). Proposed first in [11] as a goal, succinct ciphertexts are possible because ABE does not require hiding the attribute x , and the decryption algorithm can take x as input in the clear. Consequently, ciphertexts only need to contain enough information of x to enforce the *integrity* of computation on x , which does not necessitate encoding the entire x .

Succinct ABE are highly desirable. For practical applications of ABE where long attributes are involved for sophisticated access control, succinct ciphertexts are much more preferable. From a theoretical point of view, succinct ciphertexts have (asymptotically) *optimal* size, as dependency on the message length and security parameter is inevitable. From a technical point of view, succinct ABE provides interesting mechanism for enforcing the integrity of computation without encoding the input. So far, several succinct ABE schemes have been proposed [5–7, 22, 23, 27, 28], but almost all schemes either rely on non-standard assumption or provide only weak security, as summarized in Tables 1 and 2.

Our Results. In this work, we first construct a *succinct* key-policy ABE (KP-ABE) simultaneously satisfying the following properties.

- (1) Expressiveness. Support policies expressed as arithmetic branching programs (ABPs).
- (2) Security. Satisfy adaptive security, as opposed to selective or semi-adaptive security.
- (3) Assumption. Based on the standard assumptions as opposed to, e.g., q -type assumptions. Specifically, our scheme relies on the matrix decisional Diffie–Hellman (MDDH) assumption over pairing groups.
- (4) Efficiency. Has succinct ciphertext.

Concretely, each ciphertext consists of 5 group elements when assuming SXDH, and $2k + 3$ elements for MDDH $_k$ (implied by k -Lin). Decryption involves the same number of pairing operations. Additionally, our scheme can work with the more efficient asymmetric prime-order pairing groups.

Next, we construct ciphertext-policy ABE (CP-ABE) with the same properties. Here, the secret keys are tied to attributes and ciphertexts to policies, and succinctness refers to having constant-size secret keys. Our scheme has keys consisting of 7 group elements based on SXDH and $3k + 4$ based on MDDH $_k$.

Besides succinctness (4), achieving the strong notion of adaptive security (2) based on standard assumptions (3) is also highly desirable from both a practical and a theoretical point of view. Prior to this work, only the recent construction of (KP and CP) ABE schemes by Tomida and Attrapadung [23] simultaneously achieves (2)–(4), and their scheme handles policies expressed as Boolean formulae. Our construction expands the class of policies to *arithmetic* branching programs, which is a more expressive model of computation. Our succinct ABE is also the first scheme natively supporting *arithmetic* computation over large fields,¹ whereas all prior succinct ABE schemes (even ones relying on q -type

¹ One can always convert an arithmetic computation into a Boolean one, which we consider non-native.

Table 1. KP-ABE schemes with succinct ciphertext.

Reference	Policy	Assumption	Adaptive	mpk	sk	ct	Dec
ALP [7]	MSP	q -type		$2n + 1$	$m(n + 1)$	3	3
YAHK [27]	MSP	q -type		$n + 2$	$m(n + 1)$	2	2
Tak [22]	MSP	2-Lin	✓	$18(2n + 1)$	$6m(n + 1)$	17	17
Att [5]	MSP	q -type	✓	$6n + 42$	$3m(n + 3) + 9$	18	18
ZGT ⁺ [28]	MSP	k -Lin	✓	$2k^2(n + 1)$	$2km(n + 1)$	$4k$	$4k$
TA [23]	NC ¹	MDDH _{k}	✓	$k(k + 1)(n + 3)$	$(k + 1)m(n + 2)$	$2k + 2$	$2k + 2$
Sect. 5	ABP	MDDH _{k}	✓	$k(k + 2)(n + 2)$	$(k + 1)m(n + 2) + m$	$2k + 3$	$2k + 3$

MSP: monotone span programs. NC¹: Boolean formulae. ABP: arithmetic branching programs. n = attribute length, m = policy size, p = group order. |mpk|, |sk|, |ct| counts non-generator elements in source groups. Dec counts the number of pairing operations in decryption. Schemes based on k -Lin can be based on MDDH _{k} at the cost of a few more elements in mpk. ABE for arithmetic span programs can be obtained by reduction to MSP [6].

Table 2. CP-ABE schemes with succinct secret key.

Reference	Policy	Assumption	Adaptive	mpk	sk	ct	Dec
Att [5]	MSP	q -type	✓	$6n + 54$	24	$3m(n + 3) + 15$	24
AHY [6]	ASP	q -type	✓	$O(n \log p)$	$O(1)$	$O(mn \log p)$	$O(1)$
TA [23]	NC ¹	MDDH _{k}	✓	$O(k^2 n)$	$O(k)$	$O(kmn)$	$O(k)$
Ours [18]	ABP	MDDH _{k}	✓	$k(k + 1)(n + 4) + k$	$3k + 4$	$(k + 1)m(n + 2) + m + k + 1$	$3k + 4$

assumptions and/or achieving only selective security) only work natively with Boolean computation. Lastly, we note that even when relaxing the efficiency requirement from having succinct ciphertext to *compact* ciphertext, whose size grows linearly with the length of the attribute, only a few schemes [13, 15, 17] simultaneously achieve (2)–(4), and the most expressive class of policies supported is also ABP, due to [17].

Our Techniques. The recent work of [17] presented a general framework for constructing *compact* adaptively secure ABE from MDDH. In this work, we improve their general framework to achieve *succinctness*. The framework of [17] yields linear-size ciphertexts because it crucially relies on *function-hiding* inner-product functional encryption (IPFE) [10, 19]. IPFE allows issuing secret keys and ciphertexts tied to vectors \mathbf{v}, \mathbf{u} respectively, and decryption reveals their inner product $\langle \mathbf{u}, \mathbf{v} \rangle$. The function-hiding property guarantees that nothing about \mathbf{u}, \mathbf{v} beyond the inner product is revealed, which entails that ciphertexts and secret keys must have size linear in the length of the vectors.

Towards succinctness, our key idea is relaxing function-hiding to a *new and weaker* guarantee, called *gradual simulation security*, where only the vectors encrypted in the ciphertexts are hidden. Such IPFE can have succinct (constant-size) secret keys and can be public-key. We use new ideas to modify the framework of [17] to work with the weaker gradual simulation security and obtain suc-

cinct ciphertexts. Furthermore, we extend the framework to construct ciphertext-policy ABE, which is not handled in [17]. In summary, our techniques give a general and modular approach for constructing succinct and adaptively secure (KP and CP) ABE from MDDH.

Organization. In Sect. 1.1, we give an overview of how we construct our ABE schemes using inner-product functional encryption (IPFE) schemes with gradual simulation security and dual system encryption. We discuss related works in Sect. 1.2. After introducing the preliminaries in Sect. 2, we define gradual simulation security of IPFE and construct such an IPFE scheme in Sect. 3. In Sect. 4, we define 1-ABE and construct CP-1-ABE for ABP with succinct keys from gradually simulation-secure IPFE with succinct keys. In Sect. 5, we show how to construct KP-ABE with succinct ciphertexts using our CP-1-ABE and dual system encryption. Due to the lack of space, we refer the reader to the full version [18] for our construction of CP-ABE with succinct keys similarly to our KP-ABE construction.

1.1 Technical Overview

In this section, we give an overview of our construction of succinct ABE schemes, following the roadmap shown in Fig. 1.

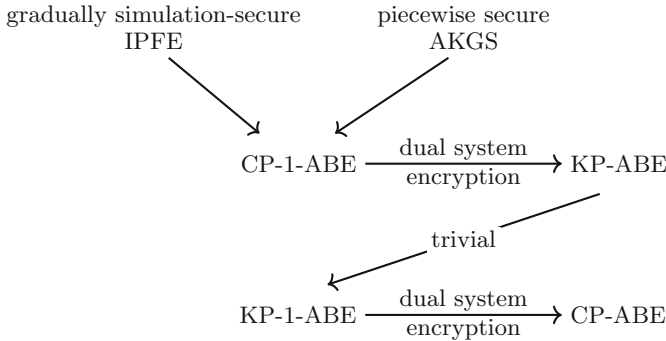


Fig. 1. The roadmap of our constructions.

1-ABE. The core of many ABE schemes is a 1-key 1-ciphertext secure secret-key ABE, or 1-ABE for short. Our construction improves the recent 1-ABE scheme for ABP by Lin and Luo (LL) [17], which achieves adaptive security but not succinctness.

Suppose we want decryption to recover the message $\mu \in \mathbb{Z}_p$ if (and only if) $f(\mathbf{x}) \neq 0$ for policy function $f : \mathbb{Z}_p^n \rightarrow \mathbb{Z}_p$ and attribute $\mathbf{x} \in \mathbb{Z}_p^n$. This is

equivalent to computing $\mu f(\mathbf{x})$ upon decryption. The basic idea of the LL 1-ABE is that when a key (tied to f, μ)² and a ciphertext (tied to \mathbf{x}) are put together, one can compute a randomized encoding of $\mu f(\mathbf{x})$, denoted by $\widehat{\mu f(\mathbf{x})}$, which reveals $\mu f(\mathbf{x})$ and hence μ if $f(\mathbf{x}) \neq 0$. Since in ABE, we do not try to hide f or \mathbf{x} , the randomized encoding only needs to hide μ beyond the output $\mu f(\mathbf{x})$, referred to as the partially hiding property, first introduced by [14]. Due to the weak security guarantee, partially hiding randomized encoding can have extremely simple structure. In particular, LL defined a refined version of such randomized encoding, called arithmetic key garbling scheme (AKGS), with the following properties:

Linear Encoding. The encoding is in the form of

$$\widehat{\mu f(\mathbf{x})} = (L_1(\mathbf{x}), \dots, L_m(\mathbf{x})),$$

where L_j 's are *affine* functions of \mathbf{x} and the coefficients of L_j 's are *linear* in the message μ and the garbling randomness. L_j 's are called *label functions* and $\ell_j = L_j(\mathbf{x})$ are called *labels*.

Linear Evaluation. There is a procedure *Eval* that can compute $\mu f(\mathbf{x})$ from f, \mathbf{x} and the labels:

$$\text{Eval}(f, \mathbf{x}, \ell_1, \dots, \ell_m) = \mu f(\mathbf{x}).$$

Importantly, *Eval* is linear in the labels.³

The basic security of AKGS is simulation security. There needs to be an efficient simulator *Sim* that can perfectly simulate the labels given $f, \mathbf{x}, \mu f(\mathbf{x})$:

$$\text{Sim}(f, \mathbf{x}, \mu f(\mathbf{x})) \rightarrow (\ell_1, \dots, \ell_m) \equiv (L_1(\mathbf{x}), \dots, L_m(\mathbf{x})).$$

Since the label functions are affine in \mathbf{x} thus linear in $(1, \mathbf{x})$, the labels $\ell_j = L_j(\mathbf{x})$ can be securely computed using a *function-hiding* IPFE. In IPFE, keys $\text{isk}(\mathbf{v})$ and ciphertexts $\text{ict}(\mathbf{u})$ are generated for vectors \mathbf{v}, \mathbf{u} , and decryption yields their inner product $\langle \mathbf{u}, \mathbf{v} \rangle$ but nothing else. More precisely, function-hiding says two sets of keys and ciphertexts encoding different vectors are indistinguishable as long as they yield identical inner products:

$$(\{\text{isk}_j(\mathbf{v}_j)\}, \{\text{ict}_i(\mathbf{u}_i)\}) \approx (\{\text{isk}_j(\mathbf{v}'_j)\}, \{\text{ict}_i(\mathbf{u}'_i)\}) \text{ if } \langle \mathbf{u}_i, \mathbf{v}_j \rangle = \langle \mathbf{u}'_i, \mathbf{v}'_j \rangle \text{ for all } i, j.$$

That is, all vectors no matter encoded in keys or ciphertexts are protected. Moreover, function-hiding should hold even when these vectors are chosen adaptively by the adversary, depending on previously observed keys and ciphertexts.

In the LL 1-ABE scheme, an ABE key consists of many IPFE keys encoding the coefficients of the label functions (also denoted by L_j), and an ABE

² The reason why we put the message μ in the key will become clear later in the overview.

³ In contrast, linear evaluation is impossible for fully hiding randomized encoding that hides \mathbf{x} and f .

ciphertext is an IPFE ciphertext encrypting $(1, \mathbf{x})$, as illustrated below in Real Algorithms. When they are put together, IPFE decryption recovers exactly the labels $\ell_j = L_j(\mathbf{x}) = \langle L_j, (1, \mathbf{x}) \rangle$, from which we can recover $\mu f(\mathbf{x})$ using the evaluation procedure. A technicality is that known IPFE are built from pairing groups, and decryption only reveals $\mu f(\mathbf{x})$ in the exponent of the target group. Nevertheless, one can recover $\mu f(\mathbf{x})$ also in the exponent, thanks to the linearity of AKGS evaluation.

