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 handle 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.
Contents

1 Introduction 1
   1.1 Technical Overview ................................................. 3
   1.2 Related Works ..................................................... 10
2 Preliminaries 11
   2.1 Arithmetic Branching Programs and Arithmetic Key Garbling ...... 11
   2.2 Attribute-Based Encryption ........................................ 13
   2.3 Pairing Groups and Matrix Diffie–Hellman Assumption ............. 14
3 IPFE with Gradual Simulation Security 15
   3.1 Construction of Gradually Simulation-Secure IPFE ................. 18
   3.2 One-Time Pad IPFE Scheme ......................................... 19
   3.3 Proof of Gradual Simulation Security ................................ 21
4 Ciphertext-Policy I-ABE for ABP 25
5 Key-Policy ABE for ABP 28
6 Ciphertext-Policy ABE for ABP 33
   6.1 KP-I-ABE ....................................................... 33
   6.2 CP-ABE ....................................................... 34
References 37
1 Introduction

Attribute-based encryption (ABE) [SW05] 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 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 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 desirata 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 [EMN+09] 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 [ALdP11,YAHK14,Tak14,Att16,AHY15,ZGT*16,TA20], 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.** Have 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

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 [TA20] 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, whereas all prior succinct ABE schemes (even ones relying on q-type 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 [KW19,GW20,LL20] simultaneously achieve (2)–(4), and the most expressive class of policies supported is also ABP, due to [LL20].

**Our Techniques.** The recent work of [LL20] 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 [LL20] yields linear-size ciphertexts because it crucially relies on function-hiding inner-product functional encryption (IPFE) [DDM16,LV16]. IPFE allows issuing secret keys and ciphertexts tied to vectors v, u respectively, and decryption reveals their

---

1One can always convert an arithmetic computation into a Boolean one, which we consider non-native.
inner product $\langle u, v \rangle$. The function-hiding property guarantees that nothing about $u, v$ 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 [LL20] to work with the weaker gradual simulation security and obtain succinct ciphertexts. Furthermore, we extend the framework to construct ciphertext-policy ABE, which is not handled in [LL20]. In summary, our techniques give a general and modular approach for constructing succinct and adaptively secure (KP and CP) ABE from MDDH.

**Organization.** In Section 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 Section 1.2. After introducing the preliminaries in Section 2, we define gradual simulation security of IPFE and construct such an IPFE scheme in Section 3. In Section 4, we define 1-ABE and construct CP-1-ABE for ABP with succinct keys from gradually simulation-secure IPFE with succinct keys. In Section 5, we show how to construct KP-ABE with succinct ciphertexts using our CP-1-ABE and dual system encryption. In Section 6, we construct 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 Figure 1.

![Figure 1: The roadmap of our constructions.](image)

**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 [LL20], which achieves adaptive security but not succinctness.

Suppose we want decryption to recover the message $\mu \in \mathbb{Z}_p$ if (and only if) $f(x) \neq 0$ for policy function $f : \mathbb{Z}_p^n \rightarrow \mathbb{Z}_p$ and attribute $x \in \mathbb{Z}_p^n$. This is equivalent to computing $\mu f(x)$ upon decryption. The basic idea of the LL 1-ABE is that when a key (tied to $f, \mu$)\(^2\) and a ciphertext (tied to $x$) are put together, one can compute a randomized encoding of $\mu f(x)$, denoted by $\overline{\mu f(x)}$, which reveals $\mu f(x)$ and hence $\mu$ if $f(x) \neq 0$. Since in ABE, we do not try to hide $f$ or $x$, the

\[^2\]The reason why we put the message $\mu$ in the key will become clear later in the overview.
randomized encoding only needs to hide $\mu$ beyond the output $\mu f(x)$, referred to as the partially hiding property, first introduced by [IW14]. 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

$$\mu f(x) = (L_1(x), \ldots, L_m(x)),$$

where $L_j$’s are affine functions of $x$ and the coefficients of $L_j$’s are linear in the message $\mu$ and the garbling randomness. $L_j$’s are called label functions and $\ell_j = L_j(x)$ are called labels.

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

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

Importantly, $\text{Eval}$ is linear in the labels.\(^3\)

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

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

Since the label functions are affine in $x$ thus linear in $(1, x)$, the labels $\ell_j = L_j(x)$ can be securely computed using a function-hiding IPFE. In IPFE, keys $\text{isk}(v)$ and ciphertexts $\text{ict}(u)$ are generated for vectors $v, u$, and decryption yields their inner product $\langle u, v \rangle$ 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:

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

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 ciphertext is an IPFE ciphertext encrypting $(1, x)$, as illustrated below in Real Algorithms. When they are put together, IPFE decryption recovers exactly the labels $\ell_j = L_j(x) = \langle L_j, (1, x) \rangle$, from which we can recover $\mu f(x)$ using the evaluation procedure. A technicality is that known IPFE are built from pairing groups, and decryption only reveals $\mu f(x)$ in the exponent of the target group. Nevertheless, one can recover $\mu f(x)$ 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 $\mu f(x)$ is revealed, given the labels. It is simple to formalize this idea in the selective setting, where $x$ 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.

**REAL ALGORITHMS**

$$\begin{align*}
\{\text{ct}_x: & \text{ict} (1, x) \} \approx \{\text{ct}_x: \text{ict} (1, x) \} \\
\{\text{sk}_{f, \mu}: & \{\text{isk}_j( L_j)\}_{j \in [m]} \} \end{align*}$$

**HYBRID**

$$\begin{align*}
\{\text{ct}_x: & \text{ict} (1, x) \} \approx \{\text{ct}_x: \text{ict} (1, x) \} \\
\{\text{sk}_{f, \mu}: & \{\text{isk}_j( L_j(x)) \} \}_{j \in [m]} \end{align*}$$

---

\(^3\)In contrast, linear evaluation is impossible for fully hiding randomized encoding that hides $x$ and $f$. 

4
After labels \( L_j(x) \) are hardwired and label functions removed, AKGS security guarantees that the labels only reveal \( \mu f(x) \), and \( \mu \) is hidden if \( f(x) = 0 \). 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, \( x \) is unknown and consequently the labels \( L_j(x) \) 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 \( \ell_2, \ldots, \ell_m \) is uniformly random, and \( \ell_1 \) can be reversely computed from these other labels \( \ell_2, \ldots, \ell_m \) and \( f, x \), by finding the unique \( \ell_1 \) satisfying the constraint of evaluation correctness.\(^4\)

- The other labels are marginally random even given the coefficients of all subsequent label functions, i.e.,

\[
(L_j(x), L_{j+1}, \ldots, L_m) \equiv (z, L_{j+1}, \ldots, L_m) \quad \text{for } z \leftarrow \mathbb{Z}_p, \quad \text{for all } j > 1.
\]

The first property implies a specific simulation strategy: Simply sample \( \ell_2, \ldots, \ell_m \) as random, then solve for \( \ell_1 \) from the correctness constraint. This strategy is particularly suitable for the adaptive setting, as only the simulation of \( \ell_1 \) depends on the input \( x \). Thus, a conceivable simulation strategy for 1-ABE is to hardwire \( \ell_2, \ldots, \ell_m \) in the secret key and \( \ell_1 \) in the ciphertext. This would not hurt the compactness of the ciphertext.

Proving the indistinguishability of the real and the simulated worlds takes two steps. In the first step, the first label \( \ell_1 = L_1(x) \) is hardwired into the IPFE ciphertext \( \text{ict} \), and then changed to be reversely computed from the other labels and \( f, x \), which is possible since by the time we generate \( \text{ict} \), we know both \( f \) and \( x \). In the second step, each \( \text{isk}_j \) for \( j > 1 \) is, one by one, switched from encoding the label function to encoding a random label. To do so, the \( j^{\text{th}} \) label \( \ell_j = L_j(x) \) is first hardwired into \( \text{ict} \), after which it is switched to random relying on piecewise security, and lastly moved back to \( \text{isk}_j \). Observe that the proof uses two extra slots in the vectors (one for \( \ell_1 \), the other for each \( \ell_j \) 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, \( x \) 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 [ABDP15, ALS16] have succinct keys.

Since the coefficients of the label functions (which contains information about \( \mu \) and the garbling randomness) must be hidden for the 1-ABE to be secure, and \( x \) is public, we should encrypt the coefficients of the label functions in IPFE ciphertexts and use an IPFE key for \((1, x)\) to compute the garbling. Since the message \( \mu \) is together with \( f \) and the generation of IPFE ciphertexts is

\(^4\)The original definition only requires \( \ell_1 \) to be reversely sampleable. In [LL20], 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, \( \ell_1 \) also depends on the computation result. For the purpose of this overview, the result is always \( \mu f(x) = 0 \) as the adversary is restricted to non-decrypting queries.
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 $x$. 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:

\[
\text{sk}_x \colon \text{isk} (1, x) \xrightarrow{\text{IPFE encryption}} \{\langle L_j, (1, x) \rangle \mid j \in [m] \} \xrightarrow{\text{decryption}} \{\ell_j \mid j \in [m] \} \xrightarrow{\text{AKGS evaluation}} \mu f(x).
\]

Our CP-1-ABE is $x$-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 $\ell_1$ and one of $\ell_j$'s with $x$, 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 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.

\[
\begin{array}{ll}
\text{REAL} & \text{SIMULATION} \\
\{\text{isk}_j (v_j) \mid j \leq J^* \} & \{\text{isk}_j (v_j) \mid \emptyset \} \\
\text{ict} (u) & \text{ict} (\emptyset) \\
\{\text{isk}_j (v_j) \mid j > J^* \} & \{\text{isk}_j (v_j) \mid \{\langle u, v_j \rangle \} \} \\
\end{array}
\]

\(J^*\) 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 $u$ 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 $v_j$, the inner product $\langle u, v_j \rangle$ is provided with $v_j$.

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

\(^5\)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.

\(^6\)Anyone can encrypt the standard basis vectors using $\text{mpk}$, 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 $\{v_j\}_{j \leq J^*}$ in the keys are public, these inner products are determined by those with any maximal subset of linearly independent $v_j$'s, the number of which will not exceed the dimension. As such, the simulated ciphertext can still be compact.
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 skᵭ = isk(1, x) after we have created the ciphertext ctᵮ,ᵩ (in which ictᵮ has been simulated), the inner product ℓᵪ is supplied to the simulator when we simulate isk, after the simulation of ictᵮ. This means the label ℓᵪ is hardwired into isk.

Let us exemplify the proof of adaptive security in the more difficult case where skᵭ is queried after ctᵮ,ᵩ. First, we simulate ict₁ so that the first label is hardwired into isk.

**REAL ALGORITHMS**

\[
\begin{align*}
\text{ctᵮ,ᵩ: } & \{ \text{ictᵮ} (L₁) \} \\
& \{ \text{ictᵮ} (Lᵩ) \}_{j>1} \\
\text{skᵭ: } & \text{isk (1, x)}
\end{align*}
\]

\[
\begin{align*}
\text{ℓᵩ HARDWIRED} & \\
\text{ctᵮ,ᵩ: } & \{ \text{ictᵮ} (Ø) | Ø \} \\
& \{ \text{ictᵮ} (LⱤ) \}_{j>1} \\
\text{skᵭ: } & \text{isk (1, x | ℓ₁)}
\end{align*}
\]

(Wo omitted the master public key for brevity.) Note that ictᵮ’s for j > 1 do not use ciphertext simulation but are created using the master public key (honest or simulated). Once ℓ₁ is hardwired, we can instead solve for it from the correctness equation.

The second step is to switch ictᵮ(LⱤ) to ictᵮ(ℓⱤ, 0) for ℓⱤ ⇥ ℤₚ one by one, i.e., to simulate ℓⱤ as random. To do so, we first simulate ictᵮ (hardwiring ℓ₁ = L₁(x) into isk), then switch ℓⱤ to random (via piecewise security), and lastly revert ictᵮ back to encryption (not simulated), but encrypting (ℓⱤ, 0) instead.

**BEFORE/AFTER SIMULATING ℓⱤ**

\[
\begin{align*}
\text{ctᵮ,ᵩ: } & \{ \text{ictᵮ} (Ø) | Ø \} \\
& \{ \text{ictᵮ} (ℓⱤ, 0) \}_{j<j'} \{ \text{ictᵮ} (LⱤ) \}_{j>j'} \\
\text{skᵭ: } & \text{isk (1, x | ℓ₁)}
\end{align*}
\]

\[
\begin{align*}
\text{ℓ₁, ℓⱤ HARDWIRED} & \\
\text{ctᵮ,ᵩ: } & \{ \text{ictᵮ} (Ø) | Ø \} \\
& \{ \text{ictᵮ} (ℓⱤ, 0) \}_{j<j'} \{ \text{ictᵮ} (LⱤ) \}_{j>j'} \\
\text{skᵭ: } & \text{isk (1, x | ℓ₁, ℓⱤ)}
\end{align*}
\]

\[ℓ₁, ℓⱤ \leftarrow \text{Z}_p, \text{SOLVE FOR ℓ₁} \]

\[ℓⱤ = LⱤ(x) \text{ OR ℓⱤ} \leftarrow \text{Z}_p\]

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 (⋆) only requires the real world to be indistinguishable from simulation. However, in the step of simulating ℓⱤ as random, we need to switch ictⱤ to simulation when ict₁ is already simulated (and symmetrically, reverting ictⱤ back to encryption while keeping ict₁ 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.,

\[(\text{mpk, ict₁, ict₂, } \{ \text{iskⱤ} \}_{j}) \approx (\text{mpk, ict₁, ict₂, } \{ \text{iskⱤ} \}_{j})?\]

Note that when we want to simulate ℓⱤ, the computation of ℓ₁ has complicated dependency on x, and we cannot hope to get around the issue by first reverting ict₁ back to normal encryption then simultaneously simulating ict₁, ictⱤ, because we do not know what to encrypt in ict₁.

**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

---

8In fact, the computation is as complex as the computation of f(x).
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 [ALS16] 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.

