Compact Adaptively Secure ABE from k-Lin: Beyond NC^1 and towards NL

Huijia Lin Ji Luo

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

February 2020

Abstract

We present a new general framework for constructing *compact* and *adaptively secure* attribute-based encryption (ABE) schemes from k-Lin in asymmetric bilinear pairing groups. Previously, the only construction [Kowalczyk and Wee, Eurocrypt '19] that simultaneously achieves compactness and adaptive security from static assumptions supports policies represented by *Boolean formulae*. Our framework enables supporting more expressive policies represented by *arithmetic branching programs*.

Our framework extends to ABE for policies represented by uniform models of computation such as Turing machines. Such policies enjoy the feature of being applicable to attributes of arbitrary lengths. We obtain the first compact adaptively secure ABE for deterministic and non-deterministic finite automata (DFA and NFA) from k-Lin, previously unknown from any static assumptions. Beyond finite automata, we obtain the first ABE for large classes of uniform computation, captured by deterministic and non-deterministic *logspace* Turing machines (the complexity classes L and NL) based on k-Lin. Our ABE scheme has compact secret keys of size linear in the description size of the Turing machine M. The ciphertext size grows linearly in the input length, but also linearly in the time complexity, and exponentially in the space complexity. Irrespective of compactness, we stress that our scheme is the first that supports large classes of Turing machines based solely on standard assumptions. In comparison, previous ABE for general Turing machines all rely on strong primitives related to indistinguishability obfuscation.

Keywords: attribute-based encryption · adaptive security · compactness · arithmetic branching program · logspace Turing machine · inner-product functional encryption

Contents

1 Introduction **1**

2 Technical Overview **4**
- 2.1 1-ABE from Arithmetic Key Garbling and IPFE Schemes 5
- 2.2 Full-Fledged ABE via IPFE 7
- 2.3 1-ABE for Logspace Turing Machines 9

3 Preliminaries **15**
- 3.1 Notational Conventions 15
- 3.2 Bilinear Pairing and Matrix Diffie-Hellman Assumption 17
- 3.3 Attribute-Based Encryption 18
- 3.4 Function-Hiding Slotted Inner-Product Functional Encryption 19

4 Computation Models **20**
- 4.1 Arithmetic Branching Programs 21
- 4.2 Turing Machines 22

5 Arithmetic Key Garbling Scheme **23**
- 5.1 Security Notions of AKGS 24

6 ABE for ABPs **26**
- 6.1 AKGS for ABPs 26
- 6.2 1-Key 1-Ciphertext Secure Secret-Key ABE 29
- 6.3 KP-ABE for ABPs 36

7 ABE for Uniform Logspace Turing Machines **44**
- 7.1 AKGS for Turing Machines with Time/Space Bounds 44
- 7.2 1-ABE for L 48
- 7.3 KP-ABE for L 63
- 7.4 Extension to NL 72
- 7.5 ABE for DFA/NFA 76

References **78**

Appendices

A Construction of Function-Hiding Slotted IPFE **84**

B Key Delegation **88**
- B.1 1-ABE Supporting Conjunctions 89
- B.2 Perfectly Key-Homomorphic IPFE 91
- B.3 KP-ABE Supporting Delegation 93

C Proof of Theorem 20 **100**

1 Introduction

Attribute-based encryption (ABE) [GPSW06] is an advanced form of public-key encryption that enables fine-grained access control. The encryption algorithm using the master public key mpk can encrypt a message m with a descriptive attribute x,[1] producing a ciphertext $\mathsf{ct}_x(m)$. The key generation algorithm using the master secret key msk can produce a secret key sk_y associated with an access policy y. Decrypting $\mathsf{ct}_x(m)$ using sk_y reveals the message m if the attribute x satisfies the policy y; otherwise, no information about m is revealed. The security requirement of ABE stipulates resilience to collusion attacks — any group of users holding secret keys for different policies learn nothing about the plaintext as long as none of them is individually authorized to decrypt the ciphertext.