Intuitively, the LL scheme is secure since IPFE only reveals the labels, and AKGS security guarantees only $\mu f(\mathbf{x})$ is revealed, given the labels. It is simple to formalize this idea in the selective setting, where \mathbf{x} is chosen before querying the key for f . By the function-hiding property, it is indistinguishable to hardwire the labels in the IPFE keys as follows.

$$\begin{array}{ccc}
 \text{REAL ALGORITHMS} & & \text{HYBRID} \\
 \left\{ \begin{array}{l} \text{ct}_{\mathbf{x}}: \text{ict} (1, \mathbf{x}) \\ \text{sk}_{f,\mu}: \{ \text{isk}_j (L_j) \}_{j \in [m]} \} \end{array} \right\} & \approx & \left\{ \begin{array}{l} \text{ct}_{\mathbf{x}}: \text{ict} (1, \mathbf{x}) \\ \text{sk}_{f,\mu}: \{ \text{isk}_j (L_j(\mathbf{x}), \mathbf{0}) \}_{j \in [m]} \} \end{array} \right\}
 \end{array}$$

After labels $L_j(\mathbf{x})$ are hardwired and label functions removed, AKGS security guarantees that the labels only reveal $\mu f(\mathbf{x})$, and μ is hidden if $f(\mathbf{x}) = 0$. Observe that for selective security, we only need hiding in the keys and not the ciphertext.

The above proof fails for adaptive security, in particular in the case where the secret key is queried before the ciphertext (we will focus on this harder case below). At key generation time, \mathbf{x} is unknown and consequently the labels $L_j(\mathbf{x})$ are unknown. We also do not want to hardwire all the labels in the ciphertext as that would make the ciphertext as large as the policy. LL solves this problem by relying on a stronger security notion of AKGS called *piecewise security*:

- The marginal distribution of ℓ_2, \dots, ℓ_m is uniformly random, and ℓ_1 can be *reversely computed* from these other labels ℓ_2, \dots, ℓ_m and f, \mathbf{x} , by finding the unique ℓ_1 satisfying the constraint of evaluation correctness.⁴
- The other labels are *marginally random* even given the coefficients of all subsequent label functions, i.e.,

$$(L_j(\mathbf{x}), L_{j+1}, \dots, L_m) \equiv (z, L_{j+1}, \dots, L_m) \quad \text{for } z \stackrel{\$}{\leftarrow} \mathbb{Z}_p, \quad \text{for all } j > 1.$$

The first property implies a specific simulation strategy: Simply sample ℓ_2, \dots, ℓ_m as random, then solve for ℓ_1 from the correctness constraint. This strategy is particularly suitable for the adaptive setting, as only the simulation of ℓ_1 depends on the input \mathbf{x} . Thus, a conceivable simulation strategy for 1-ABE is to hardwire ℓ_2, \dots, ℓ_m in the secret key and ℓ_1 in the ciphertext. This would not hurt the compactness of the ciphertext.

⁴ The original definition only requires ℓ_1 to be reversely *sampleable*. In [17], it is shown that the two are equivalent for piecewise security, and we stick to the simpler definition in this overview. In the full definition, ℓ_1 also depends on the computation result. For the purpose of this overview, the result is always $\mu f(\mathbf{x}) = 0$ as the adversary is restricted to non-decrypting queries.

Proving the indistinguishability of the real and the simulated worlds takes two steps. In the first step, the first label $\ell_1 = L_1(\mathbf{x})$ is hardwired into the IPFE ciphertext ict , and then changed to be reversely computed from the other labels and f, \mathbf{x} , which is possible since by the time we generate ict , we know both f and \mathbf{x} . In the second step, each isk_j for $j > 1$ is, one by one, switched from encoding the label function to encoding a random label. To do so, the j^{th} label $\ell_j = L_j(\mathbf{x})$ is first hardwired into ict , after which it is switched to random relying on piecewise security, and lastly moved back to isk_j . Observe that the proof uses two extra slots in the vectors (one for ℓ_1 , the other for each ℓ_j temporarily) and relies on hiding in both the keys and the ciphertext.

Lightweight Alternative to Function-Hiding. In a function-hiding IPFE, keys and ciphertexts must be of size at least linear in the vector dimension. This means the resulting ABE scheme can never be succinct. Our first observation is that function-hiding IPFE is an overkill. Since in ABE, \mathbf{x} is not required to be hidden, it is quite wasteful to protect it inside an IPFE ciphertext. Indeed, selective security of the LL scheme does not rely on hiding in the ciphertext.

Our idea to achieve succinctness is to use a non-function-hiding IPFE scheme instead, e.g., public-key IPFE. Usually the vector in the key is included verbatim as part of the key, and the “essence” of the key (excluding the vector itself) could be significantly shorter than the vector. Indeed, many known public-key IPFE schemes [1, 3] have succinct keys.

Since the coefficients of the label functions (which contains information about μ and the garbling randomness) *must* be hidden for the 1-ABE to be secure, and \mathbf{x} is public, we should encrypt the coefficients of the label functions in IPFE ciphertexts and use an IPFE key for $(1, \mathbf{x})$ to compute the garbling. Since the message μ is together with f and the generation of IPFE ciphertexts is public-key, the 1-ABE scheme is more like a *public-key ciphertext-policy* ABE than a secret-key ABE, except we only consider security given a single key for some attribute \mathbf{x} . Therefore, we redefine 1-ABE as 1-key secure public-key CP-ABE,⁵ and the idea is to construct it from a public key IPFE and AKGS as follows:

$$\left. \begin{array}{l} \text{sk}_{\mathbf{x}}: \text{isk} (1, \mathbf{x}) \\ \text{ct}_{f, \mu}: \{\text{ict}_j(L_j)\}_{j \in [m]} \end{array} \right\} \xrightarrow[\text{Dec}]{\text{IPFE}} \{\langle L_j, (1, \mathbf{x}) \rangle = L_j(\mathbf{x}) = \ell_j\}_{j \in [m]} \xrightarrow[\text{Eval}]{\text{AKGS}} \mu f(\mathbf{x}).$$

Our CP-1-ABE is \mathbf{x} -selectively secure if the underlying IPFE is indistinguishability-secure, similar to the selective security of LL scheme.

However, it is not immediate that we can prove adaptive security of this new scheme. The LL adaptive security proof requires hardwiring ℓ_1 and one of ℓ_j 's with \mathbf{x} , which is now encoded in the secret key without hiding property. Taking a step back, hardwiring a label is really about removing its label function and only using the label, which is the inner product yielded by IPFE decryption. Our idea is to use simulation security to achieve this goal. A simulator for a

⁵ This definition has the advantage of automatically being multi-ciphertext secure (if secure at all) over the secret-key definition. It is also more convenient to use in reductions for full ABE.

public-key IPFE can simulate the master public key, the secret keys, and one (or a few) ciphertext, using only the inner products, and the simulator can do so adaptively. Let us take simulating one ciphertext as an example.

$$\begin{array}{cc}
 \text{REAL} & \text{SIMULATION} \\
 \left\{ \begin{array}{l} \text{mpk} \\ \{\text{isk}_j(\mathbf{v}_j)\}_{j \leq J^*} \\ \text{ict}(\mathbf{u}) \\ \{\text{isk}_j(\mathbf{v}_j)\}_{j > J^*} \end{array} \right\} & \approx \left\{ \begin{array}{l} \widetilde{\text{mpk}} \\ \{\widetilde{\text{isk}}_j(\mathbf{v}_j \mid \emptyset)\}_{j \leq J^*} \\ \widetilde{\text{ict}}(\emptyset \mid \{\langle \mathbf{u}, \mathbf{v}_j \rangle\}_{j \leq J^*}) \\ \{\widetilde{\text{isk}}_j(\mathbf{v}_j \mid \langle \mathbf{u}, \mathbf{v}_j \rangle)\}_{j > J^*} \end{array} \right\} \quad (*)
 \end{array}$$

J^* is the number of keys issued before ciphertext generation. On the left are the honestly generated master public key, secret keys, and ciphertext. On the right is their simulation. The vertical bar separates what the real algorithms use and what the simulator (additionally) use. Since public-key IPFE completely reveals the key vectors,⁶ they are always provided to the simulator. As for the other values:

- Before ciphertext simulation, there is no additional information supplied.
- When the ciphertext is simulated, the vector \mathbf{u} is *not* provided, but its inner products with already simulated keys are provided to the simulator.⁷
- After ciphertext simulation, when simulating a key for \mathbf{v}_j , the inner product $\langle \mathbf{u}, \mathbf{v}_j \rangle$ is provided with \mathbf{v}_j .

Observe that the values after the vertical bar are exactly those computable using the functionality of IPFE *at that time*, so in simulation, anything about the encrypted vector *not yet* computable by the functionality of IPFE, simply does not exist (information-theoretically) at all. In the setting of our CP-1-ABE, we will simulate an IPFE ciphertext to remove its corresponding label function and only retain the label. Looking from the perspective of hardwiring, when we issue $\text{sk}_x = \text{isk}(1, \mathbf{x})$ after we have created the ciphertext $\text{ct}_{f,\mu}$ (in which ict_j has been simulated), the inner product ℓ_j is supplied to the simulator when we simulate isk , after the simulation of ict_j . This means the label ℓ_j is hardwired into isk .

Let us exemplify the proof of adaptive security in the more difficult case where sk_x is queried after $\text{ct}_{f,\mu}$. First, we simulate ict_1 so that the first label is hardwired into isk .

$$\begin{array}{cc}
 \text{REAL ALGORITHMS} & \ell_1 \text{ HARDWIRED} \\
 \left\{ \begin{array}{l} \text{ct}_{f,\mu}: \text{ict}_1(L_1) \\ \{\text{ict}_j(L_j)\}_{j > 1} \\ \text{sk}_x: \text{isk}(1, \mathbf{x}) \end{array} \right\} & \approx \left\{ \begin{array}{l} \text{ct}_{f,\mu}: \widetilde{\text{ict}}_1(\emptyset \mid \emptyset) \\ \{\widetilde{\text{ict}}_j(L_j)\}_{j > 1} \\ \text{sk}_x: \widetilde{\text{isk}}(1, \mathbf{x} \mid \ell_1 = L_1(\mathbf{x})) \end{array} \right\}
 \end{array}$$

⁶ Anyone can encrypt the standard basis vectors using mpk , and use decryption algorithm to obtain each component of the vector in a secret key.

⁷ Though the number J^* of inner products with already simulated keys is unbounded, since the vectors $\{\mathbf{v}_j\}_{j \leq J^*}$ in the keys are public, these inner products are determined by those with any maximal subset of linearly independent \mathbf{v}_j 's, the number of which will not exceed the dimension. As such, the simulated ciphertext can still be compact.

(We omitted the master public key for brevity.) Note that $\widetilde{\text{ict}}_j$'s for $j > 1$ do not use *ciphertext* simulation but are created using the master public key (honest or simulated). Once ℓ_1 is hardwired, we can instead solve for it from the correctness equation.

The second step is to switch $\text{ict}_j(L_j)$ to $\text{ict}_j(\ell_j, \mathbf{0})$ for $\ell_j \xleftarrow{\$} \mathbb{Z}_p$ one by one, i.e., to simulate ℓ_j as random. To do so, we first simulate ict_j (hardwiring $\ell_j = L_j(\mathbf{x})$ into isk), then switch ℓ_j to random (via piecewise security), and lastly revert ict_j back to encryption (not simulated), but encrypting $(\ell_j, \mathbf{0})$ instead.

$$\begin{array}{l}
 \text{BEFORE/AFTER SIMULATING } \ell_j \\
 \left\{ \begin{array}{l}
 \text{ct}_{f,\mu}: \widetilde{\text{ict}}_1 (\emptyset \mid \emptyset) \\
 \{ \text{ict}_{j'}(\ell_{j'}, \mathbf{0}) \}_{1 < j' < j} \\
 \text{ict}_j (L_j / (\ell_j, \mathbf{0})) \\
 \{ \text{ict}_{j'}(L_{j'}) \}_{j' > j} \\
 \text{sk}_{\mathbf{x}}: \widetilde{\text{isk}} (1, \mathbf{x} \mid \ell_1)
 \end{array} \right\} \Rightarrow \left\{ \begin{array}{l}
 \ell_1, \ell_j \text{ HARDWIRED} \\
 \text{ct}_{f,\mu}: \widetilde{\text{ict}}_1 (\emptyset \mid \emptyset) \\
 \{ \text{ict}_{j'}(\ell_{j'}, \mathbf{0}) \}_{1 < j' < j} \\
 \text{ict}_j (\emptyset \mid \emptyset) \\
 \{ \text{ict}_{j'}(L_{j'}) \}_{j' > j} \\
 \text{sk}_{\mathbf{x}}: \widetilde{\text{isk}} (1, \mathbf{x} \mid \ell_1, \ell_j)
 \end{array} \right\} \\
 \ell_2, \dots, \ell_{j-1}, \ell_j \xleftarrow{\$} \mathbb{Z}_p, \text{ SOLVE FOR } \ell_1 \qquad \qquad \qquad \ell_j = L_j(\mathbf{x}) \text{ OR } \ell_j \xleftarrow{\$} \mathbb{Z}_p
 \end{array}$$

During the proof, there are at most two simulated ciphertexts at any time, so it appears that we can just use a simulation-secure IPFE capable of simulating at most two ciphertexts. This is *not* the case. The tricky part is that the usual definition of simulation security in (\star) only requires the *real world* to be indistinguishable from *simulation*. However, in the step of simulating ℓ_j as random, we need to switch ict_j to simulation when $\widetilde{\text{ict}}_1$ is already simulated (and symmetrically, reverting ict_j back to encryption while keeping $\widetilde{\text{ict}}_1$ simulated). It is unclear whether this transition is indistinguishable just via simulation security, because the definition says nothing about the indistinguishability of simulating *one more* ciphertext when there is already one simulated ciphertext, i.e.,

$$(\widetilde{\text{mpk}}, \widetilde{\text{ict}}_1, \text{ict}_2, \{ \widetilde{\text{isk}}_j \}_j) \approx (\widetilde{\text{mpk}}, \widetilde{\text{ict}}_1, \widetilde{\text{ict}}_2, \{ \widetilde{\text{isk}}_j \}_j) ?$$

Note that when we want to simulate ℓ_j , the computation of ℓ_1 has complicated dependency on \mathbf{x} ,⁸ and we cannot hope to get around the issue by first reverting ict_1 back to normal encryption then simultaneously simulating $\text{ict}_1, \text{ict}_j$, because we do not know what to encrypt in ict_1 .