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 $\text{ict}_1$. Therefore, we can simply prepare two instances of IPFE (with independently generated master public and secret keys), one dedicated to $\text{ict}_1$ and the other to $\text{ict}_j$'s (for $j > 1$). During the proof, the instance for $\text{ict}_1$ 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 [ALdPi11,YAHK14,Tak14,Att16,AHY15], 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 [AHY15] 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 [ZGT$^*$16,TA20], 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 $\text{ciphertext}(f, \mu)$, where $\mu$ is the message in CP-1-ABE and encapsulated key in KP-ABE. A KP-ABE ciphertext for attribute $x$ and message $m$ consists of a CP-1-ABE key $\text{cipher}(x)$ and the masked message $\mu + m$. If decryption is authorized, CP-1-ABE decryption will give us $\mu$, 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 $\mu$ (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 $\text{cpmsk}, \text{cpsk}, \text{cpct}$, messages are vector spaces over $\mathbb{Z}_p$, and\(^9\)

\[
\begin{align*}
  k_1 \text{cpsk}(\text{cpmsk}_1, x) + k_2 \text{cpsk}(\text{cpmsk}_2, x) &= \text{cpsk}(k_1 \text{cpmsk}_1 + k_2 \text{cpmsk}_2, x), \\
  k_1 \text{cpct}(\text{cpmsk}_1, f, \mu_1) + k_2 \text{cpct}(\text{cpmsk}_1, f, \mu_2) &= \text{cpct}(k_1 \text{cpmsk}_1 + k_2 \text{cpmsk}_2, f, k_1 \mu_1 + k_2 \mu_2).
\end{align*}
\]

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

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

\[
\lceil \text{cpsk}(\text{cpmsk}, x) \rceil = \text{cpsk}(\lceil \text{cpmsk} \rceil, x).
\]

Assume for the moment that this can also be done for $\text{cpct}'s$ and decryption still works.\(^10\) Given the linearity, we can employ dual system encryption [Wat09] 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\(_k\) [CS02, EHK+13].\(^11\)

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:

\[
\begin{align*}
  \text{kpmpk} &= \lceil b_1, b_2, b_1 \text{cpmsk}_1 + b_2 \text{cpmsk}_2, b_1 \mu_1 + b_2 \mu_2 \rceil \quad \text{for } b_1, b_2 \overset{\$}{\leftarrow} \mathbb{Z}_p, \\
  \text{kpmsk} &= (\text{cpmpk}_1, \text{cpmpk}_2, \text{cpmsk}_1, \text{cpmsk}_2, \mu_1, \mu_2).
\end{align*}
\]

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 $\mu$ to mask the message. A KP-ABE key consists of two CP-1-ABE ciphertexts, one in each instance encrypting the corresponding encapsulated key.

\[
\begin{align*}
  \text{kpct}(x, m) &= (s \lceil b_1, b_2 \rceil, \text{cpsk}(s \lceil b_1 \text{cpmsk}_1 + b_2 \text{cpmsk}_2 \rceil, x), m + s \lceil b_1 \mu_1 + b_2 \mu_2 \rceil) \quad \text{for } s \overset{\$}{\leftarrow} \mathbb{Z}_p, \\
  \text{cpsk}(f) &= (\text{cpct}(\text{cpmsk}_1, f, \mu_1), \text{cpct}(\text{cpmsk}_2, f, \mu_2)).
\end{align*}
\]

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

\[
\begin{align*}
  \text{cpct}(s \lceil b_1 \text{cpmsk}_1 + s_2 \text{cpmsk}_2 \rceil, f, s \lceil b_1 \mu_1 + s_2 \mu_2 \rceil) \\
  = s \lceil b_1 \rceil \text{cpct}(\text{cpmsk}_1, f, \mu_1) + s \lceil b_2 \rceil \text{cpct}(\text{cpmsk}_2, f, \mu_2).
\end{align*}
\]

\(^9\)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, $\text{cpct}'s$ are already group-encoded, and this is where pairing comes in.

\(^11\)A few examples are [Wee17, ALS16, KW19, GW20]. Wee [Wee14] 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.
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 \( [sb_1 \mu_1 + sb_2 \mu_2] \), using which we can unmask to obtain the message \( m \).

To argue security, we first replace \( [sb_1, sb_2] \) used in the challenge ciphertext by \( [a_1, a_2] \) for random \( a_1, a_2 \in \mathbb{Z}_p \) (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 \( \mu \)'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 \( \text{cpmsk}'_1 \) but not \( \text{cpmsk}'_2 \). A KP-ABE secret key for policy \( f \) is essentially \( \text{cpc}(\text{cpmsk}'_1, f, \mu_1) \) and \( \text{cpc}(\text{cpmsk}'_2, f, \mu_2) \). The challenge ciphertext has \( \text{cpsk}(\text{cpmsk}'_2, x) \), and the message is masked by \( \mu_2' \). By CP-1-ABE security, \( \mu_2' \) (in \( \text{cpct}'s \) should be hidden, which means the message in the challenge ciphertext is hidden by \( \mu_2' \).

The proof completes by replacing \( \mu_2' \) 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, 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)

\[
\begin{align*}
\text{cpmpk} &= (d_1, d_2, d_1 \text{kpmask}_1 + d_2 \text{kpmask}_2, d_1 v_1 + d_2 v_2) \quad \text{for } d_1, d_2 \in \mathbb{Z}_p, \\
\text{cpmsk} &= (\text{kpmask}_1, \text{kpmask}_2, \text{kpmask}_1, \text{kpmask}_2, v_1, v_2), \\
\text{cpsk} &= (\text{kpc}(\text{kpmask}_1, x, v_1), \text{kpc}(\text{kpmask}_2, x, v_2)), \\
\text{cpc}(t d_1, t d_2, \text{kpsk}(t (d_1 \text{msk}_1 + d_2 \text{msk}_2), f), m + t (d_1 v_1 + d_2 v_2)) \quad \text{for } t \in \mathbb{Z}_p.
\end{align*}
\]

Again, KP-1-ABE is used to encapsulate keys \( v_1, v_2 \), 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 \( \mu_1, \mu_2 \) 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 main content.

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 [KW19,TA20] for Boolean formulae, [GW20] for Boolean branching programs, and [LL20] for arithmetic branching programs. Among them, only [TA20] achieves succinctness.

**ABE with Succinct f-Part.** From pairing, we know several ABE schemes with succinct 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 [BGG+14] and CP-ABE schemes for NC¹ based on LWE and pairing [AY20], 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 $x$-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., [KW19] for NC¹). 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

Let $\lambda$ be the security parameter that runs through $\mathbb{N}$. Except in the definitions, we suppress the security parameter for convenience. Efficient algorithms are probabilistic polynomial-time (PPT) Turing machines. Efficient adversaries are non-uniform PPT Turing machines, or equivalently families of polynomial-sized circuits. We denote by $H_0 \approx H_1$ (resp. $\approx_s$ and $\equiv$) for the computational indistinguishability (resp. statistical indistinguishability, identity) of two distributions or experiments.

We write $[n]$ for the set $\{1, 2, \ldots, n\}$. Vectors and matrices are written in boldface, and are always indexed using $[\cdot]$, i.e., $A[i, j]$ is the $(i, j)$-entry of $A$. Let $A$ be an $m \times n$ matrix and $B$ an $m' \times n'$ matrix. The tensor product $A \otimes B$ is a block matrix of shape $mm' \times nn'$:

$$
\begin{pmatrix}
\vdots & \vdots & \ddots & \vdots \\
A[m, 1]B & A[m, 2]B & \cdots & A[m, n]B
\end{pmatrix}.
$$

We note that $AB \otimes CD = (A \otimes B)(C \otimes D)$ holds as long as the multiplications are compatible. 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 $f \in \mathbb{Z}_p^{n+1}$ (the same letter in boldface) such that $f(x) = f^T(1_x)$.

### 2.1 Arithmetic Branching Programs and Arithmetic Key Garbling

In this paper, we consider the class of decryption policies defined by arithmetic branching programs [Nis91].

**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

$$
f(x) = \sum_{s \rightarrow t \text{ path}} \prod_{j=1}^i w(e_j, x).
$$
Its size (denoted by $|f|$) is $|V|$. It induces two zero-test predicates:

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

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:

- Garble($f, \alpha, \beta; r$) 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 $r \in \mathbb{Z}_p^m$. It outputs coefficient vectors $L_1, \ldots, L_m \in \mathbb{Z}_p^{n+1}$ of an affine function $L_1, \ldots, L_m : \mathbb{Z}_p^n \rightarrow \mathbb{Z}_p$ (called label functions). The vectors $L_j$ are linear in $(\alpha, \beta, r)$. 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 $f$.

- Eval($f, x, \ell_1, \ldots, \ell_m$) takes as input a function $f : \mathbb{Z}_p^n \rightarrow \mathbb{Z}_p \in \mathcal{F}$, an input $x \in \mathbb{Z}_p^n$, and $m$ labels $\ell_1, \ldots, \ell_m \in \mathbb{Z}_p$. It outputs $\gamma \in \mathbb{Z}_p$ that is linear in $\ell_1, \ldots, \ell_m$.

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

$$\Pr \left[ \forall j \in [m], \ell_j \leftarrow L_j(x) \quad \text{Garble}(f, \alpha, \beta) \quad : \quad \text{Eval}(f, x, \ell_1, \ldots, \ell_m) = af(x) + \beta \right] = 1.$$

We rely on the strong notion of piecewise security recently introduced in [LL20].

**Definition 3** (piecewise security). Let (Garble, 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 $f : \mathbb{Z}_p^n \rightarrow \mathbb{Z}_p \in \mathcal{F}, \alpha, \beta \in \mathbb{Z}_p, x \in \mathbb{Z}_p^n$, the following distributions are identical:

$$\begin{cases} (L_1, \ldots, L_m) \leftarrow \text{Garble}(f, \alpha, \beta) \\ \ell_1 \leftarrow L_1(x) \end{cases} : (\ell_1, L_2, \ldots, L_m)$$

$$\equiv \begin{cases} (L_1, \ldots, L_m) \leftarrow \text{Garble}(f, \alpha, \beta) \\ \ell_j \leftarrow L_j(x) \quad \text{for } j \in [m], j > 1 \\ \ell_1 \leftarrow \text{RevSamp}(f, x, af(x) + \beta, \ell_2, \ldots, \ell_m) \end{cases} \quad (\ell_1, L_2, \ldots, L_m).$$

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

$$\begin{cases} (L_1, \ldots, L_m) \leftarrow \text{Garble}(f, \alpha, \beta) \\ \ell_j \leftarrow L_j(x) \end{cases} : (\ell_j, L_{j+1}, \ldots, L_m)$$

$$\equiv \begin{cases} (L_1, \ldots, L_m) \leftarrow \text{Garble}(f, \alpha, \beta) \\ \ell_j \leftarrow Z_p \ \text{and} \ \ell_{j+1} \leftarrow Z_p \ldots \ell_m \end{cases} \quad (\ell_j, L_{j+1}, \ldots, L_m).$$
A piecewise secure AKGS is known for ABPs:

**Lemma 4 ([IW14,LL20]).** 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 Garble\((f, \alpha, \beta)\) is executed component-wise with independent randomness and the output are concatenated:

\[
\text{for } t \in [k]: \quad (L_1^{(t)}, \ldots, L_m^{(t)}) \leftarrow \text{Garble}(f, \alpha[t], \beta[t]);
\]

\[
\text{for } j \in [m]: \quad L_j = \begin{pmatrix} L_j^{(1)} \\ \vdots \\ L_j^{(k)} \end{pmatrix} = \sum_{t=1}^k e_j \otimes L_j^{(t)};
\]

output \((L_1, \ldots, L_m)\).

Here, \( e_j \in \mathbb{Z}_p^k \) are the standard basis vectors and \( L_j \)'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 \( \alpha, \beta \in \mathbb{Z}_p^k \), \( R \in \mathbb{Z}_p^{k \times m'} \), \( c \in \mathbb{Z}_p^k \), and define

\[
(L_1, \ldots, L_m) \leftarrow \text{Garble}(f, \alpha, \beta; R), \\
(L_1', \ldots, L_m') \leftarrow \text{Garble}(f, c' \alpha, c' \beta; c' R),
\]

then \( L_j^T(c \otimes I_{n+1}) = (L_j')^T \) for all \( j \in [m] \).

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

\[
(L_1', \ldots, L_j') \leftarrow \text{Garble}(f, \alpha, \beta; r), \\
(L_1, \ldots, L_j) \leftarrow \text{Garble}(f, \alpha d, \beta d; d r^T),
\]

then \( d \otimes L_j' = L_j \) 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 \( M = \{M_\lambda\}_{\lambda \in \mathbb{N}} \) be a sequence of message sets and \( P = \{P_\lambda\}_{\lambda \in \mathbb{N}} \) a sequence of predicate families with \( P_\lambda = \{P : X_P \times Y_P \rightarrow \{0, 1\}\} \). An attribute-based encryption (ABE) scheme for message space \( M \) and predicate space \( P \) consists of four efficient algorithms:

- Setup\((1^4, P)\) takes as input the security parameter \( 1^4 \) and a predicate \( P \in P_\lambda \), and outputs a pair of master public/secret keys (mpk, msk).
- KeyGen\((\text{msk}, y)\) takes as input a policy \( y \in Y_P \) and outputs a secret key sk.
- 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.
- Dec\((\text{sk}, y, ct, x)\) 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 \( \lambda \in \mathbb{N}, g \in M_\lambda, P \in \mathcal{P}_\lambda, x \in X_P, y \in Y_P \) s.t. \( P(x, y) = 1 \),

\[
\begin{align*}
\Pr & \left[ (\text{mpk, msk}) \xleftarrow{\$} \text{Setup}(1^\lambda, P) \\
& \quad \text{sk} \xleftarrow{\$} \text{KeyGen}(\text{msk}, y) : \text{Dec} (\text{sk}, y, ct, x) = g \right] = 1.
\end{align*}
\]

**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:

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

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

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

**Definition 8 (succinct ABE).** An ABE scheme has succinct ciphertext if the length of \( ct \) is a fixed polynomial in security parameter \( \lambda \) (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 \( \lambda \).