A primary goal of research on ABE is designing ABE schemes for expressive classes of policies, usually defined by computation models or complexity classes. A beautiful and fruitful line of works have constructed ABE for many different policy classes. For non-uniform computation, we have ABE for Boolean [GPSW06,KW19] or arithmetic formulae, branching/span programs [LOS+10, OT10,LW11,OT12,LW12,IW14,GV15,Att16,CGKW18], and circuits [GVW13,BGG+14,Att17]. For uniform computation, we have ABE for deterministic finite automata [Wat12,Att14,Att16,AC17, GWW19,AMY19b], non-deterministic finite automata [AMY19a], and even Turing machines [AM18, AS17]. These constructions, however, achieve different trade-offs between security, efficiency, and underlying computational assumptions. It is rare to have a construction that simultaneously achieves the following natural desirata on all fronts:

- *Security*: (full) adaptive security (as opposed to selective or semi-adaptive security);
- *Efficiency*: having compact secret key and ciphertext, whose sizes grow linearly with the description size of the policy and the length of the attribute, respectively;
- *Assumptions*: relying on standard and simple assumptions, such as LWE and k-Lin or SXDH in bilinear pairing groups (in particular, it is preferable to avoid the use of strong primitives such as indistinguishability obfuscation, and instance-dependent assumptions such as q-type assumptions, whose strength can be weakened by adversarially chosen parameters).

All previous constructions of ABE fail to achieve at least one of the desirable properties, except for the recent construction of ABE for *Boolean formulae* from the k-Lin assumption by Kowalczyk and Wee [KW19]. This raises the question:

Can we construct ABE schemes with all the desirable properties above
for more expressive classes of policies than Boolean formulae?

When it comes to uniform computation, the state of affairs is even less satisfactory. All constructions of ABE for general Turing machines are based on strong primitives such as indistinguishability obfuscation and multilinear map. Without these powerful tools, existing schemes can only handle the weak computation model of finite automata.

Can we construct ABE schemes based on standard assumptions
for more expressive uniform computations than finite automata?

[1] Some works call x a set of attributes, and each bit or component of x an attribute. We treat the attribute as a single vector.

Our Result. Via a unified framework, we construct *compact and adaptively secure* ABE schemes based on the k-Lin assumption in asymmetric prime-order bilinear pairing groups for the following classes of policies:

Arithmetic Branching Programs. ABPs capture many functions of interest, including arithmetic computations like sparse polynomials, mean, and variance, as well as combinatorial computations like string-matching, finite automata, and decision trees. It is also known that Boolean/arithmetic formulae and Boolean branching programs can all be converted into ABPs with polynomial blow-up in description size. Thus, ABPs can be viewed as a more powerful computational model than them.

Previous ABE schemes for ABPs only provide selective security [GVW13,IW14] or do not have compact ciphertexts [CGKW18].[2] In addition to achieving both adaptive security and compactness, our scheme is the first one that handles ABPs directly without converting it to circuits or arithmetic span programs, which leads to an efficiency improvement in the size of the secret keys from up to quadratic to linear in the size of the ABP.[3]

(Non-)Deterministic Logspace Turing Machines (L and NL). Here, a secret key is associated with a Turing machine M, and the attribute in a ciphertext specifies an input $\mathbf{x}$, a polynomial time bound T, and a logarithmic space bound S. Decryption succeeds if and only if M accepts $\mathbf{x}$ within time T and space S. Our scheme is *unbounded* in the sense that the public parameters do not restrict the sizes of the Turing machine M and input $\mathbf{x}$, nor the time/space bounds T, S. Furthermore, it enjoys the advantage of ABE for uniform computation that a secret key for M can decrypt ciphertexts with arbitrarily long inputs and arbitrary time/space bounds. This stands in contrast with ABE for non-uniform computation (like ABPs), where a program or circuit f takes inputs of a specific length n, and a secret key for f decrypts only ciphertext of length-n inputs. Achieving this feature is precisely the challenge in constructing ABE for uniform models of computation.