Gradually Simulation-Secure IPFE. To solve the problem above, we define a stronger notion of simulation security, called *gradual simulation security*. It bridges the gap by capturing the idea that it is indistinguishable to simulate more ciphertexts even when some ciphertexts (and all the keys) are already simulated, as long as the total number of simulated ciphertexts does not exceed a preselected threshold. We show that the IPFE scheme in [3] can be adapted for gradual simulation security. The length of secret keys grows linearly in the maximum number of simulated ciphertexts, but not in the vector dimension. Plugging it into our CP-1-ABE construction, we obtain a CP-1-ABE with succinct keys.

⁸ In fact, the computation is as complex as the computation of $f(\mathbf{x})$.

We remark that another way to get around the issue of simulation security is to notice that there are at most two ciphertexts simulated at any time and one of them is ict_1 . Therefore, we can simply prepare two instances of IPFE (with independently generated master public and secret keys), one dedicated to ict_1 and the other to ict_j 's (for $j > 1$). During the proof, the instance for ict_1 is always simulated, and the other instance is switched between simulation and normal. The downside of this method is that using two instances doubles 1-ABE key size. In contrast, the solution using gradually simulation-secure IPFE only needs one more \mathbb{Z}_p element in CP-1-ABE key.

Comparison with Previous Techniques. Previous works constructing succinct ABE only natively support Boolean computations, whereas our method natively supports *arithmetic* computations. In [5–7, 22, 27], succinct ABE schemes are constructed from a special succinct ABE for set-membership policies (keys are tied to a set S and ciphertexts are tied to an element x ; decryption succeeds if $x \in S$). Based on ABE for set-membership policies, one can obtain ABE for monotone span programs, or policies admitting linear secret sharing schemes. Those ingredients (the special ABE, MSP, LSS) are inherently only native to Boolean computations. Among them, the work of [6] constructs succinct ABE for arithmetic span programs by reduction to MSP at the cost of a $\Theta(\log p)$ blow-up in key sizes.

In [23, 28], succinct ABE schemes are implicitly based on IPFE with succinct keys. The IPFE is only used to compute linear secret sharing schemes, and is used in a non-black-box way. In contrast, our 1-ABE can be constructed from any IPFE in a modular and black-box fashion, and we use it for arithmetic branching programs.

Dual System Encryption for Full ABE. To lift our CP-1-ABE to full KP-ABE, we need to flip the position of attributes and policies. Our idea is to use CP-1-ABE as a key encapsulation mechanism. More specifically, a KP-ABE key for policy f is a CP-1-ABE ciphertext $\text{cpct}(f, \mu)$, where μ is the message in CP-1-ABE and encapsulated key in KP-ABE. A KP-ABE ciphertext for attribute \mathbf{x} and message m consists of a CP-1-ABE key $\text{cpsk}(\mathbf{x})$ and the masked message $\mu + m$. If decryption is authorized, CP-1-ABE decryption will give us μ , which can be used to unmask the message. Observe that the security of KP-ABE aligns with the security of CP-1-ABE, namely, in the KP-ABE security game:

- We only need to handle one ciphertext, for which we rely on 1-key security of CP-1-ABE.
- We need to handle multiple keys, which corresponds to multi-ciphertext security of CP-1-ABE. Since our CP-1-ABE is public-key, it indeed satisfies multi-ciphertext security given only one key.

However, we need to resolve the issue that encryption of KP-ABE is now secret-key, since we need to know both the master secret key of CP-1-ABE and μ (part of the master secret key of KP-ABE) to generate KP-ABE ciphertext.

We observe that our CP-1-ABE is linear, i.e., the spaces of cpmsk , cpsk , cpct , messages are vector spaces over \mathbb{Z}_p , and⁹

$$\begin{aligned} k_1\text{cpsk}(\text{cpmsk}_1, \mathbf{x}) + k_2\text{cpsk}(\text{cpmsk}_2, \mathbf{x}) &= \text{cpsk}(k_1\text{cpmsk}_1 + k_2\text{cpmsk}_2, \mathbf{x}), \\ k_1\text{cpct}(\text{cpmsk}_1, f, \mu_1) + k_2\text{cpct}(\text{cpmsk}_1, f, \mu_2) &= \text{cpct}(k_1\text{cpmsk}_1 + k_2\text{cpmsk}_2, f, k_1\mu_1 + k_2\mu_2). \end{aligned}$$

Here, $\text{cpsk}(\text{cpmsk}, \mathbf{x})$ and $\text{cpct}(\text{cpmsk}, f, \mu)$ represent that they are generated in the CP-1-ABE instance whose master secret key is cpmsk . We instantiate our CP-1-ABE with an IPFE such that the keys are linear in the master secret key and the ciphertexts are linear in both the master secret key and the encrypted vector. CP-1-ABE master secret key and keys are IPFE master secret key and keys, so cpsk 's are linear in cpmsk . CP-1-ABE ciphertexts are IPFE ciphertexts for the label functions of AKGS, and AKGS is linear with respect to the message μ , so cpct 's are linear in msk, μ .

Let G be an additive prime-order group generated by P and write $\llbracket a \rrbracket = aP$. Concretely, cpmsk and cpsk 's will be \mathbb{Z}_p elements. Now if we encode cpmsk in G , by linearity we can compute cpsk in G , and we denote this fact by

$$\llbracket \text{cpsk}(\text{cpmsk}, \mathbf{x}) \rrbracket = \text{cpsk}(\llbracket \text{cpmsk} \rrbracket, \mathbf{x}).$$

Assume for the moment that this can also be done for cpct 's and decryption still works.¹⁰ Given the linearity, we can employ dual system encryption [24] to make the scheme public-key. In prime-order groups, the classic dual system encryption can be regarded as hash proof systems based on MDDH_k [9, 12].¹¹

Take MDDH_1 (DDH assumption) for example. KP-ABE prepares two instances of CP-1-ABE and two messages, and publishes the projection of them along a randomly sampled vector (b_1, b_2) in the exponent:

$$\begin{aligned} \text{kpmpk} &= \llbracket b_1, b_2, b_1\text{cpmsk}_1 + b_2\text{cpmsk}_2, b_1\mu_1 + b_2\mu_2 \rrbracket \quad \text{for } b_1, b_2 \xleftarrow{\$} \mathbb{Z}_p, \\ \text{kpsk} &= (\text{cpmpk}_1, \text{cpmpk}_2, \text{cpmsk}_1, \text{cpmsk}_2, \mu_1, \mu_2). \end{aligned}$$

Encryption is now public-key. A KP-ABE ciphertext simply uses a random CP-1-ABE master secret key in the projected space (a.k.a. *normal* space in dual system encryption) and use the projected μ to mask the message. A KP-ABE key consists of two CP-1-ABE ciphertexts, one in each instance encrypting the corresponding encapsulated key.

$$\text{kpct}(\mathbf{x}, m) = (s\llbracket b_1, b_2 \rrbracket, \text{cpsk}(s\llbracket b_1\text{cpmsk}_1 + b_2\text{cpmsk}_2 \rrbracket, \mathbf{x}), m + s\llbracket b_1\mu_1 + b_2\mu_2 \rrbracket) \quad \text{for } s \xleftarrow{\$} \mathbb{Z}_p,$$

$$\text{kpsk}(f) = (\text{cpct}(\text{cpmsk}_1, f, \mu_1), \text{cpct}(\text{cpmsk}_2, f, \mu_2)).$$

⁹ The randomness in key generation/encryption should also take part in the linear homomorphism, but we omit it in this overview for brevity.

¹⁰ In our case, cpct 's are already group-encoded, and this is where pairing comes in.

¹¹ A few examples are [3, 13, 15, 26]. Wee [25] also notices that certain usage of dual system encryption in composite-order groups is reminiscent of hash proof systems. There are other ways to use dual system encryption that are not captured by hash proof systems.

To decrypt, we first use linearity to combine the two CP-1-ABE ciphertexts into

$$\begin{aligned} & \text{cpct}(\llbracket sb_1 \text{cpmsk}_1 + sb_2 \text{cpmsk}_2 \rrbracket, f, \llbracket sb_1 \mu_1 + sb_2 \mu_2 \rrbracket) \\ &= \llbracket sb_1 \rrbracket \text{cpct}(\text{cpmsk}_1, f, \mu_1) + \llbracket sb_2 \rrbracket \text{cpct}(\text{cpmsk}_2, f, \mu_2). \end{aligned}$$

The master secret key of the combined cpct matches that of the cpsk in the KP-ABE ciphertext, and CP-1-ABE decryption will recover $\llbracket sb_1 \mu_1 + sb_2 \mu_2 \rrbracket$, using which we can unmask to obtain the message m .

To argue security, we first replace $\llbracket sb_1, sb_2 \rrbracket$ used in the challenge ciphertext by $\llbracket a_1, a_2 \rrbracket$ for random $a_1, a_2 \xleftarrow{\$} \mathbb{Z}_p$ (using DDH), which is not co-linear with (b_1, b_2) with overwhelming probability. Ciphertexts in this form are said to be *semi-functional* in dual system encryption.

By the linearity, we can look at the ABE scheme from a new basis, namely $(b_1, b_2), (a_1, a_2)$. We denote the CP-1-ABE components and μ 's in this basis with prime, e.g., $\text{cpmsk}'_1 = b_1 \text{cpmsk}_1 + b_2 \text{cpmsk}_2$ and $\text{cpmsk}'_2 = a_1 \text{cpmsk}_1 + a_2 \text{cpmsk}_2$. The KP-ABE master public key reveals cpmsk'_1 but not cpmsk'_2 . A KP-ABE secret key for policy f is essentially $\text{cpct}(\text{cpmsk}'_1, f, \mu'_1)$ and $\text{cpct}(\text{cpmsk}'_2, f, \mu'_2)$. The challenge ciphertext has $\text{cpsk}(\text{cpmsk}'_2, \mathbf{x})$, and the message is masked by μ'_2 . By CP-1-ABE security, μ'_2 (in cpct's) should be hidden, which means the message in the challenge ciphertext is hidden by μ'_2 .

The proof completes by replacing μ'_2 in all the KP-ABE keys by random. ABE keys in this form are said to be *semi-functional* in dual system encryption.

Lastly, to base the scheme on MDDH $_k$, we use $k + 1$ instances of CP-1-ABE, publish a k -dimensional projection (*normal space*), and reserve the unpublished dimension for the security proof (*semi-functional space*).

CP-ABE from KP-1-ABE. By symmetry, we can apply the transformation to obtain CP-ABE from KP-1-ABE. Moreover, our KP-ABE trivially serves as a KP-1-ABE. Therefore, the scheme is (ignoring group encoding)

$$\begin{aligned} \text{cpmpk} &= (d_1, d_2, d_1 \text{kpmsk}_1 + d_2 \text{kpmsk}_2, d_1 \nu_1 + d_2 \nu_2) \quad \text{for } d_1, d_2 \xleftarrow{\$} \mathbb{Z}_p, \\ \text{cpmsk} &= (\text{kpmpk}_1, \text{kpmpk}_2, \text{kpmsk}_1, \text{kpmsk}_2, \nu_1, \nu_2), \\ \text{cpsk} &= (\text{kpct}(\text{kpmsk}_1, \mathbf{x}, \nu_1), \text{kpct}(\text{kpmsk}_2, \mathbf{x}, \nu_2)), \\ \text{cpct} &= (td_1, td_2, \text{kpsk}(t(d_1 \text{msk}_1 + d_2 \text{msk}_2), f), m + t(d_1 \nu_1 + d_2 \nu_2)) \quad \text{for } t \xleftarrow{\$} \mathbb{Z}_p. \end{aligned}$$

Again, KP-1-ABE is used to encapsulate keys ν_1, ν_2 , whose projection masks the message in CP-ABE. Dual system encryption or hash proof system is used to obtain public-key encryption by publishing a random projection of KP-1-ABE master secret keys (in this case, along (d_1, d_2)).

One final observation is that only μ_1, μ_2 in KP-(1-)ABE need to be duplicated and projected, yielding only a small overhead in CP-ABE compared to KP-ABE. We leave the details to the full version [18].

We note that once we obtain KP-ABE from CP-1-ABE, going to CP-ABE using the same method is natural and simple.

1.2 Related Works

Succinct ABE. We compare our scheme with previous KP-ABE schemes with constant-size ciphertexts in Table 1 and CP-ABE schemes with constant-size secret keys in Table 2.