The above definition does not rule out trivially succinct schemes, e.g., one only supporting \( x, y \) of length at most \( \lambda \). 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 [LOS+10]).** Adopt the notations in Definition 6. The scheme is IND-CPA secure if \( \text{Exp}^0_{\text{CPA}} = \text{Exp}^1_{\text{CPA}} \), where \( \text{Exp}^b_{\text{CPA}} \) with adversary \( A \) proceeds as follows:

- **Setup.** Launch \( A(1^\lambda) \) and receive from it a predicate \( P \in \mathcal{P}_\lambda \). Run \( (\text{mpk}, \text{msk}) \xleftarrow{\$} \text{Setup}(1^\lambda, P) \) and send \( \text{mpk} \) to \( A \).

- **Query I.** Repeat the following for arbitrarily many rounds determined by \( A \): In each round, \( A \) submits a policy \( y_q \in Y_p \) for a secret key. Upon this query, run \( \text{sk}_q \xleftarrow{\$} \text{KeyGen}(\text{msk}, y) \) and send \( \text{sk}_q \) to \( A \).

- **Challenge.** The adversary submits the challenge attribute \( x^* \in X_P \) and two messages \( g_0, g_1 \in M_\lambda \). Run \( ct \xleftarrow{\$} \text{Enc}(\text{mpk}, x, g_0) \) and return \( ct \) to \( A \).

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

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

Definition 10 (MDDH [EHK’13]). 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 MDDH\(_q^{k,\ell}\) assumption holds in \(G\) if

\[
\{[\mathcal{A}, S^TA]\}_{\lambda \in \mathbb{N}} \approx \{[\mathcal{A}, C^T]\}_{\lambda \in \mathbb{N}} \quad \text{for} \quad A \leftarrow Z_{p(\lambda)}^{k \times \ell(\lambda)}, S \leftarrow Z_{p(\lambda)}^{k \times q(\lambda)}, C \leftarrow Z_{p(\lambda)}^{\ell(\lambda) \times q(\lambda)}.
\]

By default, \(\ell = k + 1\) and \(q = 1\). It is known [EHK’13] that \(k\)-Lin implies MDDH\(_k\), which further implies MDDH\(_{q}^{k,\ell}\) for any polynomial \(\ell, q\).

3 IPFE with Gradual Simulation Security

In an inner-product functional encryption scheme, secret keys and ciphertexts are associated with vectors. Decryption reveals the inner product and nothing more about the plaintext vector. We will use a public-key IPFE as our building block for 1-ABE, a precursor to full ABE. 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**\((1^4, 1^n, 1^T)\) takes as input the security parameter \(1^4\), 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 \((\text{mpk}, \text{msk})\).
- **KeyGen**\((\text{msk}, \underline{v})\) takes the master secret key and a vector as input, and outputs a secret key \(\text{sk}\).
- **Enc**\((\text{mpk}, [\underline{u}]_i)\) takes the master public key and a vector (encoded in \(G\)) as input, and outputs a ciphertext \(\text{ct}\).
- **Dec**\((\text{sk}, \underline{v}, \text{ct})\) 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 $\lambda, n, T \in \mathbb{N}, u, v \in \mathbb{Z}_{p(\lambda)}^n$, it holds that

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

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

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 [ALMT20]). 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 $J, J', J_i$) always counts the keys.

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

- **SimSetup**$(1^1, 1^n, 1^T)$ takes the same input as Setup, and outputs a simulated master public key $mpk$ and an internal state $st$.

- **SimKeyGen**$(st, v, z_1, \ldots, z_I)$ takes as input the internal state $st$, a vector $v$, and a list $z_1, \ldots, z_I$ 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 $st'$.

- **SimEnc**$(st, z_1, \ldots, z_J)$ takes as input the internal state $st$ and a list $z_1, \ldots, z_J$ 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 $A$:

---

12It 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.
• **Setup.** Launch \( A(1^λ) \) and receive from it \((1^n, 1^T)\). Run

\[
\begin{align*}
\text{in } \text{Exp}_{\text{real}}: & \quad (\text{mpk}, \text{msk}) \leftarrow \text{Setup}(1^λ, 1^n, 1^T) \\
\text{in } \text{Exp}_{\text{sim}}: & \quad (\text{mpk}, \text{st}) \leftarrow \text{SimSetup}(1^λ, 1^n, 1^T)
\end{align*}
\]

and send mpk to \( A \).

• **Challenge.** Repeat the following for arbitrarily many rounds determined by \( A \): In each round, \( A \) submits a vector \( v_j \). Upon this challenge, run

\[
\begin{align*}
\text{in } \text{Exp}_{\text{real}}: & \quad \text{sk}_j \leftarrow \text{KeyGen}(\text{msk}, v_j) \\
\text{in } \text{Exp}_{\text{sim}}: & \quad (\text{sk}_j, \text{st}') \leftarrow \text{SimKeyGen}(\text{st}, v_j) \quad \text{st} \leftarrow \text{st}'
\end{align*}
\]

and send \( \text{sk}_j \) to \( A \).

• **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 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 \( t \).

\( \text{Exp}^b_{T-\text{GS}} \) ( \( b \in \{0, 1\} \)) with adversary \( A \) proceeds as follows:

• **Setup.** Launch \( A(1^λ) \) and receive from it \((1^n, 1^T)\). Run

\[(\text{mpk}, \text{st}) \leftarrow \text{SimSetup}(1^λ, 1^n, 1^T)\]

and send mpk to \( A \).

• **Query I.** Repeat the following for arbitrarily many rounds determined by \( A \): In each round, \( A \) has two options.

  – *Key Simulation Query:* \( A \) can submit a vector \( v_j \) with a list \( z_{\leq I_t,j} \) of inner products for a secret key \( \text{sk}_j \). The list \( z_{\leq I_t,j} \) consists of \( z_{1,j}, \ldots, z_{I_t,j} \), all the decryption outcomes between \( \text{sk}_j \) and the simulated ciphertexts queried up to this point. Upon this query, run

\[(\text{sk}_j, \text{st}') \leftarrow \text{SimKeyGen}(\text{st}, v_j, z_{1,j}, \ldots, z_{I_t,j}) \quad \text{st} \leftarrow \text{st}'\]

and send \( \text{sk}_j \) to \( A \).

  – *Ciphertext Simulation Query:* \( A \) can submit a list \( z_{i,\leq J_t} \) of inner products for a simulated ciphertext \( \text{ct}_i \). The list \( z_{i,\leq J_t} \) consists of \( z_{i,1}, \ldots, z_{i,J_t} \), all the decryption outcomes between \( \text{ct}_i \) and the simulated secret keys queried up to this point. Upon this query, run

\[(\text{ct}_i, \text{st}') \leftarrow \text{SimEnc}(\text{st}, z_{i,1}, \ldots, z_{i,J_t}) \quad \text{st} \leftarrow \text{st}'\]

and send \( \text{ct}_i \) to \( A \).
• **Challenge.** The adversary submits a vector \( u^* \). 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

\[
\begin{align*}
  b = 0: & \quad ct^* \leftarrow Enc(mpk, \|u^*\|) \\
  b = 1: & \quad ct^* \leftarrow SimEnc(st, (u^*)^Tv_1, \ldots, (u^*)^Tv_{J^*}) \quad st \leftarrow st'
\end{align*}
\]

and send \( ct^* \) to \( A \).

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

\[
\begin{align*}
  b = 0: & \quad (sk_j, st') \leftarrow \text{SimKeyGen}(st, v_j, z_{1,j}, \ldots, z_{I^*,j}, z_{I^*+1,j}, \ldots, z_{I,j}); \\
  b = 1: & \quad (sk_j, st') \leftarrow \text{SimKeyGen}(st, v_j, z_{1,j}, \ldots, z_{I^*,j}, (u^*)^Tv_j, z_{I^*+1,j}, \ldots, z_{I,j}); \\
  & \quad st \leftarrow st' \quad \text{(in either case)}.
\end{align*}
\]

• **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 \( \langle u^i, v_j \rangle = z_{i,j} \quad \forall i, j \) (about \( u_i \)'s) has a solution.

Otherwise, the outcome is set to 0.

**Remarks.** In \( \text{Exp}_{T,GS}^b \), 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 \( u^* \) in \( \text{Exp}_{T,GS}^b \) 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 [ALS16] has been proven simulation-secure [ALMT20]. 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 \( \lambda \), and is independent of \( n \), which eventually translates into the succinctness of our ABE scheme.

**Construction 13 ([ALS16]).** The construction is described for a fixed value of \( \lambda \), and \( \lambda \) is suppressed for brevity. Let \( G \) be a group (with generator \( g \)) of prime order \( p \) such that MDDH holds in \( G \). Our \( G \)-encoded IPFE works as follows:
Setup($1^n, 1^T$) takes as input the dimension $n$ and the maximum number $T$ of simulated ciphertexts. It samples $A \leftarrow \mathbb{Z}_p^{k \times (k+T)}$, $W \leftarrow \mathbb{Z}_p^{(k+T) \times n}$ and outputs $\text{mpk} = [A, AW]$, $\text{msk} = W$.

KeyGen($\text{msk}, \mathbf{v}$) outputs $\text{sk} = W\mathbf{v}$.

Enc($\text{mpk}, [\mathbf{u}]$) samples $\mathbf{s} \leftarrow \mathbb{Z}_p^k$ and outputs $\text{ct} = (s' [A], s' [AW] + [\mathbf{u}'])$.

Dec($\text{sk}, \mathbf{v}, \text{ct}$) parses $\text{ct}$ as $([\mathbf{c'}], [\mathbf{t'}])$ and outputs $-[\mathbf{c'}] \text{sk} + [\mathbf{t'}] \mathbf{v}$.

The correctness is readily verified by

$$-[\mathbf{c'}] \text{sk} + [\mathbf{t'}] \mathbf{v} = -(s' A)(W \mathbf{v}) + (s' A W + \mathbf{u'}) \mathbf{v} = [\mathbf{u'} \mathbf{v}].$$

The scheme is succinct as $\text{sk}$ 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.

To prove this theorem, we first study the simple one-time pad IPFE scheme (OTP-IPFE). The simulator for Construction 13 can be constructed from the simulator of OTP-IPFE in a modular fashion.

### 3.2 One-Time Pad IPFE Scheme

OTP-IPFE is a secret-key IPFE scheme, whose syntax and security are defined below.

**Definition 15** (secret-key IPFE). A secret-key IPFE consists of four efficient algorithms:

- Setup($p, 1^n$) takes as input the modulus $p$ and the dimension $1^n$. It outputs a master secret key $\text{msk}$.
- KeyGen($\text{msk}, \mathbf{v}$) generates the secret key $\text{sk}$ for $\mathbf{v} \in \mathbb{Z}_p^n$.
- Enc($\text{msk}, \mathbf{u}$) encrypts $\mathbf{u} \in \mathbb{Z}_p^n$ into a ciphertext $\text{ct}$.
- Dec($\text{sk}, \mathbf{v}, \text{ct}$) is supposed to compute the inner product.

The scheme is required to be correct, i.e., for all prime $p$, natural number $n$, and $\mathbf{u}, \mathbf{v} \in \mathbb{Z}_p^n$,

$$\Pr \begin{bmatrix} \text{msk} \leftarrow \text{Setup}(p, 1^n) \\ \text{sk} \leftarrow \text{KeyGen}(\text{msk}, \mathbf{v}) : \text{Dec}(\text{sk}, \mathbf{v}, \text{ct}) = \mathbf{u}' \mathbf{v} \\ \text{ct} \leftarrow \text{Enc}(\text{msk}, \mathbf{u}) \end{bmatrix} = 1.$$

**Definition 16** (simulation security). Adopt the notations in Definition 15. A simulator consists of three efficient algorithms:

- SimSetup($p, 1^n$) takes the same input as Setup. It outputs an internal state $\text{st}$.
- SimKeyGen($\text{st}, \mathbf{v}, z$) takes as input the state, the vector, and optionally an inner product $z \in \mathbb{Z}_p$. It outputs a simulated secret key $\text{sk}$ and a new state $\text{st}'$.
- SimEnc($\text{st}, z_1, \ldots, z_J$) takes as input the state and a list of inner products. It outputs a simulated ciphertext $\text{ct}$ and a new state $\text{st}'$.
The simulator perfectly simulates the scheme if $\text{Exp}_{\text{real}} \equiv \text{Exp}_{\text{sim}}$, where the experiments with an (unbounded) adversary $A$ proceed as follows:

- **Setup.** Launch $A$ and receive from it $(p, 1^n)$. Run
  
  \begin{align*}
  \text{in } \text{Exp}_{\text{real}}: & \quad \text{msk} \leftarrow \text{Setup}(p, 1^n); \\
  \text{in } \text{Exp}_{\text{sim}}: & \quad \text{st} \leftarrow \text{SimSetup}(p, 1^n).
  \end{align*}

- **Query I.** Repeat the following for arbitrarily many rounds determined by $A$: In each round, $A$ submits a vector $v_j$. Upon this query, run
  
  \begin{align*}
  \text{in } \text{Exp}_{\text{real}}: & \quad \text{sk}_j \leftarrow \text{KeyGen}(\text{msk}, v_j); \\
  \text{in } \text{Exp}_{\text{sim}}: & \quad (\text{sk}_j, \text{st}') \leftarrow \text{SimKeyGen}(\text{st}, v_j) \quad \text{st} \leftarrow \text{st'};
  \end{align*}

  and send $\text{sk}_j$ to $A$.