Our scheme is the first ABE for large classes of Turing machine computation, captured by the complexity classes L and NL, *without using the heavy machineries* of multilinear map, extractable witness encryption, or indistinguishability obfuscation as in previous works [GKP+13a,AS16,AM18, KNTY19]. In addition, our scheme is adaptively secure and *half-compact*. The secret keys are compact, of size $O(|M|)$ linear in the description size of M, while the ciphertext size depends linearly in $|\mathbf{x}|TS2^S$ (both ignoring fixed polynomial factors in the security parameter). Note that the same secret key decrypt cipheretxts with different parameters $|x|, T, S$.

Removing the dependency on 2^S or T is an interesting open problem that requires technical breakthrough. In particular, removing the dependency on 2^S would give an ABE for polynomial-time Turing machine computation from pairing, a long sought-after goal that has remained elusive for more than a decade. Removing the dependency of encryption time on T even only in the 1-key 1-ciphertext setting implies a succinct message-hiding encoding [KLW15],[4] which is only known from strong primitives like indistinguishability obfuscation or functional encryption [KLW15,CHJV15,

[2] More precisely, they construct ABE for read-once branching programs. For general branching programs, one can duplicate each component in the attribute for the number of times it is accessed [KLMM19]. As such, the ciphertext size grows linearly with the size of the branching program.

[3] An ABP is specified by a directed graph, with edges weighted by affine functions of the input. The size of an ABP is measured by the number of vertices (instead of edges) in the graph.

[4] Message-hiding encodings [KLW15] are a weaker variant of randomized encodings that allow encoding a public computation f, x with a secret message m such that the encoding reveals m if and only if $f(x) = 1$. Such encodings are *succinct* if the time to encode is much smaller than the running time of the computation. A pair of ABE secret key for predicate f and ciphertext for attribute x and message m is a message-hiding encoding.

BGL+15,KNTY19]. Removing the dependency of ciphertext size on T might be an easier task, but would need new techniques different from ours.

Finite Automata. As a special case of ABE for L and NL, we obtain ABE for deterministic finite automata (DFA) and non-deterministic finite automata (NFA).[5] This simply follows from the fact that DFA and NFA can be represented as simple deterministic and non-deterministic Turing machines with space complexity 1 and time complexity N that always move the input tape pointer to the right and never use the work tape.

Previous schemes for DFA based on pairing either achieve only selective security [Wat12, GWW19,AMY19b] or rely on q-type assumptions [Att14,Att16,AC17]. The only direct construction of ABE for NFA [AMY19a] based on LWE, however, is symmetric-key and only selectively secure. We settle the open problem of constructing adaptively secure ABE for DFA from static assumptions [GWW19] and that of constructing ABE for NFA that is public-key, adaptively secure, or based on assumptions other than LWE [AMY19a].

New Techniques for Constructing Adaptively Secure ABE. Constructing adaptively secure ABE is a challenging task. Roughly speaking, previous constructions proceed in two steps. First, a secure core secret-key ABE component for a single ciphertext and a single secret key — termed 1-ABE — is designed. Then, dual system encryption framework, originally proposed in [Wat09] and refined in [Att14,Wee14,CGW15,Att16,AC17], provides guidance on how to lift 1-ABE to the public-key and multi-secret-key setting. The main technical challenge lies in the first step: Adaptively secure schemes prior to that of Kowalczyk and Wee [KW19] either impose a read-once restriction on the attribute[6] [LOS+10,Wee14] or rely on q-type assumptions [LW12,BSW07,Att14,AC17]. Kowalczyk and Wee [KW19] elegantly applied the "partial selectivization" framework [ACC+16,JKK+17] for achieving adaptive security in general to constructing 1-ABE. In particular, they used a variant of the secret-sharing scheme for Boolean formulae in [JKK+17] whose selective simulation security can be proven via a sequence of hybrids, each only requiring *partial* information of the input to be chosen selectively. Then, to show adaptive security, the reduction can *guess* this partial information while incurring only a polynomial security loss.

However, secret-sharing schemes as needed in [KW19] are only known for Boolean formulae. When dealing with computation over arithmetic domains of potentially exponential size, we have the additional challenge that it is hard to guess even a single component of the input, except with exponentially small probability, rendering the partial selectivization framework ineffective. When dealing with uniform computation, we further encounter the challenge that neither the secret key nor the ciphertext is as large as the secret-sharing, making it impossible to directly use information-theoretically secure secret-sharing schemes. We develop new techniques to overcome these challenges.