Compact ABE. Previous schemes achieving compactness (linear-size keys and ciphertexts, also known as “unbounded multi-use of attributes”) and adaptive security based on standard assumptions are [15, 23] for Boolean formulae, [13] for Boolean branching programs, and [17] for arithmetic branching programs. Among them, only [23] achieves succinctness.

ABE with Succinct f -Part. From pairing, we know several ABE schemes with succinct \mathbf{x} -part (ciphertexts in KP-ABE and keys in CP-ABE) and compact f -part (linear in the size of f), including ones in this work. One can also investigate succinctness in f -part (keys in KP-ABE and ciphertexts in CP-ABE). So far, the only schemes with succinct f -part are KP-ABE for polynomial-sized circuits based on LWE [8] and CP-ABE schemes for NC^1 based on LWE and pairing [4], in which the size of f -part depends on the depth but not the size of the circuit. Yet these schemes have compact but non-succinct \mathbf{x} -part.

Unbounded ABE. Our succinct ABE schemes have master public key of size linear in the attribute length. In general, one can further improve the size of master keys to be a constant, which requires the scheme to be able to handle attributes of any polynomial length. Such schemes are called unbounded ABE. So far, there are unbounded and compact ABE schemes (e.g., [15] for NC^1). It remains an interesting open problem to construct unbounded succinct schemes.

In summary, to the best of our knowledge, our schemes achieve one of the currently best trade-offs in terms of master key/secret key/ciphertext sizes.

2 Preliminaries

For two matrices \mathbf{A}, \mathbf{B} , their tensor product is denoted by $\mathbf{A} \otimes \mathbf{B}$. An affine function $f : \mathbb{Z}_p^n \rightarrow \mathbb{Z}_p$ over prime field \mathbb{Z}_p is conveniently associated with its coefficient vector $\mathbf{f} \in \mathbb{Z}_p^{n+1}$ (the same letter in boldface) such that $f(\mathbf{x}) = \mathbf{f}^\top \begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix}$.

2.1 Arithmetic Branching Programs and Arithmetic Key Garbling

In this paper, we consider the class of decryption policies defined by arithmetic branching programs [20].

Definition 1 (ABP). An arithmetic branching program (ABP) $f = (V, E, s, t, p, n, w)$ consists of a directed acyclic graph (V, E) , two distinguished vertices $s, t \in V$, a prime field order p , an arity n , and a weight function

$w : E \times \mathbb{Z}_p^n \rightarrow \mathbb{Z}_p$ that is affine in the second input. It computes the function $f : \mathbb{Z}_p^n \rightarrow \mathbb{Z}_p$ (written as the same letter) defined by

$$f(\mathbf{x}) = \sum_{\substack{s-t \text{ path} \\ e_1 \cdots e_i}} \prod_{j=1}^i w(e_j, \mathbf{x}).$$

Its size (denoted by $|f|$) is $|V|$. It induces two zero-test predicates:

$$f_{\neq 0}(\mathbf{x}) = \begin{cases} 0, & \text{if } f(\mathbf{x}) = 0; \\ 1, & \text{if } f(\mathbf{x}) \neq 0; \end{cases} \quad f_{=0}(\mathbf{x}) = \neg f_{\neq 0}(\mathbf{x}).$$

Denote by **ABP** (resp. ABP_p^n) the class of all ABPs (resp. of field order p and arity n), and by ztABP_p^n the set of zero-test predicates induced by ABPs in ABP_p^n .

We rely on an arithmetic key garbling scheme for ABP.

Definition 2 (AKGS). Let $\mathcal{F} = \{f\}$ be a class of functions $f : \mathbb{Z}_p^n \rightarrow \mathbb{Z}_p$. An arithmetic key garbling scheme (AKGS) for \mathcal{F} consists of two efficient algorithms:

- $\text{Garble}(f, \alpha, \beta; \mathbf{r})$ takes a function $f : \mathbb{Z}_p^n \rightarrow \mathbb{Z}_p \in \mathcal{F}$ and two secrets $\alpha, \beta \in \mathbb{Z}_p$ as input, and uses uniform randomness $\mathbf{r} \in \mathbb{Z}_p^{m'}$. It outputs coefficient vectors $\mathbf{L}_1, \dots, \mathbf{L}_m \in \mathbb{Z}_p^{n+1}$ of m affine functions $L_1, \dots, L_m : \mathbb{Z}_p^n \rightarrow \mathbb{Z}_p$ (called label functions). The vectors \mathbf{L}_j are linear in $(\alpha, \beta, \mathbf{r})$. The amount of randomness m' and the number m of label functions are solely determined by f , and m is called the garbling size of f .
- $\text{Eval}(f, \mathbf{x}, \ell_1, \dots, \ell_m)$ takes as input a function $f : \mathbb{Z}_p^n \rightarrow \mathbb{Z}_p \in \mathcal{F}$, an input $\mathbf{x} \in \mathbb{Z}_p^n$, and m labels $\ell_1, \dots, \ell_m \in \mathbb{Z}_p$. It outputs $\gamma \in \mathbb{Z}_p$ that is linear in ℓ_1, \dots, ℓ_m .

The scheme is required to be correct, i.e., for all $f : \mathbb{Z}_p^n \rightarrow \mathbb{Z}_p \in \mathcal{F}, \alpha, \beta \in \mathbb{Z}_p, \mathbf{x} \in \mathbb{Z}_p^n$, it holds that

$$\Pr \left[\begin{array}{l} (\mathbf{L}_1, \dots, \mathbf{L}_m) \stackrel{\$}{\leftarrow} \text{Garble}(f, \alpha, \beta) \\ \forall j \in [m], \ell_j \leftarrow L_j(\mathbf{x}) \end{array} : \text{Eval}(f, \mathbf{x}, \ell_1, \dots, \ell_m) = \alpha f(\mathbf{x}) + \beta \right] = 1.$$

We rely on the strong notion of piecewise security recently introduced in [17].

Definition 3 (piecewise security). Let $(\text{Garble}, \text{Eval})$ be an AKGS for some function class \mathcal{F} . The scheme is piecewise secure if it satisfies the following two properties:

- The first label is reversely sampleable given the input, the output, and the other labels. That is, there is an efficient algorithm RevSamp such that for all $f : \mathbb{Z}_p^n \rightarrow \mathbb{Z}_p \in \mathcal{F}, \alpha, \beta \in \mathbb{Z}_p, \mathbf{x} \in \mathbb{Z}_p^n$, the following distributions are identical:

$$\begin{aligned} & \left\{ \begin{array}{l} (\mathbf{L}_1, \dots, \mathbf{L}_m) \stackrel{\$}{\leftarrow} \text{Garble}(f, \alpha, \beta) \\ \ell_1 \leftarrow L_1(\mathbf{x}) \end{array} : (\ell_1, \mathbf{L}_2, \dots, \mathbf{L}_m) \right\} \\ \equiv & \left\{ \begin{array}{l} (\mathbf{L}_1, \dots, \mathbf{L}_m) \stackrel{\$}{\leftarrow} \text{Garble}(f, \alpha, \beta) \\ \ell_j \leftarrow L_j(\mathbf{x}) \text{ for } j \in [m], j > 1 \\ \ell_1 \leftarrow \text{RevSamp}(f, \mathbf{x}, \alpha f(\mathbf{x}) + \beta, \ell_2, \dots, \ell_m) \end{array} : (\ell_1, \mathbf{L}_2, \dots, \mathbf{L}_m) \right\}. \end{aligned}$$

– The other labels are marginally random even given all the subsequent label functions. That is, for all $f : \mathbb{Z}_p^n \rightarrow \mathbb{Z}_p \in \mathcal{F}, \alpha, \beta \in \mathbb{Z}_p, \mathbf{x} \in \mathbb{Z}_p^n$, suppose the garbling size of f is m , then for all $j \in [m], j > 1$, the following distributions are identical:

$$\begin{aligned} & \left\{ \begin{array}{l} (\mathbf{L}_1, \dots, \mathbf{L}_m) \stackrel{\$}{\leftarrow} \text{Garble}(f, \alpha, \beta) \\ \ell_j \leftarrow L_j(\mathbf{x}) \end{array} : (\ell_j, \mathbf{L}_{j+1}, \dots, \mathbf{L}_m) \right\} \\ \equiv & \left\{ \begin{array}{l} (\mathbf{L}_1, \dots, \mathbf{L}_m) \stackrel{\$}{\leftarrow} \text{Garble}(f, \alpha, \beta) \\ \ell_j \stackrel{\$}{\leftarrow} \mathbb{Z}_p \end{array} : (\ell_j, \mathbf{L}_{j+1}, \dots, \mathbf{L}_m) \right\}. \end{aligned}$$

A piecewise secure AKGS is known for ABPs:

Lemma 4 ([14,17]). *There exists a piecewise secure AKGS for ABP, for which the garbling size of an ABP is the same as its size.*

Throughout the paper, we will use a vectorized version of the garbling algorithm. Let $\alpha, \beta \in \mathbb{Z}_p^k$, then $\text{Garble}(f, \alpha, \beta)$ is executed component-wise with independent randomness and the output are concatenated:

$$\begin{aligned} \text{for } t \in [k]: & \quad (\mathbf{L}_1^{(t)}, \dots, \mathbf{L}_m^{(t)}) \stackrel{\$}{\leftarrow} \text{Garble}(f, \alpha[t], \beta[t]); \\ \text{for } j \in [m]: & \quad \mathbf{L}_j = \begin{pmatrix} \mathbf{L}_j^{(1)} \\ \vdots \\ \mathbf{L}_j^{(k)} \end{pmatrix} = \sum_{t=1}^k \mathbf{e}_j \otimes \mathbf{L}_j^{(t)}; \\ & \quad \text{output}(\mathbf{L}_1, \dots, \mathbf{L}_m). \end{aligned}$$

Here, $\mathbf{e}_j \in \mathbb{Z}_p^k$ are the standard basis vectors and \mathbf{L}_j 's are column vectors of length $k(n+1)$. In the vectorized version, the randomness is a matrix and each row of the matrix is used for one invocation of the non-vectorized garbling. This notation is compatible with tensor products:

Lemma 5 (mixing and stitching). *Suppose $f : \mathbb{Z}_p^n \rightarrow \mathbb{Z}_p$.*

Let $\alpha, \beta \in \mathbb{Z}_p^k, \mathbf{R} \in \mathbb{Z}_p^{k \times m'}, \mathbf{c} \in \mathbb{Z}_p^k$, and define

$$(\mathbf{L}_1, \dots, \mathbf{L}_m) \leftarrow \text{Garble}(f, \alpha, \beta; \mathbf{R}), \quad (\mathbf{L}'_1, \dots, \mathbf{L}'_{m'}) \leftarrow \text{Garble}(f, \mathbf{c}^\top \alpha, \mathbf{c}^\top \beta; \mathbf{c}^\top \mathbf{R}),$$

then $\mathbf{L}_j^\top (\mathbf{c} \otimes \mathbf{I}_{n+1}) = (\mathbf{L}'_j)^\top$ for all $j \in [m]$.

Now let $\alpha, \beta \in \mathbb{Z}_p, \mathbf{r} \in \mathbb{Z}_p^{m'}, \mathbf{d} \in \mathbb{Z}_p^k$, and define

$$(\mathbf{L}'_1, \dots, \mathbf{L}'_{j'}) \leftarrow \text{Garble}(f, \alpha, \beta; \mathbf{r}), \quad (\mathbf{L}_1, \dots, \mathbf{L}_j) \leftarrow \text{Garble}(f, \alpha \mathbf{d}, \beta \mathbf{d}; \mathbf{d} \mathbf{r}^\top),$$

then $\mathbf{d} \otimes \mathbf{L}'_j = \mathbf{L}_j$ for all $j \in [m]$.

2.2 Attribute-Based Encryption

In the definition below, we explicitly take the description of policy/attribute out of the secret key/ciphertext so that we can characterize succinctness.

Definition 6 (ABE). *Let $\mathcal{M} = \{M_\lambda\}_{\lambda \in \mathbb{N}}$ be a sequence of message sets and $\mathcal{P} = \{\mathcal{P}_\lambda\}_{\lambda \in \mathbb{N}}$ a sequence of predicate families with $\mathcal{P}_\lambda = \{P : X_P \times Y_P \rightarrow \{0, 1\}\}$. An attribute-based encryption (ABE) scheme for message space \mathcal{M} and predicate space \mathcal{P} consists of four efficient algorithms:*

- $\text{Setup}(1^\lambda, P)$ takes as input the security parameter 1^λ and a predicate $P \in \mathcal{P}_\lambda$, and outputs a pair of master public/secret keys (mpk, msk) .
- $\text{KeyGen}(\text{msk}, y)$ takes as input a policy $y \in Y_P$ and outputs a secret key sk .
- $\text{Enc}(\text{mpk}, x, g)$ takes as input an attribute $x \in X_P$ and a message $g \in M_\lambda$, and outputs a ciphertext ct .
- $\text{Dec}(\text{sk}, y, \text{ct}, x)$ takes as input a secret key, the policy of the key, a ciphertext, and the attribute of the ciphertext, and is supposed to recover the message if $P(x, y) = 1$.