- **Challenge.** The adversary submits a vector $u^*$. Upon the challenge, let the total number of secret key queries in Query I be $J^*$, run
  
  \begin{align*}
  \text{in } \text{Exp}_{\text{real}}: & \quad \text{ct}^* \leftarrow \text{Enc}(\text{msk}, u^*); \\
  \text{in } \text{Exp}_{\text{sim}}: & \quad (\text{ct}^*, \text{st}') \leftarrow \text{SimEnc}(\text{st}, (u^*)^TV_j, \ldots, (u^*)^TV_{J^*}) \quad \text{st} \leftarrow \text{st'};
  \end{align*}

  and send $\text{ct}^*$ to $A$.

- **Query II.** Same as Query I, except that in $\text{Exp}_{\text{sim}}$, we supply the inner product to the simulator:
  
  \begin{align*}
  \left(\text{sk}_j, \text{st}' \right) \leftarrow \text{SimGen}(\text{st}, v_j, (u^*)^TV_j) \quad \text{st} \leftarrow \text{st'}.
  \end{align*}

- **Guess.** The adversary outputs a bit $b'$, which is the output of the experiment.

We now describe the one-time pad IPFE scheme.

**Construction 17** (OTP-IPFE). The one-time pad IPFE works as follows:

- **Setup** $(p, 1^n)$ takes as input the prime $p$ and the dimension $n$. It samples $w \leftarrow \mathbb{Z}_p^n$ and outputs $\text{msk} = w$.
- **KeyGen** $(\text{msk}, v)$ outputs $\text{sk} = w^Tv$.
- **Enc** $(\text{msk}, u)$ outputs $\text{ct} = (w + u)^T$.
- **Dec** $(\text{sk}, v, \text{ct})$ outputs $-\text{sk} + \text{ct}v$.

The correctness follows by $-w^Tv + (w + u)^Tv = u^Tv$.


**Construction 18** (simulator for OTP-IPFE). The simulator works as follows:
• SimSetup($p, 1^n$) samples $\mathbf{w} \xleftarrow{\$} \mathbb{Z}_p^n$ and outputs $\text{st} = (\mathbf{w}, \bot)$.

• SimKeyGen$(\text{st}, \mathbf{v}, z)$ takes as input the state $\text{st}$, a vector $\mathbf{v}$, and optionally an inner product $z$. It parses $\text{st}$ as $(\mathbf{w}, ct', v_1, \ldots, v_{J'})$ and outputs

\[
\begin{align*}
\text{(Query I)} & \quad \text{if } \mathbf{w} \neq \bot: \quad \text{sk} = \mathbf{w}' \mathbf{v}, \quad \text{st}' = (\mathbf{w}, \bot, v_1, \ldots, v_{J'}, \mathbf{v}); \\
\text{(Query II)} & \quad \text{if } \mathbf{w} = \bot: \quad \text{sk} = ct' \mathbf{v} - z, \quad \text{st}' = (\bot, ct').
\end{align*}
\]

• SimEnc$(\text{st}, z_1, \ldots, z_J)$ takes as input the state $\text{st}$ and a list $z_1, \ldots, z_J$ of inner products. It parses the state $\text{st}$ as $(\mathbf{w}, \bot, v_1, \ldots, v_{J'})$. If $J \neq J'$, the algorithm outputs $\bot$ and terminates. Otherwise, it outputs the ciphertext $ct'$ as a uniformly random solution of

\[-\mathbf{w}' \mathbf{v}_j + ct' \mathbf{v}_j = z_j \quad \forall j \in [J],\]

and the updated state as $\text{st}' = (\bot, ct')$.

The following lemma shows that the simulator is perfect:

**Lemma 19.** The simulator in Construction 18 perfectly simulates the one-time pad scheme in Construction 17.

**Proof.** In the security experiment, the adversary can query arbitrarily many keys and one challenge ciphertext, both with vectors of its choice. The keys and ciphertext are either honestly generated or simulated, and we want to show that the adversary has no advantage in distinguishing the two cases.

We consider the view of the adversary at each phase of the security experiment. Before the ciphertext challenge, the simulator is perfect since the distribution of $\mathbf{w}$ (in the real scheme) and how the secret keys are generated are the same as in the real scheme.

Once the adversary chooses the challenge vector $\mathbf{u}^*$, the distribution of $\mathbf{w}$ is uniformly random conditioned on $\mathbf{w}' \mathbf{v}_j = \mathbf{w}' \mathbf{v}_j$ for all $j$. Moreover, $\mathbf{u}^*$ is a particular solution of $\mathbf{u}' \mathbf{v}_j = z_j$ for all $j$. This implies $ct^* = (\mathbf{w} + \mathbf{u}^*)^\top$ is a uniformly random solution of $ct^* \mathbf{v}_j = \mathbf{w}' \mathbf{v}_j + z_j$, and the simulator does the same thing.

After the ciphertext challenge, the secret key of $\mathbf{v}$ is

\[
\mathbf{w}' \mathbf{v} = (ct^* - (\mathbf{u}^*)^\top)\mathbf{v} = ct^* \mathbf{v} - (\mathbf{u}^*)^\top \mathbf{v},
\]

and the simulator computes the same value. $\Box$

### 3.3 Proof of Gradual Simulation Security

We first discuss the intuition of our simulator. Recall that in the scheme, the public key consists of a random matrix $\mathbf{A}$ and the projection $\mathbf{A} \mathbf{W}$ of the master secret key $\mathbf{W}$. A ciphertext can be thought of as an OTP-IPFE ciphertext using $\mathbf{s}' \mathbf{A} \mathbf{W}$ as its master secret key, which is a random combination of $\mathbf{A} \mathbf{W}$. It also includes the combination coefficients $\mathbf{s}' \mathbf{A}$, with respect to $\mathbf{W}$. MDDH assumption tells us that $\mathbf{s}' \mathbf{A} \mathbf{W}$ is a pseudorandom combination of $\mathbf{W}$, so it is indistinguishable for the ciphertext to use $\mathbf{c}^\top \mathbf{W}$ as its OTP-IPFE master secret key for a random $\mathbf{c}$, which lies outside the row space of $\mathbf{A}$ with overwhelming probability. A ciphertext in this form is using an OTP msk independent of mpk, or equivalently, a fresh instance of OTP-IPFE dedicated to this ciphertext.

A recurring technique in dual system encryption is to perform a change of variable for $\mathbf{W}$ to explicitly separate out the instance for the challenge ciphertext. We program $\mathbf{W} = \mathbf{W} + \mathbf{a}^\top \mathbf{w}'$, where $\mathbf{w}'$ is a uniformly random vector.
where $\mathbf{w}$ is an OTP-IPFE master secret key and $\mathbf{a}^\perp$ is any vector such that $\mathbf{Aa}^\perp = \mathbf{0}, \mathbf{c}^i\mathbf{a}^\perp = 1$. (This step is known as using the parameter hiding property.) Under this change of variable, keys and ciphertext become

$$\text{sk} = \mathbf{Wv} = \mathbf{Wv} + \mathbf{a}^\perp \mathbf{w}^\perp \mathbf{v}, \quad \text{ct} = \left[\mathbf{c}^i, \mathbf{c}^i \mathbf{W}^\perp + \mathbf{u}^\perp\right] = \left[\mathbf{c}^i, \mathbf{c}^i \mathbf{W} + (\mathbf{w} + \mathbf{u})^\perp\right].$$

Note that the terms in the boxes are precisely OTP-IPFE keys and ciphertext, and the terms outside the boxes does not depend on $\mathbf{u}$. Therefore, our simulator prepares an OTP-IPFE simulator instance, and samples $\mathbf{A}, \mathbf{c}, \mathbf{W}$ and computes $\mathbf{a}^\perp$ on its own. To simulate keys and ciphertext, it invokes the OTP-IPFE simulator and stitches the result. The construction below extends this idea to handle multiple simulated ciphertexts.

**Construction 20.** Adopt the notations in Construction 13 and let (OTP.SimSetup, OTP.SimKeyGen, OTP.SimEnc) be the simulator in Construction 18. The simulator for Construction 13 works as follows:

- **SimSetup($1^n, 1^T$)** takes as input the dimension $n$ and the maximum number $T$ of simulated ciphertexts. It samples

$$\mathbf{T} \leftarrow \mathbb{Z}_p^{k \times k}, \quad \mathbf{W} \leftarrow \mathbb{Z}_p^{(k+T) \times n}, \quad \mathbf{P} = (\mathbf{a}_1^\perp, \ldots, \mathbf{a}_T^\perp, \mathbf{p}_1, \ldots, \mathbf{p}_k) \leftarrow \mathbb{Z}_p^{(k+T) \times (k+T)}$$

conditioned on $\mathbf{P}$ being invertible. The algorithm prepares $T$ instance of OTP-IPFE simulator: $\text{st}_t \leftarrow \text{OTP.SimSetup}(p, 1^n)$ for $t \in [T]$. It sets

$$(\mathbf{c}_1, \ldots, \mathbf{c}_{T-1}, \mathbf{c}_T) = (\mathbf{P}^\perp)^{-1} \begin{pmatrix} \mathbf{I}_{T-1} \\ \mathbf{1}^T \end{pmatrix}$$

and outputs $\text{mpk} = \left[\mathbf{A}, \mathbf{AW}\right]$ and $\text{st} = (0, \mathbf{a}_1^\perp, \left[\mathbf{c}_1^i\right], \text{st}_1, \ldots, \mathbf{a}_T^\perp, \left[\mathbf{c}_T^i\right], \text{st}_T)$.

Note: Conceptually, we program $\mathbf{W} = \mathbf{W} + \mathbf{a}^\perp \mathbf{w}_1^\perp + \cdots + \mathbf{a}_T^\perp \mathbf{w}_T^\perp$, where $\mathbf{w}_t$ is the OTP-IPFE master secret key for the $t$th simulated ciphertext.\(^{13}\) By definition, $\mathbf{Aa}_i^\perp = \mathbf{0}, \mathbf{c}_i^i\mathbf{a}_i^\perp = 0$ for $i \neq i'$ and $\mathbf{c}_i^i\mathbf{a}_i^\perp = 1$. The first component of $\text{st}$ keeps track of the number of simulated ciphertext at any point.

- **SimKeyGen(st, $t$, $\mathbf{v}, z_1, \ldots, z_T$)** takes as input the state $\text{st}$, a vector $\mathbf{v}$, and a list of inner products $z_1, \ldots, z_T$. It parses $\text{st}$ as $(I', \mathbf{a}_1^\perp, \left[\mathbf{c}_1^i\right], \text{st}_1, \ldots, \mathbf{a}_T^\perp, \left[\mathbf{c}_T^i\right], \text{st}_T)$. If $I \neq I'$, the algorithm outputs $\perp$ and terminates. Otherwise, it runs

$$\text{for } t \leq I': \quad (\text{OTP.sk}_t, \text{st}'_t) \leftarrow \text{OTP.SimKeyGen(st}_t, \mathbf{v}, z_t);$$

$$\text{for } t > I': \quad (\text{OTP.sk}_t, \text{st}'_t) \leftarrow \text{OTP.SimKeyGen(st}_t, \mathbf{v}).$$

The algorithm outputs

$$\text{sk} = \mathbf{Wv} + \mathbf{a}_T^\perp \text{OTP.sk}_1 + \cdots + \mathbf{a}_T^\perp \text{OTP.sk}_T, \quad \text{ct} = (I', \mathbf{a}_1^\perp, \left[\mathbf{c}_1^i\right], \text{st}_1', \ldots, \mathbf{a}_T^\perp, \left[\mathbf{c}_T^i\right], \text{st}_T').$$

\(^{13}\) The programming of $\text{msk}$'s is only conceptual because they are in fact simulated.
The secret key for \(v\) is

\[
\text{SimEnc}(\text{st}, z_1, \ldots, z_J) \text{ takes as input the state } \text{st} \text{ and a list of inner products } z_1, \ldots, z_J. \text{ It parses } \text{st} \text{ as } (I, a_i, [c_i], \text{st}_1, \ldots, a_T, [c_T], \text{st}_T). \text{ If } I = T, \text{ the algorithm outputs } \perp \text{ and terminates. Otherwise, it runs}
\]

\[
(\text{OTP}.\text{ct}, \text{st}_t) \leftarrow \text{OTP.\text{SimEnc}(\text{st}_{t+1}, z_1, \ldots, z_J),} \quad \text{st}_t' \leftarrow \text{st}_t \text{ for } t \neq I + 1,
\]

and outputs

\[
\begin{align*}
\text{ct} &= (\langle c_i^1 \rangle, \langle c_i^{t+1} \rangle \bar{W} + [\text{OTP.}\text{ct}]), \\
\text{st}' &= (I + 1, a_i, [c_i], \text{st}_i', \ldots, a_T, [c_T], \text{st}_T).
\end{align*}
\]

Proving Theorem 14 amounts to proving the following two claims:

**Claim 21.** The simulator in Construction 20 satisfies key simulation security.

**Claim 22.** Suppose in Construction 13, MDDH\(_k\) holds in \(G\), then the simulator in Construction 20 satisfies \(T\)-ciphertext simulation security for all \(T = \text{poly}(\lambda)\).

**Proof (Claim 21).** Recall that in \(\text{Exp}_{\text{real}}\) and \(\text{Exp}_{\text{sim}}\), the adversary only gets the master public key and the secret keys. We show that their distributions are identical via these hybrids:

1. \(H_1\) proceeds identically as \(\text{Exp}_{\text{real}}\), except that \(A\) is sampled as in \(\text{Exp}_{\text{sim}}\).
2. \(H_2\) proceeds identically as \(H_1\), except that we set

\[
W = \bar{W} + a_i w_i^1 + \cdots + a_T w_T^i
\]

for uniformly random \(\bar{W}, w_1, \ldots, w_T\).