1. First, we present a generic framework for constructing adaptively secure 1-ABE from i) an information theoretic primitive called *arithmetic key garbling*, and ii) a computational primitive called *function-hiding inner-product functional encryption* (IPFE) [BJK15,TAO16,Lin17, KLM+18]. Our arithmetic key garbling schemes are partial garbling schemes [IW14] with special structures, which act as the counterpart of secret-sharing schemes for arithmetic computation. Our framework is modular: It decomposes the task of constructing 1-ABE to

[5] DFA and NFA both characterize regular languages, yet a DFA recognizing a language could have exponentially more states than an NFA recognizing the same language. In this work, by ABE for DFA/NFA, we mean ABE schemes that run in time polynomial in the description size of the finite automata.

[6] As mentioned in Footnote 2, read-once restriction can be circumvented by duplicating attribute components at the cost of losing ciphertext compactness.

first designing an arithmetic key garbling scheme for the computation class of interest, and *second* applying a generic transformation depending solely on structural properties of the garbling and agnostic of the underlying computation. In particular, the security proof of the transformation does not attempt to trace the computation, unlike [KW19,GWW19].

2. Second, we formulate structural properties of arithmetic key garbling schemes — called piecewise security — sufficient for achieving adaptive security. The properties are natural and satisfied by the garbling scheme for ABPs in [IW14]. For logspace Turing machine computation, we present a simple arithmetic key garbling scheme for L and NL, inspired by the garbling schemes in [AIK11,BGG+14].

3. Third, we present a new method of lifting 1-ABE to full-fledged ABE using function-hiding IPFE. Our method can be cast into the dual system encryption framework, but is natural on its own, without seeing through the lens of dual system encryption. One feature of IPFE is that it provides a conceptually simple abstraction which allows moving information between ABE keys and ciphertexts easily, and hides away lower-level details on how to guarantee security. This feature makes it a convenient tool in many other parts of the security proof as well.

4. Lastly, to overcome the unique challenge related to ABE for uniform computation, we further enhance our generic method to be able to use partial garbling generated with *pseudorandomness* so that the total size of the secret keys and ciphertexts can be smaller than the garbling.

Organization. In Section 2, we give an overview of our framework for constructing compact adaptively secure ABE schemes for ABPs, logspace Turing machines and finite automata, using as tools IPFE and arithmetic key garbling schemes (AKGS, a refinement of partial garbling schemes). After introducing basic notations and definitions in Section 3, we define AKGS and its security in Section 5. In Section 6, we construct the AKGS for ABPs, show how to construct 1-ABE (the core component of our ABE schemes) from an AKGS and how to lift a 1-ABE to a full-fledged ABE. In Section 7, we construct ABE for L, NL, DFA and NFA. In Appendix A, we show how to construct slotted IPFE from bilinear groups assuming k-Lin. Lastly, in Appendix B we show how to support key delegation.

2 Technical Overview

We now give an overview of our technique, starting with introducing the two key tools arithmetic key garbling schemes and IPFE. Below, by bilinear pairing groups, we mean asymmetric prime-order bilinear pairing groups, denoted as $(G_1, G_2, G_\mathrm{T}, g_1, g_2, e)$ and implicitly, $g_\mathrm{T} = e(g_1, g_2)$. We use $[\![a]\!]_b$ to represent the encoding g_b^a of a in group G_b.

Arithmetic Key Garbling Scheme. We use a refinement of the notion of partial garbling schemes [IW14] (which in turn is based on the notion of garbling and randomized encoding [Yao86, IK02,AIK04]). An *arithmetic key garbling scheme* (AKGS) is an information-theoretic partial garbling scheme for computing $\alpha f(\mathbf{x}) + \beta$ that hides the *secrets* $\alpha, \beta \in \mathbb{Z}_p$, but not $f, \mathbf{x}$:

- A garbling procedure $(\mathbf{L}_1, \ldots, \mathbf{L}_m) \leftarrow \mathsf{Garble}(f, \alpha, \beta; \mathbf{r})$ turns f and two secrets α, β (using randomness $\mathbf{r}$) into m affine *label functions* $L_1, \ldots, L_m$, described by their coefficient vectors