The scheme is required to be correct, i.e., for all $\lambda \in \mathbb{N}, g \in M_\lambda, P \in \mathcal{P}_\lambda, x \in X_P, y \in Y_P$ such that $P(x, y) = 1$,

$$\Pr \left[\begin{array}{l} (\text{mpk}, \text{msk}) \stackrel{\$}{\leftarrow} \text{Setup}(1^\lambda, P) \\ \text{sk} \stackrel{\$}{\leftarrow} \text{KeyGen}(\text{msk}, y) : \text{Dec}(\text{sk}, y, \text{ct}, x) = g \\ \text{ct} \stackrel{\$}{\leftarrow} \text{Enc}(\text{mpk}, x, g) \end{array} \right] = 1.$$

Definition 7 (ABE for ABP). *Let $p = p(\lambda)$ be a sequence of prime numbers. A key-policy ABE (KP-ABE) for ABP over $\mathbb{Z}_{p(\lambda)}$ is defined for the following predicate family:*

$$\mathcal{P} = \{\mathcal{P}_\lambda\}, \quad \mathcal{P}_\lambda = \{P_{\lambda,n} : \mathbb{Z}_{p(\lambda)}^n \times \text{ztABP}_{p(\lambda)}^n \rightarrow \{0, 1\}\}, \quad P_{\lambda,n}(\mathbf{x}, y) = y(\mathbf{x}).$$

In a ciphertext-policy ABE (CP-ABE) for ABP over $\mathbb{Z}_{p(\lambda)}$, the predicates are

$$P_{\lambda,n} : \text{ztABP}_{p(\lambda)}^n \times \mathbb{Z}_{p(\lambda)}^n \rightarrow \{0, 1\}, \quad (y, \mathbf{x}) \mapsto y(\mathbf{x}).$$

Definition 8 (succinct ABE). *An ABE scheme has succinct ciphertext if the length of ct is a fixed polynomial in security parameter λ (independent of the length of x, y and the choice of P). Similarly, the scheme has succinct secret key if the length of sk is a fixed polynomial in λ .*

The above definition does not rule out trivially succinct schemes, e.g., one only supporting x, y of length at most λ . In this work, we construct KP-ABE for ABP with succinct ciphertexts and CP-ABE for ABP with succinct secret keys. These constructions are non-trivial because Setup can be run with any predicate $P_{\lambda,n}$ for attribute length n , the scheme works with policies of arbitrary size, and the ciphertexts in KP-ABE and the secret keys in CP-ABE have fixed size $\text{poly}(\lambda)$, independent of n .

Security. We consider the standard IND-CPA security of ABE.

Definition 9 (IND-CPA of ABE [16]). *Adopt the notations in Definition 6. The scheme is IND-CPA secure if $\text{Exp}_{CPA}^0 \approx \text{Exp}_{CPA}^1$, where Exp_{CPA}^b with adversary \mathcal{A} proceeds as follows:*

- **Setup.** Launch $\mathcal{A}(1^\lambda)$ and receive from it a predicate $P \in \mathcal{P}_\lambda$. Run $(\text{mpk}, \text{msk}) \stackrel{\$}{\leftarrow} \text{Setup}(1^\lambda, P)$ and send mpk to \mathcal{A} .
- **Query I.** Repeat the following for arbitrarily many rounds determined by \mathcal{A} : In each round, \mathcal{A} submits a policy $y_q \in Y_P$ for a secret key. Upon this query, run $\text{sk}_q \stackrel{\$}{\leftarrow} \text{KeyGen}(\text{msk}, y)$ and send sk_q to \mathcal{A} .
- **Challenge.** The adversary submits the challenge attribute $x^* \in X_P$ and two messages $g_0, g_1 \in M_\lambda$. Run $\text{ct} \stackrel{\$}{\leftarrow} \text{Enc}(\text{mpk}, x, g_b)$ and return ct to \mathcal{A} .
- **Query II.** Same as Query I.
- **Guess.** The adversary outputs a bit b' . The outcome of the experiment is b' if $P(x^*, y_q) = 0$ for all y_q queried in Query I/II. Otherwise, the outcome is set to 0.

2.3 Pairing Groups and Matrix Diffie–Hellman Assumption

Throughout the paper, we use a sequence of pairing groups

$$\mathcal{G} = \{(G_{\lambda,1}, G_{\lambda,2}, G_{\lambda,T}, g_{\lambda,1}, g_{\lambda,2}, e_\lambda)\}_{\lambda \in \mathbb{N}},$$

where $G_{\lambda,1}, G_{\lambda,2}, G_{\lambda,T}$ are groups of prime order $p = p(\lambda)$, and $G_{\lambda,1}$ (resp. $G_{\lambda,2}$) is generated by $g_{\lambda,1}$ (resp. $g_{\lambda,2}$). The maps $e_\lambda : G_{\lambda,1} \times G_{\lambda,2} \rightarrow G_{\lambda,T}$ are

- *bilinear:* $e_\lambda(g_{\lambda,1}^a, g_{\lambda,2}^b) = (e_\lambda(g_{\lambda,1}, g_{\lambda,2}))^{ab}$ for all $a, b \in \mathbb{Z}_{p(\lambda)}$; and
- *non-degenerate:* $g_{\lambda,T} \stackrel{\text{def}}{=} e_\lambda(g_{\lambda,1}, g_{\lambda,2})$ generates $G_{\lambda,T}$.

The group operations as well as the pairing e_λ must be efficiently computable.

When we talk about one group without thinking about pairing, the subscripts 1, 2, T are dropped.

Bracket Notation. Fix a security parameter, for $i = 1, 2, T$, we write $[\mathbf{A}]_i$ for $g_{\lambda,i}^{\mathbf{A}}$, where the exponentiation is element-wise. When bracket notation is used, group operations are written additively and pairing is written multiplicatively, so that $[\mathbf{A}]_i + [\mathbf{B}]_i = [\mathbf{A} + \mathbf{B}]_i$ and $[\mathbf{A}]_1 [\mathbf{B}]_2 = [\mathbf{A}]_2 [\mathbf{B}]_1 = [\mathbf{AB}]_T$. Furthermore, numbers can always operate with group elements, e.g., $\mathbf{A} [\mathbf{B}]_1 = [\mathbf{AB}]_1$.

Matrix Diffie–Hellman Assumption. In this paper, we rely on the MDDH assumptions.

Definition 10 (MDDH [12]). *Let $G = \{(G_\lambda, g_\lambda)\}_{\lambda \in \mathbb{N}}$ be a sequence of groups of prime order $p = p(\lambda)$ with their generators, and $\ell = \ell(\lambda), q = q(\lambda)$ polynomials. The $\text{MDDH}_{k,\ell}^q$ assumption holds in G if*

$$\{[\mathbf{A}, \mathbf{S}^\top \mathbf{A}]\}_{\lambda \in \mathbb{N}} \approx \{[\mathbf{A}, \mathbf{C}^\top]\}_{\lambda \in \mathbb{N}} \text{ for } \mathbf{A} \stackrel{\$}{\leftarrow} \mathbb{Z}_{p(\lambda)}^{k \times \ell(\lambda)}, \mathbf{S} \stackrel{\$}{\leftarrow} \mathbb{Z}_{p(\lambda)}^{k \times q(\lambda)}, \mathbf{C} \stackrel{\$}{\leftarrow} \mathbb{Z}_{p(\lambda)}^{\ell(\lambda) \times q(\lambda)}.$$

By default, $\ell = k + 1$ and $q = 1$. It is known [12] that k -Lin implies MDDH_k , which further implies $\text{MDDH}_{k,\ell}^q$ for any polynomial ℓ, q .

3 IPFE with Gradual Simulation Security

In this work, we consider IPFE schemes based on MDDH-hard groups (potentially without pairing), where the ciphertext encodes the encrypted vector in the exponent of the group, and decryption computes the inner product in the exponent. In our definition below, we directly define such group-based IPFE. The definition can be easily modified for IPFE that are not group-based.

Definition 11 (IPFE). *Let $G = \{(G_\lambda, g_\lambda)\}_{\lambda \in \mathbb{N}}$ be a sequence of groups of prime order $p = p(\lambda)$ with their generators. A G -encoded public-key inner-product functional encryption (IPFE) scheme consists of four efficient algorithms:*

- **Setup**($1^\lambda, 1^n, 1^T$) takes as input the security parameter 1^λ , the dimension 1^n of the vectors, and an additional parameter 1^T (see Definition 12). It outputs a pair of master public/secret keys (mpk, msk).
- **KeyGen**(msk, \mathbf{v}) takes the master secret key and a vector as input, and outputs a secret key sk.
- **Enc**(mpk, $\llbracket \mathbf{u} \rrbracket$) takes the master public key and a vector (encoded in G) as input, and outputs a ciphertext ct.
- **Dec**(sk, \mathbf{v} , ct) takes a secret key, the vector in the secret key, and a ciphertext as input, and is supposed to compute the inner product in the exponent.

The scheme is required to be correct, meaning that for all $\lambda, n, T \in \mathbb{N}$, $\mathbf{u}, \mathbf{v} \in \mathbb{Z}_{p(\lambda)}^n$, it holds that

$$\Pr \left[\begin{array}{l} (\text{mpk}, \text{msk}) \stackrel{\$}{\leftarrow} \text{Setup}(1^\lambda, 1^n, 1^T) \\ \text{sk} \stackrel{\$}{\leftarrow} \text{KeyGen}(\text{msk}, \mathbf{v}) \\ \text{ct} \stackrel{\$}{\leftarrow} \text{Enc}(\text{mpk}, \llbracket \mathbf{u} \rrbracket) \end{array} : \text{Dec}(\text{sk}, \mathbf{v}, \text{ct}) = \llbracket \mathbf{u}^T \mathbf{v} \rrbracket \right] = 1.$$

The scheme is succinct if the length of sk is independent of n and only depends on λ, T .

Setup algorithm in the above definition takes an additional input 1^T specifying the desired level of simulation security, which we define next.

Gradual Simulation Security. When building the 1-ABE scheme, we rely on the notion of gradual simulation security, which is stronger than the usual simulation security (see [2]). Roughly speaking, on top of the requirement that simulation should be indistinguishable from the real scheme, the notion stipulates that even when some ciphertexts are already simulated, whether another ciphertext is honest or simulated should be indistinguishable. The parameter T specifies the maximum number of ciphertexts that can be simulated.

To navigate around the many indices involved in the definition, it is the easiest to keep in mind that i (hence I, I^*, I_t) always counts the ciphertexts, and that j (hence J, J^*, J_t) always counts the keys.

Definition 12 (gradual simulation security). *Adopt the notations in Definition 11. A simulator consists of three efficient algorithms:*

- $\text{SimSetup}(1^\lambda, 1^n, 1^T)$ takes the same input as Setup , and outputs a simulated master public key mpk and an internal state st .¹²
- $\text{SimKeyGen}(\text{st}, \mathbf{v}, z_1, \dots, z_I)$ takes as input the internal state st , a vector \mathbf{v} , and a list z_1, \dots, z_I of inner products in $\mathbb{Z}_{p(\lambda)}$ (which are the intended inner products between this simulated key and all previously simulated ciphertexts). It outputs a simulated secret key sk and a new state st' .
- $\text{SimEnc}(\text{st}, z_1, \dots, z_J)$ takes as input the internal state st and a list z_1, \dots, z_J of inner products in $\mathbb{Z}_{p(\lambda)}$ (which are the intended inner products between this simulated ciphertext and all previously simulated keys). It outputs a simulated ciphertext ct and a new state st' .

The simulator gradually T -simulates the scheme if it satisfies both key simulation security and T -ciphertext simulation security defined below.

An IPFE scheme is gradually T -simulation-secure if it can be gradually T -simulated by some simulator. The scheme is gradually simulation-secure if there exists a simulator such that the simulator gradually T -simulates the scheme for all $T = \text{poly}(\lambda)$.

Key Simulation Security. Roughly speaking, this captures the idea that it is indistinguishable to interact with the real authority (who generates and distributes mpk and sk 's) versus the simulator issuing simulated mpk and sk 's (without simulating any ciphertext). We require $\text{Exp}_{\text{real}} \approx \text{Exp}_{\text{sim}}$, which proceed as follows when run with an adversary \mathcal{A} :

- **Setup.** Launch $\mathcal{A}(1^\lambda)$ and receive from it $(1^n, 1^T)$. Run

$$\begin{aligned} \text{in Exp}_{\text{real}}: & \quad (\text{mpk}, \text{msk}) \stackrel{\$}{\leftarrow} \text{Setup}(1^\lambda, 1^n, 1^T) \\ \text{in Exp}_{\text{sim}}: & \quad (\text{mpk}, \text{st}) \stackrel{\$}{\leftarrow} \text{SimSetup}(1^\lambda, 1^n, 1^T) \end{aligned}$$

and send mpk to \mathcal{A} .

- **Challenge.** Repeat the following for arbitrarily many rounds determined by \mathcal{A} : In each round, \mathcal{A} submits a vector \mathbf{v}_j . Upon this challenge, run

$$\begin{aligned} \text{in Exp}_{\text{real}}: & \quad \text{sk}_j \stackrel{\$}{\leftarrow} \text{KeyGen}(\text{msk}, \mathbf{v}_j) \\ \text{in Exp}_{\text{sim}}: & \quad (\text{sk}_j, \text{st}') \stackrel{\$}{\leftarrow} \text{SimKeyGen}(\text{st}, \mathbf{v}_j) \quad \text{st} \leftarrow \text{st}' \end{aligned}$$

and send sk_j to \mathcal{A} .