In both \(\text{Exp}_{\text{real}}\) and \(\text{Exp}_{\text{sim}}\), the matrix \(A\) is a uniformly random matrix, so \(\text{Exp}_{\text{real}} \equiv H_1\). We also have \(H_1 \equiv H_2\) as the change of variable does not alter the distribution of \(W\) and \(AW = \bar{W}W\). In \(H_2\), the secret key for \(v\) is

\[
Wv = \bar{W}v + a_i w_i^1 v + \cdots + a_T w_T^i v.
\]

In \(\text{Exp}_{\text{sim}}\), the terms \(w_i^1 v\) are replaced by their simulation. Since the OTP-IPFE simulator is perfect, we conclude \(H_2 \equiv \text{Exp}_{\text{sim}}\).

**Proof (Claim 22).** Recall that in \(\text{Exp}_{b}^{\text{GS}}\), the adversary receives the simulated \(\text{mpk}\), can query for arbitrarily many simulated secret keys and at most \(T - 1\) simulated ciphertexts, and submits one challenge vector to obtain either a challenge ciphertext encrypted using the simulated \(\text{mpk}\), or a simulated challenge ciphertext. We want to show that the the two cases are indistinguishable.

To simulate the challenge ciphertext, we need to first switch it from using \(s'A\) to \(c^i\). At first sight, it seems that the reduction cannot know \(a^i\)'s (since it receives \(\langle A \rangle\) as an MDDH\(_k\) challenge) thus cannot simulate \emph{non-challenge} ciphertexts. This is why the simulator samples \(A\) in a fashion similar to the reduction of MDDH\(_{k,k+T}\) to MDDH\(_k\), which originates from a technique in [CGKW18].

We go through these hybrids:
H₁ proceeds identically to Exp⁰_{T-GS}, except that the T-th instance of OTP-IPFE simulator is replaced by an instance of OTP-IPFE itself and c_T is removed. In particular, the state of the simulator is

\[ st = (I, aₙ^+, [c]_T^i, \ldots, a_{T-1}^+, [c]_{T-1}^i, st_{T-1}, a_T^+, [w]_T^*) \]

and a simulated secret key for v is

\[ sk = \bar{W}v + aₙ^+ \text{OTP.sk}_1 + \ldots + a_{T-1}^+ \text{OTP.sk}_{T-1} + a_T^+ [w]_T^*v \]

Since the adversary can query at most T - 1 simulated ciphertexts (excluding the challenge ciphertext), this instance is never used to simulate a ciphertext in Exp⁰_{T-GS}, and it does not matter if we remove c_T. The transition from Exp⁰_{T-GS} to H₁ is essentially undoing simulation for the last OTP-IPFE instance. Since the OTP-IPFE simulator is perfect, Exp⁰_{T-GS} ≡ H₁.

H₂ proceeds identically to H₁, except that aₙ^+ and the last OTP-IPFE instance are removed. The state of the simulator and a simulated secret key for v are

\[ st = (I, aₙ^+, [c]_T^i, \ldots, a_{T-1}^+, [c]_{T-1}^i, st_{T-1} \text{, } [c]_{T-1}^i \text{, } st_T = \bar{W} + a_T^+ [w]_T^*v) \]

This can be interpreted as a change of variable \( \bar{W} \rightarrow \bar{W} - a_T^+ [w]_T^* \), which does not affect the distribution of \( \bar{W} \), so H₁ ≡ H₂.

H₃ proceeds identically to H₂, except that the challenge ciphertext becomes

\[ ct^* = ([c]_T^i, [c]_T^i \bar{W} + (u^*)^i) \text{ for } c \leftarrow \mathbb{Z}_p^{k+T}. \]

Claim 23. H₂ ≈ H₃ reduces to MDDHₖ in G.

H₄ proceeds identically to H₃, except that c in the challenge ciphertext becomes c_T sampled by SimSetup. Note that we are not using aₙ^+ and c_T is statistically close to c. Therefore, H₄ ≈ H₃.

H₅ proceeds identically to H₄, except that we perform a change of variable \( \bar{W} \rightarrow \bar{W} + a_T^+ [w]_T^* \). Clearly, H₄ ≡ H₅.

H₆ proceeds identically to H₅, except that we resurrect the last instance of OTP-IPFE simulator and use it to simulate the challenge ciphertext. In H₅, the challenge ciphertext and a simulated secret key for v are

\[ ct^* = ([c]_T^i, [c]_T^i \bar{W} + (w_T^* + u^*)^T) \]

\[ sk = \bar{W} + aₙ^+ \text{OTP.sk}_1 + \ldots + a_{T-1}^+ \text{OTP.sk}_{T-1} + a_T^+ [w]_T^*v. \]

The terms in boxes are OTP-IPFE ciphertexts and keys, and in H₆ they are replaced by simulation. Since the OTP-IPFE simulator is perfect, we have H₅ ≡ H₆.

We argue H₆ ≡ Exp¹_{T-GS}. Note that the only difference between the two is which instance is used to simulate which ciphertext. This amounts to permuting \((aₙ^+, [c]_T^i, st_i)\)'s. It is easy to verify that the distribution of these tuples is permutation invariant. Once we prove H₂ ≈ H₃ (Claim 23), we conclude Exp⁰_{T-GS} ≈ Exp¹_{T-GS} by hybrid argument.
Proof (Claim 23). Let \( A \) be an efficient adversary distinguishing \( H_2 \) and \( H_3 \). We construct \( B \) against the MDDH\(_k\) assumption in \( G \). Upon receiving MDDH\(_k\) challenge \([B, d']\) where \( d' = s'B \) or \( d' \) is random, \( B \) launches \( A \) and receives \( 1^n, 1^T \) from it, samples and defines

\[
\vec{W} \leftarrow Z_p^{(k+T) \times n}, \quad P = (a_1 \dot{\cdots} a_{T-1}^\perp, p_0, p_1, \dot{\cdots}, p_k) \leftarrow Z_p^{(k+T) \times (k+T)}, \\
(c_1 \dot{\cdots} c_{T-1} c A') = (P^{s_T})^{-1} \left[ 1^T, d B' \right].
\]

Here, \((d, B')\) occupies the position of \( \left[ \begin{array}{c} 1 \\ T \end{array} \right] \). \( B \) prepares \( T - 1 \) instances of OTP-IPFE simulator:

\[
st_t \leftarrow \text{OTP.SimSetup}(p, 1^n) \text{ for } t \in [T-1].
\]

It sets

\[
\text{mpk} = ([A], [A] \| \vec{W}), \quad \text{st} = (0, a_1^\perp, [c_1^\perp], st_1, \dot{\cdots}, a_{T-1}^\perp, [c_{T-1}^\perp], st_{T-1}).
\]

\( B \) answers queries from \( A \) as defined in \( H_2, H_3 \). It responds to the ciphertext challenge as

\[
c_t^* = ([c^*], [c^*] \| \vec{W} + [u^*]).
\]

Lastly, it outputs whatever \( A \) outputs.

Clearly \( B \) is efficient, produces correct distribution for \((a_t^\perp, [c_t^\perp], st_t)\)'s and \( A, \vec{W}, \) and answers queries (non-challenges) as required. If \( d' = s'B \), we have \( c^* = s'A \) and \( B \) has simulated \( H_2 \) for \( A \). If \( d' \) is random, so is \( c^* \) and \( B \) has simulated \( H_3 \) for \( A \). Therefore, their advantages are the same. If MDDH\(_k\) holds in \( G \), we have \( H_2 \approx H_3 \).

\[ \Box \]

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 \( (\text{Exp}^0_{1-\text{ABE}}) \), and is uniformly random in the other experiment \( (\text{Exp}^1_{1-\text{ABE}}) \).

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 [KW19,LL20] 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 [KW19,LL20] 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.

\[ \text{14} \text{The adversary also does not receive the potential random message.} \]
Construction 24 (CP-1-ABE). The construction is described for a fixed value of $\lambda$, and $\lambda$ 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

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

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 $(\text{mpk}, \text{msk}) \leftarrow \text{IPFE.Setu}(1^{n+1})$.
- KeyGen(\text{msk}, \text{x}) outputs $\text{sk} \leftarrow \text{IPFE.KeyGen}(\text{msk}, (1, \text{x}))$.

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(\text{mpk}, y, \mu) garbles $y$ with $\mu$ and encrypts the label functions in IPFE ciphertexts as follows:

  - if $y = f_{\neq 0}$, $\alpha \leftarrow \mu$, $\beta \leftarrow 0$;
  - if $y = f_{= 0}$, $\alpha \leftarrow \mathbb{Z}_p$, $\beta \leftarrow \mu$;
  - $(L_1, \ldots, L_m) \leftarrow \text{Garble}(f, \alpha, \beta)$,
  - for $j \in [m]$, $ict_j \leftarrow \text{IPFE.Enc}($mpk, $L_j$).

The algorithm outputs $\text{ct} = (ict_1, \ldots, ict_m)$.

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

  $$
  \text{for } j \in [m]: \quad \ell_j \leftarrow \text{IPFE.Dec}(\text{sk}, \text{ict}_j)
  $$

  $$
  \mu' = \begin{cases} 
  \frac{1}{f(x)\ell_1, \ldots, \ell_m} \text{Eval}(f, \text{x}, \ell_1, \ldots, \ell_m), & \text{if } y = f_{\neq 0}; \\
  \text{Eval}(f, \text{x}, \ell_1, \ldots, \ell_m), & \text{if } y = f_{= 0}; 
  \end{cases}
  $$

and outputs $\mu'$ as the message.

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

$$
\ell_j = L_j \left[ \frac{1}{x} \right] = L_j(x),
$$

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,

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

Theorem 25. Suppose in Construction 24, the IPFE is gradually 2-simulation-secure and the AKGS is piecewise secure, then the constructed 1-ABE is secure.
Lastly, we argue \(H_{\text{ABE}}\), the adversary receives \(mpk\) and can query one secret key for attribute \(x \in \mathbb{Z}_p^n\). In addition, it receives a challenge ciphertext tied to some policy \(y \in X\text{ABP}_p\) of its choice, encrypting either 0 (in \(\text{Exp}^0_{\text{ABE}}\)) or \(\delta \leftarrow \mathbb{Z}_p\) (in \(\text{Exp}^1_{\text{ABE}}\)). The adversary is free to ask for either the secret key or the ciphertext first. Its goal is to distinguish the two cases, and we want to show \(\text{Exp}^0_{\text{ABE}} \approx \text{Exp}^1_{\text{ABE}}\).

Suppose the garbling size of the underlying ABP of \(y\) is \(m\), we go through a series of hybrids \(\text{Exp}^b_{\text{ABE}}, H^b_{1-m,1-3}\). In these hybrids, we gradually modify the ciphertext queried by the adversary, which starts out as encrypting label functions \(L_j\) generated by AKGS Garble and ends up by encrypting simulated labels \(\ell_j \leftarrow \mathbb{Z}_p\). The proof interleaves between using the gradual simulation security of IPFE and the piecewise security of AKGS, in a fashion similar to [LL20].

The hybrids \(H^b_{1,1-3}\) changes \(ict_1\) from encrypting the label function \(L_1\) to being simulated using the reversely sampled first label \(\ell_1\):

- \(H^b_{1,1}\) proceeds identically to \(\text{Exp}^1_{\text{ABE}}\), except that we use the simulator of IPFE to generate \(mpk\) and \(sk\) for \(x\), instead of using the real IPFE algorithms. By gradual 2-simulation security of IPFE (\(\text{Exp}_\text{real} \approx \text{Exp}_\text{sim}\)), we have \(\text{Exp}^b_{\text{ABE}} \approx H^b_{1,1}\).

- \(H^b_{1,2}\) proceeds identically to \(H^b_{1,1}\), except that \(ict_1\) (encoding the label function \(L_1\) in the challenge ciphertext) is simulated. By gradual 2-simulation security of IPFE (\(\text{Exp}^0_{2-GS} \approx \text{Exp}^1_{2-GS}\)), we have \(H^b_{1,1} \approx H^b_{1,2}\). Note that in \(H^b_{1,2}\), the first label function \(L_1\) is not directly used, and only the first label \(\ell_1 = L_1(x)\) is used.

- \(H^b_{1,3}\) proceeds identically to \(H^b_{1,2}\), except that the \(\ell_1\) used to simulate \(ict_1\) (or \(isk\), whichever comes later) is reversely sampled instead of being computed from \(L_1\). By the reverse sample-pleability of AKGS, \(H^b_{1,2} \equiv H^b_{1,3}\).

Next, the hybrids \(H^b_{j,1-3}\) (for \(j = 2,\ldots,m\)) change \(ict_j\) from encrypting the label function \(L_j\) to encrypting the simulated label \(\ell_j \leftarrow \mathbb{Z}_p\), via a temporary simulated form.

- \(H^b_{j,1}\) proceeds identically to \(H^b_{1,3}\), except that \(ict_j\) for \(1 < j' < j\) encrypts \((\ell_{j'}, 0)\) for \(\ell_{j'} \leftarrow \mathbb{Z}_p\) (here, \(0\) is of the same length as \(x\)), and \(ict_j\) is simulated using \(\ell_j = L_j(x)\). By gradual 2-simulation security of IPFE (\(\text{Exp}^0_{2-GS} \approx \text{Exp}^1_{2-GS}\)), we have \(H^b_{1,3} \approx H^b_{2,1}\).

Similarly, \(H^b_{j,1} \approx H^b_{j,1}\), which should be clear once we specify \(H^b_{j,3}\).

- \(H^b_{j,2}\) proceeds identically to \(H^b_{1,1}\), except that \(\ell_j\) is sampled uniformly at random instead of being computed as \(L_j(x)\). Note that both \(H^b_{j,1}\) and \(H^b_{j,2}\) can be generated using just \(\ell_j\) and \(L_{j+1},\ldots,L_m\) from the garbling, so marginal randomness property applies and \(H^b_{j,1} \equiv H^b_{j,2}\).

- \(H^b_{j,3}\) proceeds identically to \(H^b_{j,2}\), except that \(ict_j\) is reverted to a normal ciphertext, now encrypting \((\ell_j, 0)\), where \(\ell_j \leftarrow \mathbb{Z}_p\) and \(0\) is of the same length as \(x\). By gradual 2-simulation security of IPFE (\(\text{Exp}^1_{2-GS} \approx \text{Exp}^0_{2-GS}\)), we have \(H^b_{j,2} \approx H^b_{j,3}\).

Note that in \(H^b_{j+1,1}\), the next hybrid, \(ict_{j+1}\) becomes simulated using \(\ell_{j+1} = L_{j+1}(x)\). We have \(H^b_{j+1,1} \approx H^b_{j+1,1}\) again by gradual 2-simulation security of IPFE.

Lastly, we argue \(H^b_{m,3} \equiv H^b_{m,3}\). In both hybrids, \(ict\)'s for \(1 < j \leq m\) encrypt \((\ell_j, 0)\) for \(\ell_j \leftarrow \mathbb{Z}_p\), and \(ict_1\) is simulated using \(\ell_1\), which is reversely sampled. We have two cases under the constraint \(y(x) = 0\):

\[
\text{if } y = f_0, \quad \ell_1 \leftarrow \text{RevSamp}(f, x, \delta f(x) + 0, \ell_2, \ldots, \ell_m) = \text{RevSamp}(f, x, 0, \ell_2, \ldots, \ell_m);
\]
if \( y = f = 0 \), \( \ell_1 \leftarrow \text{RevSamp}(f, x, af(x) + \delta, \ell_2, \ldots, \ell_m) \equiv \text{RevSamp}(f, x, \gamma, \ell_2, \ldots, \ell_m) \) for \( \gamma \leftarrow Z_p \).

Here, \( \delta \) is either 0 (in \( H_{m,2}^0 \)) or random (in \( H_{m,2}^1 \)), and when \( y = f = 0, f(x) \neq 0 \), the random \( \alpha \) is not used elsewhere and \( af(x) \) is a one-time pad that hides \( \delta \). In both cases, \( \ell_1 \) as well as any other value used to generate the response to the adversary contains no information about \( \delta \). Therefore, \( H_{m,2}^0 \equiv H_{m,2}^1 \). By hybrid argument, we have \( \text{Exp}_{\text{I-ABE}}^0 \approx \text{Exp}_{\text{I-ABE}}^1 \).

5 Key-Policy ABE for ABP

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

**Construction 26 (KP-ABE).** The construction is described for a fixed value of \( \lambda \), and \( \lambda \) is suppressed for brevity. Let \( G_1, G_2, G_T \) be pairing groups of prime order \( p \) for which MDDH 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 \( \mathcal{P} = \{ P_n | n \in \mathbb{N} \} \).

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
  \[
  A \leftarrow Z_p^{k \times (k+2)}, B \leftarrow Z_p^{k \times (k+1)}, \quad W \leftarrow Z_p^{(k+2) \times (k+1) \times (n+1)}, \mu \leftarrow Z_p^{k+1},
  \]
  \[
  xpk = ([B^1_1], [W(B^1 \otimes I_{n+1})]), \quad fpk = ([A^2_1], [AW^2]),
  \]
  \[
  mpk = ([\mu^1 B^1_T], xpk), \quad msk = (fpk, \mu).
  \]

Note: We explain the connection with CP-I-ABE and dual system encryption (as demonstrated in Section 1.1). The matrix \( W = (W_1, \ldots, W_{k+1}) \) consists of \( k + 1 \) master secret keys of CP-I-ABE concatenated by columns, each of shape \((k + 2) \times (n + 1)\). Its projection along a vector \( b = (b_1, \ldots, b_{k+1})^\top \) is

\[
\begin{align*}
  b_1 W_1 + \cdots + b_{k+1} W_{k+1} &= W_1 \cdot b_1 I_{n+1} + \cdots + W_{k+1} \cdot b_{k+1} I_{n+1} \\
  &= (W_1 \cdots W_{k+1}) \begin{pmatrix} b_1 I_{n+1} \\ \vdots \\ b_{k+1} I_{n+1} \end{pmatrix} = W(b \otimes I_{n+1}).
\end{align*}
\]

The matrix \( B = (b_1^1 \cdots b_{k+1}^1)^\top \) consists of all the projection vectors, and \( W(B^1 \otimes I_{n+1}) \) is the projections of \( W \) along \( B \) concatenated by columns.

- KeyGen(msk, y) garbles \( y \) with \( \mu \) as follows:
  \[
  \begin{align*}
  \text{if } y = f = 0: & \quad \alpha \leftarrow \mu, \quad \beta \leftarrow 0; \\
  \text{if } y = f = 0: & \quad \alpha \leftarrow Z_p^{k+1}, \quad \beta \leftarrow \mu; \\
  (L_1, \ldots, L_m) & \leftarrow \text{Garble}(f, \alpha, \beta).
  \end{align*}
  \]

\(^{15}\)In fact, \( H_{m,2}^0 \equiv H_{m,2}^1 \), but this requires more careful argument.
It samples $s_j \xleftarrow{\$} \mathbb{Z}_p^k$ for $j \in [m]$ and sets

$$sk_{j,1} = s_j^T [A]_2, \quad sk_{j,2} = s_j^T [AW]_2 + [L_j]_2.$$ 

The algorithm outputs $sk = (sk_{1,1}, sk_{1,2}, \ldots, sk_{m,1}, sk_{m,2})$.

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

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

$$ct_1 = [B^T]_1 r, \quad ct_2 = [W(B^T \otimes I_{n+1})]_1 \left(r \otimes \left(\frac{1}{x}\right)\right), \quad ct_3 = [\mu' B^T]_T r + g.$$ 

The algorithm outputs $ct = (ct_1, ct_2, ct_3)$.

Note: We remark that $r$ (resp. $B' r$) is the coefficients of random linear combination w.r.t. the projections (resp. the CP-ABE instances). Here, $ct_2$ corresponds to a CP-ABE key w.r.t. randomly combined master secret key $W(B' r \otimes I_{n+1})$, which is an IPFE secret key for $(\frac{1}{x})$, i.e.,

$$ct_2 = [W(B' r \otimes I_{n+1})]_1 \left(\frac{1}{x}\right).$$ 

The ciphertext consists of $2k + 3$ elements in $G_1$ and one element in $G_T$, hence is succinct.

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

\begin{align*}
\text{for } j \in [m]: \quad \ell_j & \leftarrow -sk_{j,1} ct_2 + sk_{j,2} \left( ct_1 \otimes \left(\frac{1}{x}\right)\right); \\
\mu' & \leftarrow \begin{cases} 
\frac{1}{f(x)} \text{Eval}(f, x, [\ell_1, \ldots, \ell_m]_T), & \text{if } y = f \neq 0; \\
\text{Eval}(f, x, [\ell_1, \ldots, \ell_m]_T), & \text{if } y = f = 0;
\end{cases}
\end{align*}

and outputs $ct_3 - [\mu']_T$ as the recovered message.

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

$$\ell_j = -s_j^T AW(B^T \otimes I_{n+1}) \left(r \otimes \left(\frac{1}{x}\right)\right) + (s_j^T AW + L_j^T) \left(B' r \otimes \left(\frac{1}{x}\right)\right)$$

$$= L_j^T \left(B' r \otimes \left(\frac{1}{x}\right)\right) = L_j^T (B' r \otimes I_{n+1}) \left(\frac{1}{x}\right).$$

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

$$\ell_j = L_j^T (B' r \otimes I_{n+1}) \left(\frac{1}{x}\right) = (L'_j)^\top \left(\frac{1}{x}\right) = L'_j(x).$$

By the correctness of AKGS, we have

$$\text{Eval}(f, x, \ell_1, \ldots, \ell_m) = r'B\alpha f(x) + r'B\beta.$$
In the two cases where decryption should succeed,

\[
\begin{align*}
\text{if } y = f_{\neq 0}, f(x) \neq 0: \quad \mu' &= \frac{1}{f(x)} \left( r' B \mu f(x) + r' B 0 \right) = r' B \mu; \\
\text{if } y = f_{\neq 0}, f(x) = 0: \quad \mu' &= r' B \alpha f(x) + r' B \mu = r' B \mu.
\end{align*}
\]

Therefore, in both cases, we have \( \text{ct}_3 - \left\llbracket \mu' \right\rrbracket_T = \left\llbracket \mu' B' r \right\rrbracket_T + g - \left\llbracket r' B \mu \right\rrbracket_T = g. \)

**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.\(^{16}\) Since Eval is linear in the labels, the decryption algorithm can first find \(\gamma_1, \ldots, \gamma_m \in \mathbb{Z}_p\) such that

\[
\text{Eval}(f, x, \ell_1, \ldots, \ell_m) = \sum_{j=1}^{m} \gamma_j \ell_j.
\]

The computation of \(\left\llbracket \mu' \right\rrbracket_T\) can be rewritten as

\[
\left\llbracket \mu' \right\rrbracket_T = \sum_{j=1}^{m} \gamma_j \left\llbracket \ell_j \right\rrbracket_T = \sum_{j=1}^{m} \gamma_j \left( -s k_{j,1} \text{ct}_2 + s k_{j,2} \left( ct_1 \otimes \left( \frac{1}{x} \right) \right) \right)
\]

\[
= - \left( \sum_{j=1}^{m} \gamma_j s k_{j,1} \right) \text{ct}_2 + \left( \sum_{j=1}^{m} \gamma_j s k_{j,2} \right) \left( I_{k+1} \otimes \left( \frac{1}{x} \right) \right) \text{ct}_1.
\]

Note that \(\text{ct}_1, \text{ct}_2\) 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.

**Theorem 27.** Suppose in Construction 26, MDDH\(_k\) holds in both \(G_1\) and \(G_2\), and the AKGS is piecewise secure, then the constructed scheme is IND-CPA secure.

**Proof.** (Theorem 27) Recall that in the security experiments, the adversary receives the master public key, many secret keys, and a challenge ciphertext. Suppose the queried keys are for \(y_1, \ldots, y_Q\) and the challenge ciphertext encrypts one of \(g_0, g_1\) with attribute \(x\). The constraint is \(y_q(x) = 0\) for all \(q \in [Q]\), and we want to show the adversary cannot distinguish which of \(g_0, g_1\) is encrypted.

The proof follows dual system encryption methodology, as discussed in Section 1.1. More formally, we go through hybrids \(\text{Exp}_{\text{CPA}}^b, \text{H}_1^b, \text{H}_2^b:\)

- \(\text{H}_1^b\) proceeds identically to \(\text{Exp}_{\text{CPA}}^b\), except that the challenge ciphertext becomes

\[
\text{ct}_1 = \left\llbracket \left[ \mathbf{c} \right] \right\rrbracket_1, \quad \text{ct}_2 = \left\llbracket W \left( \mathbf{c} \otimes \left( \frac{1}{x} \right) \right) \right\rrbracket_1, \quad \text{ct}_3 = \left\llbracket \mu' B \mathbf{c} \right\rrbracket_T + g_b,
\]

where \(\mathbf{c}\) is uniformly random outside the row space of \(B\).

\(^{16}\)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.
• \( \mathcal{H}_2^b \) proceeds identically to \( \mathcal{H}_1^b \), except that each queried key\(^{17} \) is garbled for \( \mu + \delta_q b^\perp \):

\[
\text{if } y_q = f_{q,a}: \quad \alpha_q \leftarrow \mu + \delta_q b^\perp, \quad \beta_q \leftarrow 0; \\
\text{if } y_q = f_{q,a}: \quad \alpha_q \leftarrow Z_p^{k+1}, \quad \beta_q \leftarrow \mu + \delta_q b^\perp; \\
(L_{q,1}, \ldots, L_{q,m}) \overset{\$}{\rightarrow} \text{Garble}(f_q, \alpha_q, \beta_q).
\]

Here, \( b^\perp \) is a fixed is vector such that \( Bb^\perp = 0 \) and \( c^\perp b^\perp = 1 \), and \( \delta_q, r^\perp \in Z_p \) is fresh per key.

The basis of indistinguishability is as follows:

**Claim 28.** \( \text{Exp}_{C}^b \approx \mathcal{H}_1^b \) reduces to \( \text{MDDH}_k \) in \( G_1 \).

**Claim 29.** \( \mathcal{H}_1^b \approx \mathcal{H}_2^b \) reduces to the security of \( \text{CP-1-ABE} \) (Construction 24 instantiated with gradually 2-simulation-secure Construction 13 in \( G_2 \)).

**Claim 30.** \( \mathcal{H}_2^b \equiv \mathcal{H}_1^b \).

Once we prove these claims, we have \( \text{Exp}_{CP}^b = \text{Exp}_{CP}^1 \) by hybrid argument.

**Proof (Claim 28).** A uniformly random \( c \) is outside the row space of \( B \) with overwhelming probability, so we can instead let \( c \) be uniformly random in \( H_2^b \). Let \( A \) be an efficient adversary distinguishing \( \text{Exp}_{CP}^b \) and \( H_2^b \), we construct \( B \) against \( \text{MDDH}_k \) in \( G_1 \). Upon receiving \( \text{MDDH}_k \) challenge \( [B,c]_i \), where either \( c' = r'B \) or \( c' \) is random, \( B \) launches \( A \), receives the attribute length \( 1^n \) from it, samples \( A, W, \mu \), and sets

\[
xpk = ([B]_1, W([B]_1 \otimes I_{n+1})), \quad \text{fpk} = ([A]_2, [AW]_2), \quad \text{msk} = (\text{fpk}, \mu).
\]

It sends \( mpk \) to \( A \) and responds to the key queries normally. To respond to the ciphertext challenge for one of \( g_0, g_1 \) with attribute \( x \), our \( B \) sets

\[
ct_1 = [c]_1, \quad ct_2 = W([c]_1 \otimes \begin{pmatrix} 1 \\ x \end{pmatrix}), \quad ct_3 = [\mu^\perp]_2 [c]_1 + g_b.
\]

Lastly, it outputs whatever \( A \) outputs.