- **Guess.** The adversary outputs a bit b' , the outcome of the experiment.

We emphasize that there is no ciphertext challenge in the experiments. The adversary can generate ciphertexts on its own using mpk .

T -Ciphertext Simulation Security. Roughly speaking, this captures the idea that when interacting with the simulator, it is indistinguishable whether any subset of

¹² It is understood that the state is maintained by one instance of simulator, and except in definitions, its creation, persistence, and update are suppressed when there is no danger of ambiguity.

ciphertexts are normally generated or simulated, as long as at most T ciphertexts are simulated. In the experiments below, we denote by $z_{i,j} \in \mathbb{Z}_p$ the decryption outcome (inner product) between the j^{th} simulated secret key (ordered temporally among all queried secret keys) and the i^{th} simulated ciphertext (ordered temporally among all queried ciphertexts, excluding the challenge ciphertext). We also let I_t, J_t be the number of simulated ciphertexts (excluding the challenge ciphertext) and secret keys at any time t . $\text{Exp}_{T\text{-GS}}^b$ ($b \in \{0, 1\}$) with adversary \mathcal{A} proceeds as follows:

- **Setup.** Launch $\mathcal{A}(1^\lambda)$ and receive from it $(1^n, 1^T)$. Run

$$(\text{mpk}, \text{st}) \stackrel{\$}{\leftarrow} \text{SimSetup}(1^\lambda, 1^n, 1^T)$$

and send mpk to \mathcal{A} .

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

- *Key Simulation Query:* \mathcal{A} can submit a vector \mathbf{v}_j with a list $z_{\leq I_t, j}$ of inner products for a secret key sk_j . The list $z_{\leq I_t, j}$ consists of $z_{1,j}, \dots, z_{I_t, j}$, all the decryption outcomes between sk_j and the simulated ciphertexts queried up to this point. Upon this query, run

$$(\text{sk}_j, \text{st}') \stackrel{\$}{\leftarrow} \text{SimKeyGen}(\text{st}, \mathbf{v}_j, z_{1,j}, \dots, z_{I_t, j}) \quad \text{st} \leftarrow \text{st}'$$

and send sk_j to \mathcal{A} .

- *Ciphertext Simulation Query:* \mathcal{A} can submit a list $z_{i, \leq J_t}$ of inner products for a simulated ciphertext ct_i . The list $z_{i, \leq J_t}$ consists of $z_{i,1}, \dots, z_{i, J_t}$, all the decryption outcomes between ct_i and the simulated secret keys queried up to this point. Upon this query, run

$$(\text{ct}_i, \text{st}') \stackrel{\$}{\leftarrow} \text{SimEnc}(\text{st}, z_{i,1}, \dots, z_{i, J_t}) \quad \text{st} \leftarrow \text{st}'$$

and send ct_i to \mathcal{A} .

- **Challenge.** The adversary submits a vector \mathbf{u}^* . Upon the challenge, let the total number of secret key queries in Query I be J^* and the total number of ciphertext queries in Query I be I^* , run

$$b = 0: \quad \text{ct}^* \stackrel{\$}{\leftarrow} \text{Enc}(\text{mpk}, \llbracket \mathbf{u}^* \rrbracket)$$

$$b = 1: \quad \text{ct}^* \stackrel{\$}{\leftarrow} \text{SimEnc}(\text{st}, (\mathbf{u}^*)^\top \mathbf{v}_1, \dots, (\mathbf{u}^*)^\top \mathbf{v}_{J^*}) \quad \text{st} \leftarrow \text{st}'$$

and send ct^* to \mathcal{A} .

- **Query II.** Same as Query I, except that in $\text{Exp}_{T\text{-GS}}^1$, for each secret key query \mathbf{v}_j , we put $(\mathbf{u}^*)^\top \mathbf{v}_j$ immediately after $z_{I^*, j}$ in the argument list of SimKeyGen so that the simulator gets the correct list of inner products:

$$b = 0: \quad (\text{sk}_j, \text{st}') \stackrel{\$}{\leftarrow} \text{SimKeyGen}(\text{st}, \mathbf{v}_j, z_{1,j}, \dots, z_{I^*, j}, \quad z_{I^*+1, j}, \dots, z_{I_t, j});$$

$$b = 1: \quad (\text{sk}_j, \text{st}') \stackrel{\$}{\leftarrow} \text{SimKeyGen}(\text{st}, \mathbf{v}_j, z_{1,j}, \dots, z_{I^*, j}, (\mathbf{u}^*)^\top \mathbf{v}_j, z_{I^*+1, j}, \dots, z_{I_t, j});$$

$\text{st} \leftarrow \text{st}'$ (in either case).

- **Guess.** The adversary outputs a bit b' . The outcome of the experiment is b' if both constraints are satisfied:
 - the total number of ciphertext simulation queries in Query I/II is less than T ;
 - the equation $\{\mathbf{u}_i^\top \mathbf{v}_j = z_{i,j} \quad \forall i, j\}$ (about \mathbf{u}_i 's) has a solution.
 Otherwise, the outcome is set to 0.

Remarks. In $\text{Exp}_{T\text{-GS}}^1$, the challenge ciphertext ct^* is generated in the same way as the other simulated ciphertexts, and in Query II the inner products between sk_j and ct^* are appropriately positioned. From the simulator's perspective, there is no indication which ciphertext is the challenge ciphertext. This definition ensures that the simulator *cannot behave differently depending on whether a particular ciphertext is the challenge or not*, and simplifies the application of gradual simulation security in our construction of ABE.

Note that the simulator receives inner products $z_{i,j}$ in the clear and the adversary submits challenge \mathbf{u}^* in $\text{Exp}_{T\text{-GS}}^b$ in the clear, though the input to encryption and the output of decryption are group-encoded. This is necessary as otherwise, the simulator must solve discrete logarithm in G .

We note that when $T = 1$, gradual simulation security becomes the standard notion of simulation security. On the other hand, simulation security does not imply gradual simulation security. So this definition is a strict generalization of simulation security.

3.1 Construction of Gradually Simulation-Secure IPFE

The IPFE scheme in [3] has been proven simulation-secure [2]. We show that it can be adapted for gradual simulation security. The scheme has succinct keys, whose length grows linearly in T and polynomially in λ , and is independent of n , which eventually translates into the succinctness of our ABE scheme.

Construction 13 ([3]). The construction is described for a fixed value of λ , and λ is suppressed for brevity. Let G be a group (with generator g) of prime order p such that MDDH_k holds in G . Our G -encoded IPFE works as follows:

- $\text{Setup}(1^n, 1^T)$ takes as input the dimension n and the maximum number T of simulated ciphertexts. It samples $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_p^{k \times (k+T)}$, $\mathbf{W} \xleftarrow{\$} \mathbb{Z}_p^{(k+T) \times n}$ and outputs $\text{mpk} = \llbracket \mathbf{A}, \mathbf{AW} \rrbracket$, $\text{msk} = \mathbf{W}$.
- $\text{KeyGen}(\text{msk}, \mathbf{v})$ outputs $\text{sk} = \mathbf{W}\mathbf{v}$.
- $\text{Enc}(\text{mpk}, \llbracket \mathbf{u} \rrbracket)$ samples $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_p^k$ and outputs $\text{ct} = (\mathbf{s}^\top \llbracket \mathbf{A} \rrbracket, \mathbf{s}^\top \llbracket \mathbf{AW} \rrbracket + \llbracket \mathbf{u}^\top \rrbracket)$.
- $\text{Dec}(\text{sk}, \mathbf{v}, \text{ct})$ parses ct as $(\llbracket \mathbf{c}^\top \rrbracket, \llbracket \mathbf{t}^\top \rrbracket)$ and outputs $-\llbracket \mathbf{c}^\top \rrbracket \text{sk} + \llbracket \mathbf{t}^\top \rrbracket \mathbf{v}$.

The correctness is readily verified by

$$-\llbracket \mathbf{c}^\top \rrbracket \text{sk} + \llbracket \mathbf{t}^\top \rrbracket \mathbf{v} = \llbracket -(\mathbf{s}^\top \mathbf{A})(\mathbf{W}\mathbf{v}) + (\mathbf{s}^\top \mathbf{AW} + \mathbf{u}^\top) \mathbf{v} \rrbracket = \llbracket \mathbf{u}^\top \mathbf{v} \rrbracket.$$

The scheme is succinct as sk consists of $k + T$ elements in \mathbb{Z}_p , independent of n .

Theorem 14. *Suppose in Construction 13, the $MDDH_k$ assumption holds in G , then the constructed scheme is gradually simulation-secure, and the T in the security definition is the T as input of Setup.*

The simulator for our scheme is built modularly upon that for the one-time pad IPFE scheme, which we sketch below. We refer the readers to the full version for a more detailed exposition.

One-Time Pad IPFE. OTP-IPFE is a secret-key IPFE:

- Setup($p, 1^n$) samples the master secret key $\text{msk} = \mathbf{w} \xleftarrow{\$} \mathbb{Z}_p^n$.
- KeyGen(msk, \mathbf{v}) outputs the secret key $\text{sk} = \mathbf{w}^\top \mathbf{v}$ for $\mathbf{v} \in \mathbb{Z}_p^n$.
- Enc(msk, \mathbf{u}) outputs the ciphertext $\text{ct} = (\mathbf{w} + \mathbf{u})^\top$.
- Dec($\text{sk}, \mathbf{v}, \text{ct}$) outputs $-\text{sk} + \text{ctv}$ as the inner product.

Correctness is readily verified by $-\text{sk} + \text{ctv} = -\mathbf{w}^\top \mathbf{v} + (\mathbf{w} + \mathbf{u})^\top \mathbf{v} = \mathbf{u}^\top \mathbf{v}$.

The scheme satisfies perfect simulation security for one ciphertext (defined similarly to the usual simulation security). The simulator works as follows:

- SimSetup($p, 1^n$) samples the internal state as $\text{st} = (\tilde{\mathbf{w}}, \perp)$ with $\tilde{\mathbf{w}} \xleftarrow{\$} \mathbb{Z}_p^n$. (Here, \perp means that the ciphertext has not been simulated.)
- SimKeyGen(st, \mathbf{v}_j) simulates a *pre*-challenge key for \mathbf{v}_j as $\text{sk}_j = \tilde{\mathbf{w}}^\top \mathbf{v}_j$ and updates the state to $\text{st}' = (\text{st}, \mathbf{v}_j)$.
- SimEnc($\text{st}, z_1, \dots, z_{J^*}$) simulates the challenge ciphertext as a uniformly random solution ct^* of

$$-\tilde{\mathbf{w}}^\top \mathbf{v}_j + \text{ct}^* \mathbf{v}_j = z_j \quad \forall j \in [J^*],$$

and updates the state to $\text{st}' = (\perp, \text{ct}^*)$ so that it knows the ciphertext has been simulated. (Here, J^* is the number of keys queried before ciphertext simulation, and z_j is the intended inner product between the ciphertext and the j^{th} key.)

- SimKeyGen($\text{st}, \mathbf{v}_j, z_j$) simulates a *post*-challenge key for \mathbf{v}_j as $\text{sk}_j = \text{ct}^* \mathbf{v}_j - z_j$ and does not update the state.

Dual System Encryption and Simulator. Construction 13 can be seen as dual system encryption applied to OTP-IPFE. There are $k + T$ instances of OTP-IPFE with the master secret keys being $\mathbf{W} \in \mathbb{Z}_p^{(k+T) \times n}$. We publish k projections of them (the *normal* space) in the master public key (i.e., \mathbf{AW} with $\mathbf{A} \in \mathbb{Z}_p^{k \times (k+T)}$), and reserve T instances (the *semi-functional* space) for the simulator.

To simulate, we first switch the ciphertext into the semi-functional form, which means it uses an OTP-IPFE instance independent of the master public key. Then, we employ a change of variable to explicitly separate out *the* instance (also known as using the *parameter hiding* property). Lastly, the simulation can be delegated to OTP-IPFE simulator.

Let us take $T = 1$ for example. The first step is

$$\text{ct}^* = (\llbracket \mathbf{s}^\top \mathbf{A} \rrbracket, \llbracket \mathbf{s}^\top \mathbf{A} \mathbf{W} + \mathbf{u}^\top \rrbracket) \stackrel{\text{MDDH}_k}{\approx} (\llbracket \mathbf{c}^\top \rrbracket, \llbracket \mathbf{c}^\top \mathbf{W} + \mathbf{u}^\top \rrbracket) \quad \text{for } \mathbf{c} \xleftarrow{\$} \mathbb{Z}_p^{k+1}.$$

The second step is to perform a change of variable $\mathbf{W} = \widetilde{\mathbf{W}} + \mathbf{a}^\perp \mathbf{w}^\top$, where \mathbf{W}, \mathbf{w} are random and \mathbf{a}^\perp is the vector such that $\mathbf{A} \mathbf{a}^\perp = \mathbf{0}$ and $\mathbf{c}^\top \mathbf{a}^\perp = 1$ (which uniquely exists with overwhelming probability). With this change of variable, the keys and the challenge ciphertext become

$$\begin{aligned} \text{mpk} &= (\llbracket \mathbf{A} \rrbracket, \llbracket \mathbf{A} \widetilde{\mathbf{W}} \rrbracket), \\ \text{sk}_j &= \widetilde{\mathbf{W}} \mathbf{v}_j + \mathbf{a}^\perp \boxed{\mathbf{w}^\top \mathbf{v}_j}, \\ \text{ct}^* &= (\llbracket \mathbf{c}^\top \rrbracket, \llbracket \mathbf{c}^\top \widetilde{\mathbf{W}} + \boxed{(\mathbf{w} + \mathbf{u})^\top} \rrbracket). \end{aligned}$$

The terms highlighted in the boxes are exactly the keys and ciphertexts of OTP-IPFE with master secret key \mathbf{w} , and the last step is to use OTP-IPFE simulator to simulate these terms.

For general T , we simply prepare uniformly random $\mathbf{c}_1, \dots, \mathbf{c}_T$ for each simulated ciphertext, and set $\mathbf{W} = \widetilde{\mathbf{W}} + \mathbf{a}_1^\perp \mathbf{w}_1^\top + \dots + \mathbf{a}_T^\perp \mathbf{w}_T^\top$, where $\mathbf{a}_1^\perp, \dots, \mathbf{a}_T^\perp$ are the solution to $\mathbf{A} \mathbf{a}_i^\perp = \mathbf{0}$, $\mathbf{c}_i^\top \mathbf{a}_i^\perp = 1$, and $\mathbf{c}_i^\top \mathbf{a}_{i'}^\perp = 0$ for all $i \neq i'$, so that the T instances for simulation do not “interfere” with each other. The keys and ciphertexts after replacing OTP-IPFE by simulation are

$$\begin{aligned} \text{mpk} &= (\llbracket \mathbf{A} \rrbracket, \llbracket \mathbf{A} \widetilde{\mathbf{W}} \rrbracket), \\ \text{sk}_j &= \widetilde{\mathbf{W}} \mathbf{v}_j + \mathbf{a}_1^\perp \text{SimKeyGen}(\text{st}_1, \mathbf{v}_j, z_{1,j}) + \dots + \mathbf{a}_T^\perp \text{SimKeyGen}(\text{st}_T, \mathbf{v}_j, z_{T,j}), \\ \text{ct}_i^* &= (\llbracket \mathbf{c}_i^\top \rrbracket, \llbracket \mathbf{c}_i^\top \widetilde{\mathbf{W}} + \text{SimEnc}(\text{st}_i, z_{i,1}, \dots, z_{i,J_t}) \rrbracket), \end{aligned}$$

which is how our simulator for Construction 13 works. Here, $\text{st}_1, \dots, \text{st}_T$ track T independent instances of OTP-IPFE simulator. We refer the reader to the full version for the security proof.

4 Ciphertext-Policy 1-ABE for ABP

In this section, we construct the core component of our adaptively secure ABE, called *1-ABE*, from any gradually 2-simulation-secure IPFE. A 1-ABE has the same syntax as an ABE, except that

- The message space is \mathbb{Z}_p for some p and decryption only needs to recover the message encoded in (another) group.
- In the security definition, the adversary is allowed to query at most one secret key.
- In the security definition, the adversary only chooses the attribute but not the message. The message is 0 in one experiment ($\text{Exp}_{1\text{-ABE}}^0$), and is uniformly random in the other experiment ($\text{Exp}_{1\text{-ABE}}^1$).¹³

¹³ The adversary also does not receive the potential random message.

The relaxation of decryption correctness and the change of messages in the security definition are because 1-ABE will be used to encapsulate keys for full ABE. In full ABE, the group-encoded decryption result of 1-ABE is used to mask the message, and we argue security by replacing the encapsulated key by random.

ABE constructions in some previous works such as [15,17] go through an intermediate step of building a *secret-key* 1-ABE that is 1-key *1-ciphertext* secure. In the secret-key setting, keys and ciphertexts are symmetric, and consequently there is no distinction between ciphertext-policy and key-policy 1-ABE. In contrast, our 1-ABE is *public-key* and *1-key* secure. This asymmetry separates CP-1-ABE and KP-1-ABE. We remark that 1-ABE in [15,17] can be easily modified to fit our definition as CP-1-ABE. We will see that our definition is easier to use in reductions for full ABE.

Construction 15 (CP-1-ABE). The construction is described for a fixed value of λ , and λ is suppressed for brevity. Let G be a group (with generator g) of prime order p , (IPFE.Setup, IPFE.KeyGen, IPFE.Enc, IPFE.Dec) a G -encoded IPFE, and (Garble, Eval) be an AKGS for ABP. We construct a 1-ABE for predicate space

$$\mathcal{P} = \{P_n \mid n \in \mathbb{N}\}, \quad P_n(y, \mathbf{x}) = y(\mathbf{x}) \text{ for } y \in \text{ztABP}_p^n, \mathbf{x} \in \mathbb{Z}_p^n.$$

The scheme works as follows:

- Setup(1^n) takes as input the attribute length (i.e., P_n is represented by 1^n) and outputs (mpk, msk) $\stackrel{\$}{\leftarrow}$ IPFE.Setup(1^{n+1}).
- KeyGen(msk, \mathbf{x}) outputs sk $\stackrel{\$}{\leftarrow}$ IPFE.KeyGen(msk, (1, \mathbf{x})).
 Note: *If the underlying IPFE has succinct secret keys, so does this scheme. For instance, when instantiated with Construction 13 with $T = 2$ under DDH, each secret key consists of just three group elements.*
- Enc(mpkm, y, μ) garbles y with μ and encrypts the label functions in IPFE ciphertexts as follows:

$$\begin{aligned} &\text{if } y = f_{\neq 0}: \quad \alpha \leftarrow \mu, \beta \leftarrow 0; \\ &\text{if } y = f_{=0}: \quad \alpha \stackrel{\$}{\leftarrow} \mathbb{Z}_p, \beta \leftarrow \mu; \\ &\quad (\mathbf{L}_1, \dots, \mathbf{L}_m) \stackrel{\$}{\leftarrow} \text{Garble}(f, \alpha, \beta), \\ &\text{for } j \in [m]: \quad \text{ict}_j \stackrel{\$}{\leftarrow} \text{IPFE.Enc}(\text{mpk}, \llbracket \mathbf{L}_j \rrbracket). \end{aligned}$$

The algorithm outputs $\text{ct} = (\text{ict}_1, \dots, \text{ict}_m)$.

- Dec(sk, \mathbf{x}, ct, y) takes as input a secret key sk for \mathbf{x} and a ciphertext ct for y . If $y(\mathbf{x}) = 0$, the algorithm outputs \perp and stops. Otherwise, it computes

$$\begin{aligned} &\text{for } j \in [m]: \quad \llbracket \ell_j \rrbracket \leftarrow \text{IPFE.Dec}(\text{sk}, \text{ict}_j) \\ &\quad \llbracket \mu' \rrbracket \leftarrow \begin{cases} \frac{1}{f(\mathbf{x})} \text{Eval}(f, \mathbf{x}, \llbracket \ell_1, \dots, \ell_m \rrbracket), & \text{if } y = f_{\neq 0}; \\ \text{Eval}(f, \mathbf{x}, \llbracket \ell_1, \dots, \ell_m \rrbracket), & \text{if } y = f_{=0}; \end{cases} \end{aligned}$$

and outputs $\llbracket \mu' \rrbracket$ as the message.

Note: We show that the scheme is correct. By the correctness of IPFE, we have

$$\ell_j = \mathbf{L}_j^\top \begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix} = L_j(\mathbf{x}),$$

where L_j 's are the label functions defined by Garble. Since Eval is linear in the labels, it can be performed in the exponent. By the correctness of AKGS,

$$\begin{aligned} \text{if } y = f_{\neq 0}, f(\mathbf{x}) \neq 0: \quad & \mu' = \frac{1}{f(\mathbf{x})}(\alpha f(\mathbf{x}) + \beta) = \frac{1}{f(\mathbf{x})}(\mu f(\mathbf{x}) + 0) = \mu; \\ \text{if } y = f_{=0}, f(\mathbf{x}) = 0: \quad & \mu' = \alpha f(\mathbf{x}) + \beta = \alpha \cdot 0 + \mu = \mu. \end{aligned}$$

Theorem 16. Suppose in Construction 15, the IPFE is gradually 2-simulation-secure and the AKGS is piecewise secure, then the constructed 1-ABE is secure.

We refer the reader to the full version for the proof.

5 Key-Policy ABE for ABP

In this section, we apply the classic dual system encryption to obtain full KP-ABE from CP-1-ABE instantiated with the IPFE in Sect. 3.1.

Construction 17 (KP-ABE). The construction is described for a fixed value of λ , and λ is suppressed for brevity. Let G_1, G_2, G_T be pairing groups of prime order p for which MDDH_k holds in G_1, G_2 , and let (Garble, Eval) be an AKGS for ABP. We construct an ABE for message space G_T and predicate space

$$\mathcal{P} = \{P_n \mid n \in \mathbb{N}\}, \quad P_n(\mathbf{x}, y) = y(\mathbf{x}) \text{ for } \mathbf{x} \in \mathbb{Z}_p^n, y \in \text{ztABP}_p^n.$$

The scheme works as follows:

- Setup(1^n) takes as input the attribute length (i.e., P_n is represented by 1^n). It samples and sets

$$\begin{aligned} \mathbf{A} &\stackrel{\$}{\leftarrow} \mathbb{Z}_p^{k \times (k+2)}, \mathbf{B} \stackrel{\$}{\leftarrow} \mathbb{Z}_p^{k \times (k+1)}, & \mathbf{W} &\stackrel{\$}{\leftarrow} \mathbb{Z}_p^{(k+2) \times (k+1)(n+1)}, \boldsymbol{\mu} \stackrel{\$}{\leftarrow} \mathbb{Z}_p^{k+1}, \\ \text{xpk} &= ([\mathbf{B}^\top]_1, [\mathbf{W}(\mathbf{B}^\top \otimes \mathbf{I}_{n+1})]_1), & \text{fpk} &= ([\mathbf{A}]_2, [\mathbf{AW}]_2), \\ \text{mpk} &= ([\boldsymbol{\mu}^\top \mathbf{B}^\top]_T, \text{xpk}), & \text{msk} &= (\text{fpk}, \boldsymbol{\mu}). \end{aligned}$$

Note: We explain the connection with CP-1-ABE and dual system encryption (as demonstrated in Sect. 1.1). The matrix $\mathbf{W} = (\mathbf{W}_1 \cdots \mathbf{W}_{k+1})$ consists of $k + 1$ master secret keys of CP-1-ABE concatenated by columns, each of shape $(k + 2) \times (n + 1)$. Its projection along a vector $\mathbf{b} = (b_1, \dots, b_{k+1})^\top$ is

$$\begin{aligned} b_1 \mathbf{W}_1 + \cdots + b_{k+1} \mathbf{W}_{k+1} &= \mathbf{W}_1 \cdot b_1 \mathbf{I}_{n+1} + \cdots + \mathbf{W}_{k+1} \cdot b_{k+1} \mathbf{I}_{n+1} \\ &= (\mathbf{W}_1 \cdots \mathbf{W}_{k+1}) \begin{pmatrix} b_1 \mathbf{I}_{n+1} \\ \vdots \\ b_{k+1} \mathbf{I}_{n+1} \end{pmatrix} = \mathbf{W}(\mathbf{b} \otimes \mathbf{I}_{n+1}). \end{aligned}$$

The matrix $\mathbf{B} = (\mathbf{b}_1^\top \cdots \mathbf{b}_k^\top)^\top$ consists of all the projection vectors, and $\mathbf{W}(\mathbf{B}^\top \otimes \mathbf{I}_{n+1})$ is the projections of \mathbf{W} along \mathbf{B} concatenated by columns.

- KeyGen(msk, y) garbles y with μ as follows:

$$\begin{aligned} \text{if } y = f_{\neq 0}: & \quad \alpha \leftarrow \mu, \quad \beta \leftarrow \mathbf{0}; \\ \text{if } y = f_{=0}: & \quad \alpha \xleftarrow{\$} \mathbb{Z}_p^{k+1}, \quad \beta \leftarrow \mu; \\ & \quad (\mathbf{L}_1, \dots, \mathbf{L}_m) \xleftarrow{\$} \text{Garble}(f, \alpha, \beta). \end{aligned}$$

It samples $\mathbf{s}_j \xleftarrow{\$} \mathbb{Z}_p^k$ for $j \in [m]$ and sets

$$\text{sk}_{j,1} = \mathbf{s}_j^\top \llbracket \mathbf{A} \rrbracket_2, \quad \text{sk}_{j,2} = \mathbf{s}_j^\top \llbracket \mathbf{AW} \rrbracket_2 + \llbracket \mathbf{L}_j^\top \rrbracket_2.$$

The algorithm outputs $\text{sk} = (\text{sk}_{1,1}, \text{sk}_{1,2}, \dots, \text{sk}_{m,1}, \text{sk}_{m,2})$.