Clearly \( B \) is efficient. Note that in \( \text{Exp}_{CP}^b \), the exponents in \( ct_2 \) are

\[
W(B' \otimes I_{n+1}) \left( r \otimes \begin{pmatrix} 1 \\ x \end{pmatrix} \right) = W(B'r \otimes \begin{pmatrix} 1 \\ x \end{pmatrix}).
\]

If \( c' = r'B \), our \( B \) has simulated \( \text{Exp}_{CP}^b \) for \( A \). If \( c' \) is random, our \( B \) has (statistically closely) simulated \( \mathcal{H}_1^b \) for \( A \). Therefore, their advantages are negligibly close. If \( \text{MDDH}_k \) holds in \( G_1 \), we have \( \text{Exp}_{CP}^b \approx \mathcal{H}_1^b \). \( \square \)

**Proof (Claim 29).** Let \( b^\perp \) be such that \( Bb^\perp = 0 \) and \( c^\perp b^\perp = 1 \) for the \( B \) used in the master public key and the \( c^\perp \) used in the challenge ciphertext. We set\(^{18} \)

\[
W = \bar{W} + (b^\perp)\bar{W}, \quad \mu_q = \mu + \delta_q b^\perp,
\]

\(^{17}\)In a typical dual system proof, the keys are switched one by one. In our case, we can change them all at once since our 1-ABE is public-key.

\(^{18}\)Components with hat will come from CP-1-ABE, and components with tilde will be sampled by the reduction algorithm.
where $\mathbf{W} \in \mathbb{Z}_p^{(k+2) \times (k+1)(n+1)}$, $\mathbf{W} \in \mathbb{Z}_p^{(k+2) \times n+1}$, $\mathbf{z} \in \mathbb{Z}_p^{k+1}$, and either $\delta_q = 0$ (in $H_1$) or $\delta_q \in \mathbb{Z}_p$ (in $H_2$). Recall that to create a secret key for $y_q$, we need to garble $y_q$ with $\mu_q$. For this, we set

$\mathbf{mpk} = \mathbf{R_q} = \mathbf{L_q} + \mathbf{b^+} \otimes \mathbf{L_q}$. 

With the above preparation, we can describe the reduction. Let $\mathcal{A}$ be an efficient adversary distinguishing $H_1^p$ and $H_2^p$, we construct $\mathcal{B}$ against the security of CP-1-ABE (Construction 24 instantiated with gradually 2-simulation-secure Construction 13). $\mathcal{B}$ launches $\mathcal{A}$, receives the attribute length $l^*$ from it, and forwards $l^*$ to the security experiment of CP-1-ABE. $\mathcal{B}$ receives $\mathbf{mpk} = ([\mathbf{A}]_2, [\mathbf{AW}]_2)$. It samples $\mathbf{W}, \mathbf{B}, \mathbf{c}, \mathbf{\mu}$ and computes $\mathbf{b^+}$ as specified above. Note that

$$\mathbf{W}(\mathbf{B}^T \otimes \mathbf{I}_{n+1}) = (\mathbf{W} + (\mathbf{b^+})^T \otimes \mathbf{W})(\mathbf{B}^T \otimes \mathbf{I}_{n+1})$$

$$= \mathbf{W}(\mathbf{B}^T \otimes \mathbf{I}_{n+1}) + (\mathbf{b^+})^T \mathbf{B}^T \otimes \mathbf{W} = \mathbf{W}(\mathbf{B}^T \otimes \mathbf{I}_{n+1}) + (\mathbf{b^+})^T \mathbf{B}^T \otimes \mathbf{W} = \mathbf{W}(\mathbf{B}^T \otimes \mathbf{I}_{n+1}).$$

So $\mathcal{B}$ sets

$$\mathbf{xpk} = ([\mathbf{B}]_1, [\mathbf{W}(\mathbf{B}^T \otimes \mathbf{I}_{n+1})]_1), \quad \mathbf{mpk} = ([\mathbf{\mu}^T \mathbf{B}]^T, \mathbf{xpk}).$$

It sends $\mathbf{mpk}$ to $\mathcal{A}$ and starts to respond to its queries and challenges. To respond to a secret key query for $y_q$, our $\mathcal{B}$ computes $\mathbf{L_q}, \ldots, \mathbf{L_{q,m}}$ as specified above, and queries the security experiment of CP-1-ABE for a ciphertext for $y_q$, which consists of

$$\mathbf{ict}_{q,j} = ([\mathbf{s}_{q,j}^T \mathbf{A}]_2, [\mathbf{s}_{q,j}^T \mathbf{AW} + \mathbf{L}_{q,j}^T]_2).$$

$\mathcal{B}$ stitches the response as $\mathbf{sk}_{q,j,1} = [\mathbf{s}_{q,j}^T \mathbf{A}]_2$ and

$$\mathbf{sk}_{q,j,2} = [\mathbf{s}_{q,j}^T \mathbf{AW} + \mathbf{L}_{q,j}^T]_2 = [\mathbf{s}_{q,j}^T \mathbf{A}(\mathbf{W} + (\mathbf{b^+})^T \otimes \mathbf{W}) + \mathbf{L}_{q,j} + (\mathbf{b^+})^T \otimes \mathbf{L}_{q,j}]_2$$

$$= [\mathbf{s}_{q,j}^T \mathbf{A}\mathbf{W} + [\mathbf{L}_{q,j}]_2 + (\mathbf{b^+})^T \otimes [\mathbf{s}_{q,j}^T \mathbf{A}\mathbf{W} + [\mathbf{L}_{q,j}]_2].$$

To respond to the ciphertext challenge for one of $g_0, g_1$ with attribute $x$, our $\mathcal{B}$ queries the security experiment of CP-1-ABE to obtain a secret key for $x$, which is $\mathbf{W}(\frac{1}{x})$. It sets $\mathbf{ct}_1 = [\mathbf{c}]_1$, $\mathbf{ct}_3 = [\mathbf{\mu}^T \mathbf{c}]^T + g_b$, and stitches $\mathbf{ct}_2$ by

$$\mathbf{W}(\mathbf{c} \otimes \frac{1}{x}) = (\mathbf{W} + (\mathbf{b^+})^T \otimes \mathbf{W})(\mathbf{c} \otimes \frac{1}{x})$$

$$= \mathbf{W}(\mathbf{c} \otimes \frac{1}{x}) + (\mathbf{b^+})^T \mathbf{c} \otimes \mathbf{W}(\frac{1}{x})$$

$$(\mathbf{c}^T \mathbf{b^+} = 1) = \mathbf{W}(\mathbf{c} \otimes \frac{1}{x}) + \mathbf{W}(\frac{1}{x}).$$
Lastly, B outputs whatever A outputs.

Clearly B is efficient and only queries one secret key in the security experiments of CP-1-ABE. If B is in (the multi-ciphertext version of) Exp_{\text{ABE}}^0 (\delta_q = 0 \text{ inside } \mathbf{I}_{q,j}), A is in H^b_1. If B is in Exp_{\text{ABE}}^1 (\delta_q \leftarrow \mathbb{Z}_p \text{ inside } \mathbf{I}_{q,j}), A is in H^b_2. Therefore, their advantages are the same. Since the CP-1-ABE is secure, we have $H^b_0 = H^b_1$.

Proof (Claim 30). Inspecting the appearances of $\mu$ in $H^b_2$ and $H^b_1$, the master public key leaks $\mu' B'$, the secret keys leak $\mu + \delta_q b^\perp$, and the challenge ciphertext leaks $\mu' c$. Consider the change of variable $\mu = \bar{\mu} + \delta_q b^\perp$ for $\bar{\mu} \leftarrow \mathbb{Z}_p^{k+1}$ and $\delta \leftarrow \mathbb{Z}_p$, then the master public key leaks $\bar{\mu}' B'$, the secret keys leak $\bar{\mu} + (\delta_q + \delta) b^\perp$, and in the challenge ciphertext $\delta$ appears as

$$\mu' c + g_b = \bar{\mu}' c + \delta + g_b.$$ 

Since $\delta$ is perfectly hidden elsewhere, it perfectly hides $g_b$ (thus $b$). We conclude that $H^b_2 \equiv H^b_1$.

6 Ciphertext-Policy ABE for ABP

In this section, we show how to construct CP-ABE using exactly the same method for building KP-ABE. More specifically, in a first step, we obtain KP-1-ABE, which is simple as our KP-ABE also serves as a KP-1-ABE. In the second step, we convert KP-1-ABE to a CP-ABE using dual system encryption, similar to the transformation from CP-1-ABE to KP-ABE.

6.1 KP-1-ABE

We modify Construction 26 to obtain our KP-1-ABE, based on which we construct our CP-ABE. In Setup($1^n$), we change the master public key:

$$A \leftarrow \mathbb{Z}_p^{k \times (k+2)}, \quad B \leftarrow \mathbb{Z}_p^{k \times (k+1)}, \quad W \leftarrow \mathbb{Z}_p^{(k+2) \times (k+1) (n+1)}, \quad \mu \leftarrow \mathbb{Z}_p^{k+1},$$

$$\text{xpk} = ([B^\perp], [W (B^\perp \otimes I_{n+1})]), \quad \text{mpk} = ([\mu' B^\perp], \text{xpk}, [\text{fpk}]), \quad \text{msk} = (\text{fpk}, \mu).$$

KeyGen(msk, y) is the same as in KP-ABE. In Enc(mpk, x, g), the message $g$ is now in $\mathbb{Z}_p$, and the ciphertext $ct = (ct_1, ct_2, ct_3)$ is generated with $r \leftarrow \mathbb{Z}_p^b$ as

$$ct_1 = [B^\perp]_1 r, \quad ct_2 = [W (B^\perp \otimes I_{n+1})]_1 \left( r \otimes \frac{1}{x} \right), \quad ct_3 = [\mu' B^\perp]_1 r + [g]_1.$$

In Dec, the message is recovered as $[g]_T$.

We change the encoding of $\mu' B'$ and the last component of the ciphertext from $G_T$ to $G_1$ so that we do not “use up” the pairing at encryption time. We also include fpk in mpk, which will be useful for reductions in CP-ABE. It is straightforward to adapt the security proof of KP-ABE for these changes.

19The scheme in fact enjoys multi-key security, but we only need its 1-key security.
6.2 CP-ABE

We use dual system encryption to lift KP-1-ABE to CP-ABE. Recall that the transformation sets up the system by preparing \( k + 1 \) instances of 1-ABE and \( k + 1 \) random messages (encrypted under 1-ABE and used to generate encapsulated keys in ABE), and publishing a random projection of them. In the conversion from CP-1-ABE to KP-ABE, this is reflected the fact that \( \mathbf{W} \) is enlarged from \((k + 2) \times (n + 1)\) to \((k + 2) \times (k + 1)(n + 1)\) and a fresh \( \mathbf{\mu} \) is sampled. However, the matrix \( \mathbf{A} \) is not enlarged and is shared among all the \( k + 1 \) instances. More precisely, it is the master secret key being duplicated.

It turns out that the master secret key of KP-1-ABE is essentially the messages \( \mathbf{\mu} \), not including \( \mathbf{fpk} \). This is reflected by the fact that \( \mathbf{fpk} \) is also included in the master public key. Therefore, in CP-ABE, preparing \( k + 1 \) instances of 1-ABE just means \( \mathbf{\mu} \in \mathbb{Z}_p^{k+1} \) is enlarged to \( \mathbf{U} \leftarrow \mathbb{Z}_p^{(k+1) \times (k+1)} \)

Below, we call the set of messages \( \mathbf{\mu} \) and sets \( \mathbf{fpk} \) and \( \mathbf{fpk} \) to \( \mathbf{fpk} \) as follows:

\[
\mathbf{\mu} \in \mathbb{Z}_p^{k+1}, \quad \mathbf{fpk} = (\mathbb{A}, \mathbb{W}), \quad \mathbf{fpk} = (\mathbb{A}, \mathbb{W}).
\]

Construction 31. The construction is described for a fixed value of \( \lambda \), and \( \lambda \) is suppressed for brevity. Let \( \mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3 \) be pairing groups of prime order \( p \) for which MDDH holds in \( \mathcal{G}_1, \mathcal{G}_2, \) and let (Garble, Eval) be an AKGS for ABP. We construct an ABE for message space \( \mathcal{G}_3 \) and predicate space

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

The scheme works as follows:

- Setup(1^n) takes as input the attribute length (i.e., \( \mathcal{P}_n \) is represented by \( 1^n \)). It samples and sets

\[
\mathbf{A} \leftarrow \mathbb{Z}_p^{k \times (k+2)}, \quad \mathbf{B} \leftarrow \mathbb{Z}_p^{k \times (k+1)}, \quad \mathbf{D} \leftarrow \mathbb{Z}_p^{k \times (k+1)},
\]

\[
\mathbf{W} \leftarrow \mathbb{Z}_p^{(k+2) \times (k+1)(n+1)}, \quad \mathbf{U} \leftarrow \mathbb{Z}_p^{(k+1) \times (k+1)}, \quad \mathbf{V} \leftarrow \mathbb{Z}_p^{k+1},
\]

\[
\mathbf{xpk} = (\mathbb{B}[1], \mathbb{W}(\mathbb{B} \otimes \mathbb{I}_{n+1}[1]), \quad \mathbf{fpk} = (\mathbb{A}, \mathbb{W}),
\]

\[
\mathbf{mpk} = (\mathbb{D}, \mathbb{DU}, \mathbb{DV}, \mathbf{fpk}), \quad \mathbf{msk} = (\mathbb{UB}[1], \mathbf{xpk}, \mathbf{V}).
\]

- KeyGen(msk, \( x \)) samples \( \mathbf{r} \leftarrow \mathbb{Z}_p^k \) and sets

\[
\mathbf{sk}_1 = [\mathbb{B}[1], \mathbf{r}], \quad \mathbf{sk}_2 = [\mathbb{W}(\mathbb{B} \otimes \mathbb{I}_{n+1}[1]), \mathbf{r} \otimes (1)], \quad \mathbf{sk}_3 = [\mathbf{V}, \mathbb{UB}[1], \mathbf{r}].
\]

It outputs \( \mathbf{sk} = (\mathbf{sk}_1, \mathbf{sk}_2, \mathbf{sk}_3) \).

- Enc(mpk, \( \gamma, g \)) samples \( \mathbf{t} \leftarrow \mathbb{Z}_p^k \) and sets

\[
[\mathbb{u}]_2 = \mathbf{t} \mathbb{DU}, \quad \mathbf{ct}_0,1 = \mathbf{t} \mathbb{D}, \quad \mathbf{ct}_0,2 = \mathbf{t} \mathbb{DV} + \mathbf{g}.
\]

It garbles \( \mathbf{y} \) with \( [\mathbb{u}]_2 \) as follows:

\[
\text{if } y = f_{>0}: \quad \alpha \leftarrow \mathbf{u}, \quad \beta \leftarrow 0;
\]

\[
\text{if } y = f_{=0}: \quad \alpha \leftarrow \mathbb{Z}_p^{k+1}, \quad \beta \leftarrow \mathbf{u};
\]

\[
[\mathbb{L}_1, \ldots, \mathbb{L}_m]_2 \leftarrow \text{Garble}(\mathbb{f}, [\mathbb{\alpha}], [\mathbb{\beta}]).
\]

The algorithm then samples \( \mathbf{s}_j \leftarrow \mathbb{Z}_p^k \) for \( j \in [m] \) and sets

\[
\mathbf{ct}_{j,1} = \mathbf{s}_j \mathbb{A}, \quad \mathbf{ct}_{j,2} = \mathbf{s}_j \mathbb{AW} + [\mathbb{L}_j].
\]

It outputs \( \mathbf{ct} = (\mathbf{ct}_{0,1}, \mathbf{ct}_{0,2}, \mathbf{ct}_{1,1}, \mathbf{ct}_{1,2}, \ldots, \mathbf{ct}_{m,1}, \mathbf{ct}_{m,2}) \).
\(\cdot\) Dec\(\left(\text{sk}, \mathbf{x}, \mathbf{ct}, y\right)\) first checks whether \(y(\mathbf{x}) = 1\). If not, it outputs \(\bot\) and terminates. Otherwise, it parses \(\text{sk}, \text{ct}\) as defined in KeyGen, Enc, computes

\[
\text{for } j \in [m]: \quad \ell_j = -\text{ct}_{j,1}\text{sk}_2 + \text{ct}_{j,2}\left(\text{sk}_1 \otimes \frac{1}{\mathbf{x}}\right);
\]

\[
\left[\mu'\right] = \begin{cases} 
\frac{1}{f(\mathbf{x})}\text{Eval}(f, \mathbf{x}, \ell_1, \ldots, \ell_m), & \text{if } y \neq 0; \\
\text{Eval}(f, \mathbf{x}, \ell_1, \ldots, \ell_m), & \text{if } y = 0;
\end{cases}
\]

and outputs \(\text{ct}_{0,2} + \left[\mu'\right] - \text{ct}_{0,1}\text{sk}_3\) as the recovered message.

Note: We show that the scheme is correct. Note that \(\text{sk}_1, \text{sk}_2\) of this scheme are \(\text{ct}_1, \text{ct}_2\) in Construction 26 (KP-ABE), and \(\text{ct}_{j,1}, \text{ct}_{j,2}\) correspond to \(\text{sk}_{j,1}, \text{sk}_{j,2}\) for \(j \in [m]\), except that \(\text{ct}_{j,1}, \text{ct}_{j,2}\) are generated with the master secret \(\mu\) in KP-ABE replaced by \(U'D't\). By the same argument for KP-ABE correctness, we have \(\mu' = r^tBU'D't\) if decryption should succeed. Therefore,

\[
\text{ct}_{0,2} + \left[\mu'\right] - \text{ct}_{0,1}\text{sk}_3 = \left[t'Dv\right] + g + \left[r^tBU'D't\right]_T - \left[t'D\right]_2(v + UB'r)_1 = g.
\]

Similar to our KP-ABE, the number of pairing operations in decryption can be reduced to \(3k + 4\).

**Theorem 32.** Suppose in Construction 31, MDDH\(_k\) holds in both \(G_1\) and \(G_2\), and the AKGS is piecewise secure, then the constructed scheme is IND-CPA secure.

Similar to the proof for KP-ABE, we first switch \(t'D\) in the challenge ciphertext to random \(c't\). This corresponds to a fresh \(u' = c'U\) used as the \(\mu\) in KP-1-ABE and a fresh encapsulated key \(c'v\) encrypted under this KP-1-ABE instance. By KP-1-ABE security, \(c'v\) is hidden, which in turn hides the message in CP-ABE.

**Proof (Theorem 32).** We go through hybrids \(\text{Exp}^b_{\text{CPA}}, \text{H}^b_1, \text{H}^b_2:\)

- \(\text{H}^b_1\) proceeds identically to \(\text{Exp}^b_{\text{CPA}}\), except that when generating the challenge ciphertext, we set

\[
\left[u'\right] = \left[c'\right]U, \quad \text{ct}_{0,1} = \left[c'\right]_2, \quad \text{ct}_{0,2} = \left[c'\right]v + gb,
\]

where \(\mathbf{c}\) is uniformly random outside the row space of \(\mathbf{D}\) and the rest of the procedure to generate the ciphertext proceeds normally.

- \(\text{H}^b_2\) proceeds identically to \(\text{H}^b_1\), except that in each queried key,

\[
\text{sk}_3 = \left[v + \delta_d\mathbf{d}^\perp + UB'r\right]_1.
\]

Here, \(\mathbf{d}^\perp\) is a fixed vector such that \(\mathbf{Dd}^\perp = \mathbf{0}\) and \(\mathbf{c}'\mathbf{d}^\perp = 1\), and \(\delta_d \in \mathbb{Z}_p\) is fresh per key.

The basis of indistinguishability is as follows:

**Claim 33.** \(\text{Exp}^b_{\text{CPA}} \approx \text{H}^b_1\) reduces to MDDH\(_k\) in \(G_2\).

**Claim 34.** \(\text{H}^b_1 \approx \text{H}^b_2\) reduces to the security of KP-1-ABE (Section 6.1).

**Claim 35.** \(\text{H}^b_0 \equiv \text{H}^b_2\).

Once we prove these claims, we have \(\text{Exp}^0_{\text{CPA}} \approx \text{Exp}^1_{\text{CPA}}\) by hybrid argument. \(\Box\)
Proof (Claim 33). This is analogous to the proof of Claim 28. Let \( \mathcal{A} \) be an efficient adversary distinguishing \( \text{Exp}_{\text{CPA}}^b \) and \( H_1^b \), we construct \( \mathcal{B} \) against MDDH\(_k^b \) in \( G_2 \). Upon receiving MDDH\(_k^b \) challenge \( \left[ D, c' \right]_2 \) where either \( c' = t'D \) or \( c' \) is random, \( \mathcal{B} \) launches \( \mathcal{A} \), receives the attribute length \( 1^n \) from it, sets up the system using \( \left[ D \right]_2 \) and sampling and computing everything else as required. It sends mpk to \( \mathcal{A} \) and responds to the key queries normally. To respond to the ciphertext challenge for one of \( g_0, g_1 \) with policy \( y \), our \( \mathcal{B} \) sets

\[
\left[ u' \right]_2 = \left[ c' \right]_2 \mathbf{U}, \quad \text{ct}_{0,1} = \left[ c' \right]_2, \quad \text{ct}_{0,2} = \left[ c' \right]_2 \| v \|_1 + g_b,
\]

and finishes the rest of the encryption procedure from there. Lastly, it outputs whatever \( \mathcal{A} \) outputs.

Clearly \( \mathcal{B} \) is efficient. If \( c' = t'D \), our \( \mathcal{B} \) has simulated \( \text{Exp}_{\text{CPA}}^b \) for \( \mathcal{A} \). If \( c' \) is random, and \( \mathcal{B} \) has (statistically closely) simulated \( H_1^b \) for \( \mathcal{A} \). Their advantages are negligibly close, and if MDDH\(_k^b \) holds in \( G_2 \), we have \( \text{Exp}_{\text{CPA}}^b \approx H_1^b \).

\( \square \)

Proof (Claim 34). This is analogous to the proof of Claim 29. Let \( d^+ \) be such that \( Dd^+ = \mathbf{0} \) and \( c'd^+ = 1 \) for \( D \) in the master public key and \( c' \) in the challenge ciphertext. We set

\[
\mathbf{U} = \bar{\mathbf{U}} + d^+ \mathbf{\mu}^T, \quad \bar{\mathbf{U}} \triangleq Z_p^{(k+1) \times (k+1)}, \quad \mathbf{\mu} \triangleq Z_p^{k+1}, \quad \nu_q = v + \delta_q d^+,
\]

where \( \mathbf{\mu} \) comes from KP-1-ABE, and \( \delta_q = 0 \) or \( \delta_q \) \( \triangleq Z_p \) for each key queried.

Let \( \mathcal{A} \) be an efficient adversary distinguishing \( H_1^b \) and \( H_2^b \), we construct \( \mathcal{B} \) against the security of KP-1-ABE (Section 6.1). \( \mathcal{B} \) launches \( \mathcal{A} \), receives the attribute length \( 1^n \) from it, and forwards \( 1^n \) to the security experiment of KP-1-ABE. It receives \( \text{mpk} = (\left[ B \right]_1, \text{xpk}, \text{fpk}) \), where

\[
\text{xpk} = (\left[ B \right]_1, \left[ W(B^T \otimes L_1) \right]_1), \quad \text{fpk} = (\left[ A \right]_2, \left[ AW \right]_2).
\]

\( \mathcal{B} \) samples \( D, c, \bar{\mathbf{U}}, v \) and computes \( d^+ \). Since

\[
\mathbf{D}U = D\bar{\mathbf{U}} + Dd^+ \mathbf{\mu} = D\bar{\mathbf{U}},
\]

\( \mathcal{B} \) sets \( \text{mpk} = (\left[ D \right]_2, \left[ D\bar{\mathbf{U}} \right]_2, \left[ Dv \right]_T, \text{fpk}) \), sends it to \( \mathcal{A} \), and starts to respond to the queries and the challenge. To respond to a key query for \( x_q \), our \( \mathcal{B} \) queries the security experiment of KP-1-ABE for a ciphertext

\[
\tilde{\text{ct}}_{q,1} = \left[ B^T r \right]_1, \quad \tilde{\text{ct}}_{q,2} = \left[ W \left( B^T r \otimes \left( \frac{1}{x_q} \right) \right) \right]_1, \quad \tilde{\text{ct}}_{q,3} = \left[ \mu^T B^T r + \delta_q \right]_1.
\]

It sets \( \text{sk}_{q,1} = \tilde{\text{ct}}_{q,1}, \text{sk}_{q,2} = \tilde{\text{ct}}_{q,2} \) and stitches \( \text{sk}_{q,3} \) by

\[
\text{sk}_{q,3} = \left[ v + UB^T r \right]_1 = \left[ v + \delta_q d^+ + \bar{U}B^T r + d^+ \mu^T B^T r \right]_1 = \left[ v \right]_1 + \bar{U}\left[ B^T r \right]_1 + d^+ \left[ \mu^T B^T r + \delta_q \right]_1.
\]

To respond to the ciphertext challenge for one of \( g_0, g_1 \) with policy \( y \), our \( \mathcal{B} \) queries the security experiment of KP-1-ABE for a secret key for \( y \), receiving

\[
\tilde{\text{sk}}_{j,1} = \left[ s_j^T A \right]_2, \quad \tilde{\text{sk}}_{j,2} = \left[ s_j^T AW + \bar{L}_j \right]_2,
\]

where \( y \) is garbled with \( \mu \) to get \( \bar{L}_1, \ldots, \bar{L}_j \). Note that

\[
u = c'U = c'\bar{U} + c'd^+ \mathbf{\mu} = c'\bar{U} + \mathbf{\mu},
\]

36
so $B$ can garble $y$ with $c'\tilde{U}$ to obtain $\tilde{L}_1, \ldots, \tilde{L}_m$ and stitch the response as

$$ct_{0,1} = \llbracket c' \rrbracket_2, \quad ct_{0,2} = \llbracket c'v \rrbracket_T + gb, \quad ct_{j,1} = \hat{s}k_{j,1}, \quad ct_{j,2} = \hat{s}k_{j,2} + \llbracket \tilde{L}_j \rrbracket_2.$$  

The linearity of Garble guarantees the result is well-distributed. Lastly, $B$ outputs whatever $A$ outputs.

Clearly $B$ is efficient and queries at most one secret key in the security experiments of KP-$1$-\text{ABE}. If $B$ is in $\text{Exp}^0_{1\text{-ABE}}$, it has simulated $H_1^b$ for $A$. Otherwise, $B$ is in $\text{Exp}^1_{1\text{-ABE}}$ and has simulated $H_1^b$ for $A$. Therefore, the two have the same advantage. By KP-$1$-\text{ABE} security, $H_1^b \equiv H_2^b$.  

\textbf{Proof (Claim 35).} This is analogous to the proof of Claim 30. Observe that $v$ appears in $\text{mpk}$ as $Dv$, in $\text{sk}$’s as $v + \delta_qd$, and in the challenge ciphertext as $c'v$. Consider the change of variable $v = \tilde{v} + \delta d$, then in $\text{mpk}$ we only use $D\tilde{v}$, in $\text{sk}$’s we only use $\tilde{v} + (\delta + \delta)d$, and in the challenge ciphertext the message appears as

$$c'v + gb = c'\tilde{v} + \delta + gb,$$

which means $gb$ (thus $b$) is perfectly hidden. We conclude that $H_2^b \equiv H_2^b$.  

\textbf{Acknowledgments.} The authors were supported by NSF grants\textsuperscript{20} CNS-1528178, CNS-1929901, CNS-1936825 (CAREER). The authors thank the anonymous reviewers for their valuable comments.  

\textbf{References}


\textsuperscript{20}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.