Note: *Generating a key in KP-ABE means encrypting μ in each CP-1-ABE instance, which boils down to generating IPFE ciphertexts, as shown above.*

- Enc(mpk, x, g) samples $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_p^k$ and sets

$$\text{ct}_1 = \llbracket \mathbf{B}^\top \rrbracket_1 \mathbf{r}, \quad \text{ct}_2 = \llbracket \mathbf{W}(\mathbf{B}^\top \otimes \mathbf{I}_{n+1}) \rrbracket_1 \left(\mathbf{r} \otimes \begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix} \right), \quad \text{ct}_3 = \llbracket \mu^\top \mathbf{B}^\top \rrbracket_{\mathbf{T}} \mathbf{r} + g.$$

The algorithms outputs $\text{ct} = (\text{ct}_1, \text{ct}_2, \text{ct}_3)$.

Note: *We remark that \mathbf{r} (resp. $\mathbf{B}^\top \mathbf{r}$) is the coefficients of random linear combination w.r.t. the projections (resp. the CP-1-ABE instances). Here, ct_2 corresponds to a CP-1-ABE key w.r.t. randomly combined master secret key $\mathbf{W}(\mathbf{B}^\top \mathbf{r} \otimes \mathbf{I}_{n+1})$, which is an IPFE secret key for $\begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix}$, i.e.,*

$$\text{ct}_2 = \llbracket \mathbf{W}(\mathbf{B}^\top \mathbf{r} \otimes \mathbf{I}_{n+1}) \begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix} \rrbracket_1.$$

The ciphertext consists of $2k + 3$ elements in G_1 and one element in $G_{\mathbf{T}}$, hence is succinct.

- Dec(sk, y, ct, x) first checks whether $y(\mathbf{x}) = 1$. If not, it outputs \perp and terminates. Otherwise, it parses sk, ct as defined in KeyGen, Enc, computes

$$\begin{aligned} \text{for } j \in [m]: \quad & \llbracket \ell_j \rrbracket_{\mathbf{T}} = -\text{sk}_{j,1} \text{ct}_2 + \text{sk}_{j,2} \left(\text{ct}_1 \otimes \begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix} \right); \\ & \llbracket \mu' \rrbracket_{\mathbf{T}} \leftarrow \begin{cases} \frac{1}{f(\mathbf{x})} \text{Eval}(f, \mathbf{x}, \llbracket \ell_1, \dots, \ell_m \rrbracket_{\mathbf{T}}), & \text{if } y = f_{\neq 0}; \\ \text{Eval}(f, \mathbf{x}, \llbracket \ell_1, \dots, \ell_m \rrbracket_{\mathbf{T}}), & \text{if } y = f_{=0}; \end{cases} \end{aligned}$$

and outputs $\text{ct}_3 - \llbracket \mu' \rrbracket_{\mathbf{T}}$ as the recovered message.

Note: *We show that the scheme is correct. By definition (also cf. Construction 13),*

$$\begin{aligned} \ell_j &= -\mathbf{s}_j^\top \mathbf{AW}(\mathbf{B}^\top \otimes \mathbf{I}_{n+1}) \left(\mathbf{r} \otimes \begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix} \right) + (\mathbf{s}_j^\top \mathbf{AW} + \mathbf{L}_j^\top) \left(\mathbf{B}^\top \mathbf{r} \otimes \begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix} \right) \\ &= \mathbf{L}_j^\top \left(\mathbf{B}^\top \mathbf{r} \otimes \begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix} \right) = \mathbf{L}_j^\top (\mathbf{B}^\top \mathbf{r} \otimes \mathbf{I}_{n+1}) \begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix}. \end{aligned}$$

By Lemma 5, if we define $(\mathbf{L}'_1, \dots, \mathbf{L}'_m) \leftarrow \text{Garble}(f, \mathbf{r}^\top \mathbf{B}\boldsymbol{\alpha}, \mathbf{r}^\top \mathbf{B}\boldsymbol{\beta}; \mathbf{r}^\top \mathbf{B}\mathbf{R})$, where \mathbf{R} is the randomness used to generate \mathbf{L}_j 's, then

$$\ell_j = \mathbf{L}'_j{}^\top (\mathbf{B}^\top \mathbf{r} \otimes \mathbf{I}_{n+1}) \begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix} = (\mathbf{L}'_j)^\top \begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix} = L'_j(\mathbf{x}).$$

By the correctness of AKGS, we have

$$\text{Eval}(f, \mathbf{x}, \ell_1, \dots, \ell_m) = \mathbf{r}^\top \mathbf{B}\boldsymbol{\alpha}f(\mathbf{x}) + \mathbf{r}^\top \mathbf{B}\boldsymbol{\beta}.$$

In the two cases where decryption should succeed,

$$\text{if } y = f_{\neq 0}, f(\mathbf{x}) \neq 0: \quad \mu' = \frac{1}{f(\mathbf{x})}(\mathbf{r}^\top \mathbf{B}\boldsymbol{\mu}f(\mathbf{x}) + \mathbf{r}^\top \mathbf{B}\mathbf{0}) = \mathbf{r}^\top \mathbf{B}\boldsymbol{\mu};$$

$$\text{if } y = f_{=0}, f(\mathbf{x}) = 0: \quad \mu' = \mathbf{r}^\top \mathbf{B}\boldsymbol{\alpha}f(\mathbf{x}) + \mathbf{r}^\top \mathbf{B}\boldsymbol{\mu} = \mathbf{r}^\top \mathbf{B}\boldsymbol{\mu}.$$

Therefore, in both cases, we have $\text{ct}_3 - \llbracket \mu' \rrbracket_T = \llbracket \boldsymbol{\mu}^\top \mathbf{B}^\top \mathbf{r} \rrbracket_T + g - \llbracket \mathbf{r}^\top \mathbf{B}\boldsymbol{\mu} \rrbracket_T = g$.

Minimizing Pairing Operations. The number of pairing operations in the decryption algorithm appears to depend on the garbling size of the policy and the attribute length. It can be reduced to $2k + 3$ as follows.¹⁴ Since Eval is linear in the labels, the decryption algorithm can first find $\gamma_1, \dots, \gamma_m \in \mathbb{Z}_p$ such that

$$\text{Eval}(f, \mathbf{x}, \ell_1, \dots, \ell_m) = \sum_{j=1}^m \gamma_j \ell_j.$$

The computation of $\llbracket \mu' \rrbracket_T$ can be rewritten as

$$\begin{aligned} \llbracket \mu' \rrbracket_T &= \sum_{j=1}^m \gamma_j \llbracket \ell_j \rrbracket_T = \sum_{j=1}^m \gamma_j \left(-\text{sk}_{j,1} \text{ct}_2 + \text{sk}_{j,2} \left(\text{ct}_1 \otimes \begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix} \right) \right) \\ &= - \left(\sum_{j=1}^m \gamma_j \text{sk}_{j,1} \right) \text{ct}_2 + \left(\sum_{j=1}^m \gamma_j \text{sk}_{j,2} \right) \left(\mathbf{I}_{k+1} \otimes \begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix} \right) \text{ct}_1. \end{aligned}$$

Note that ct_1, ct_2 consist of $k + 1, k + 2$ group elements and only these elements (in G_1) take part in pairing. Therefore, the formula above only uses $2k + 3$ pairing operations.

There are further optimizations possible, such as appropriately choosing which group of the two source groups to use for the secret key to reduce the cost of exponentiation in decryption. Next, we proceed to the security of our scheme.

¹⁴ Syntactically, we use the pairing groups in black box, and there are only $2k + 3$ elements in G_1 in a ciphertext and no element in G_1 in a secret key, so the operations can always be regrouped to use at most $2k + 3$ pairing operations. The content below provides the concrete regrouping method.

Theorem 18. *Suppose in Construction 17, $MDDH_k$ holds in both G_1 and G_2 , and the AKGS is piecewise secure, then the constructed scheme is IND-CPA secure.*

We refer the reader to the full version for the proof.

Acknowledgments. The authors were supported by NSF grants CNS-1528178, CNS-1929901, CNS-1936825 (CAREER), CNS-2026774, a Hellman Fellowship, a JP Morgan AI Research Award, the Defense Advanced Research Projects Agency (DARPA) and Army Research Office (ARO) under Contract No. W911NF-15-C-0236, and a subcontract No. 2017-002 through Galois. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, or the U.S. Government. The authors thank the anonymous reviewers for their valuable comments.

References

1. Abdalla, M., Bourse, F., De Caro, A., Pointcheval, D.: Simple functional encryption schemes for inner products. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 733–751. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46447-2_33
2. Agrawal, S., Libert, B., Maitra, M., Titiu, R.: Adaptive simulation security for inner product functional encryption. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020. LNCS, vol. 12110, pp. 34–64. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45374-9_2
3. Agrawal, S., Libert, B., Stehlé, D.: Fully secure functional encryption for inner products, from standard assumptions. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9816, pp. 333–362. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53015-3_12
4. Agrawal, S., Yamada, S.: Optimal broadcast encryption from pairings and LWE. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020. LNCS, vol. 12105, pp. 13–43. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45721-1_2
5. Attrapadung, N.: Dual system encryption framework in prime-order groups via computational pair encodings. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10032, pp. 591–623. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53890-6_20
6. Attrapadung, N., Hanaoka, G., Yamada, S.: Conversions among several classes of predicate encryption and applications to ABE with various compactness tradeoffs. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015. LNCS, vol. 9452, pp. 575–601. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48797-6_24
7. Attrapadung, N., Libert, B., de Panafieu, E.: Expressive key-policy attribute-based encryption with constant-size ciphertexts. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 90–108. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19379-8_6
8. Boneh, D., et al.: Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 533–556. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55220-5_30

9. Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-46035-7_4
10. Datta, P., Dutta, R., Mukhopadhyay, S.: Functional encryption for inner product with full function privacy. In: Cheng, C.-M., Chung, K.-M., Persiano, G., Yang, B.-Y. (eds.) PKC 2016. LNCS, vol. 9614, pp. 164–195. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49384-7_7
11. Emura, K., Miyaji, A., Nomura, A., Omote, K., Soshi, M.: A ciphertext-policy attribute-based encryption scheme with constant ciphertext length. In: Bao, F., Li, H., Wang, G. (eds.) ISPEC 2009. LNCS, vol. 5451, pp. 13–23. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00843-6_2
12. Escala, A., Herold, G., Kiltz, E., Ràfols, C., Villar, J.: An algebraic framework for Diffie-Hellman assumptions. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8043, pp. 129–147. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40084-1_8
13. Gong, J., Wee, H.: Adaptively secure ABE for DFA from k -Lin and more. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020. LNCS, vol. 12107, pp. 278–308. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45727-3_10
14. Ishai, Y., Wee, H.: Partial garbling schemes and their applications. In: Esparza, J., Fraigniaud, P., Husfeldt, T., Koutsoupias, E. (eds.) ICALP 2014. LNCS, vol. 8572, pp. 650–662. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43948-7_54
15. Kowalczyk, L., Wee, H.: Compact adaptively secure ABE for NC^1 from k -Lin. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019. LNCS, vol. 11476, pp. 3–33. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17653-2_1
16. Lewko, A., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 62–91. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_4
17. Lin, H., Luo, J.: Compact adaptively secure ABE from k -Lin: beyond NC^1 and towards NL. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020. LNCS, vol. 12107, pp. 247–277. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45727-3_9
18. Lin, H., Luo, J.: Succinct and adaptively secure ABE for arithmetic branching programs from k -Lin. Cryptology ePrint Archive (2020). (to appear)
19. Lin, H., Vaikuntanathan, V.: Indistinguishability obfuscation from DDH-like assumptions on constant-degree graded encodings. In: Dinur, I. (ed.) 57th FOCS, pp. 11–20. IEEE Computer Society Press, October 2016. <https://doi.org/10.1109/FOCS.2016.11>
20. Nisan, N.: Lower bounds for non-commutative computation (extended abstract). In: 23rd ACM STOC, pp. 410–418. ACM Press, May 1991. <https://doi.org/10.1145/103418.103462>
21. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005). https://doi.org/10.1007/11426639_27
22. Takashima, K.: Expressive attribute-based encryption with constant-size ciphertexts from the decisional linear assumption. In: Abdalla, M., De Prisco, R. (eds.) SCN 2014. LNCS, vol. 8642, pp. 298–317. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10879-7_17

23. Tomida, J., Attrapadung, N.: Unbounded dynamic predicate compositions in ABE from standard assumptions. Cryptology ePrint Archive, Report 2020/231 (2020). <https://eprint.iacr.org/2020/231>
24. Waters, B.: Dual system encryption: realizing fully secure IBE and HIBE under simple assumptions. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 619–636. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03356-8_36
25. Wee, H.: Dual system encryption via predicate encodings. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 616–637. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54242-8_26
26. Wee, H.: Attribute-hiding predicate encryption in bilinear groups, revisited. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017. LNCS, vol. 10677, pp. 206–233. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70500-2_8
27. Yamada, S., Attrapadung, N., Hanaoka, G., Kunihiro, N.: A framework and compact constructions for non-monotonic attribute-based encryption. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 275–292. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54631-0_16
28. Zhang, K., et al.: Practical and efficient attribute-based encryption with constant-size ciphertexts in outsourced verifiable computation. In: Chen, X., Wang, X., Huang, X. (eds.) ASIACCS 2016, pp. 269–279. ACM Press, May/June 2016