$\mathbf{L}_1, \ldots, \mathbf{L}_m$ over $\mathbb{Z}_p$. The label functions specify how to encode an input $\mathbf{x}$ to produce the *labels* for computing $f(\mathbf{x})$ with secrets α, β:

$$\widehat{f(\mathbf{x})}_{\alpha,\beta} = (\ell_1, \ldots, \ell_m), \text{ where } \ell_j = L_j(\mathbf{x}) = \langle \mathbf{L}_j, (1, \mathbf{x}) \rangle \text{ over } \mathbb{Z}_p. \tag{1}$$

- A *linear* evaluation procedure $\gamma \leftarrow \mathsf{Eval}(f, \mathbf{x}, \ell_1, \ldots, \ell_m)$ recovers the sum $\gamma = \alpha f(\mathbf{x}) + \beta$ weighted by the function value $f(\mathbf{x})$.

AKGS is a *partial* garbling as it only hides information of the secrets α and β beyond the weighted sum $\alpha f(\mathbf{x}) + \beta$, and does not hide $(f, \mathbf{x})$, captured by a simulation procedure $(\ell'_1, \ldots, \ell'_m) \xleftarrow{\$} \mathsf{Sim}(f, \mathbf{x}, \alpha f(\mathbf{x}) + \beta)$ that produces the same distribution as the honest labels.

Ishai and Wee [IW14] proposed a partial garbling scheme for ABPs, which directly implies an AKGS for ABPs. It is also easy to observe that the (fully secure) garbling scheme for arithmetic formulae in [AIK11] can be weakened [BGG+14] to an AKGS. Later, we will introduce additional structural and security properties of AKGS needed for our 1-ABE construction. These properties are natural and satisfied by both schemes [IW14,AIK11].

Inner-Product Functional Encryption. A *function-hiding* (secret-key) inner-product functional encryption (IPFE)[7] enables generating many secret keys $\mathsf{isk}(\mathbf{v}_j)$ and ciphertexts $\mathsf{ict}(\mathbf{u}_i)$ associated with vectors $\mathbf{v}_j$ and $\mathbf{u}_i$ such that decryption yields all the inner products $\{\langle \mathbf{u}_i, \mathbf{v}_j \rangle\}_{i,j} \pmod p$ and nothing else. In this work, we need an *adaptively secure* IPFE, whose security holds even against adversaries choosing all the vectors adaptively. Such an IPFE scheme can be constructed based on the k-Lin assumption in bilinear pairing groups [Wee17,LV16]. The known scheme also has nice structural properties that will be instrumental to our construction of ABE:

- $\mathsf{isk}(\mathbf{v}) \xleftarrow{\$} \mathsf{IPFE.KeyGen}(\mathsf{msk}, [\![\mathbf{v}]\!]_2)$ operates *linearly* on $\mathbf{v}$ (in the exponent of G_2) and the size of the secret key $\mathsf{isk}(\mathbf{v})$ grows linearly with $|\mathbf{v}|$.
- $\mathsf{ict}(\mathbf{u}) \xleftarrow{\$} \mathsf{IPFE.Enc}(\mathsf{msk}, [\![\mathbf{u}]\!]_1)$ also operates *linearly* on $\mathbf{u}$ (in the exponent of G_1) and the size of the ciphertext $\mathsf{ict}(\mathbf{u})$ grows linearly with $|\mathbf{u}|$.
- $\mathsf{IPFE.Dec}(\mathsf{sk}(\mathbf{v}), \mathsf{ct}(\mathbf{u}))$ simply invokes pairing to compute the inner product $[\![\langle \mathbf{u}, \mathbf{v} \rangle]\!]_T$ in the exponent of the target group.

2.1 1-ABE from Arithmetic Key Garbling and IPFE Schemes

1-ABE is the technical heart of our ABE construction. It works in the setting where a single ciphertext $\mathsf{ct}(\mathbf{x})$ for an input vector $\mathbf{x}$ and a single secret key $\mathsf{sk}(f, \mu)$ for a policy $y = f_{\neq 0}$ and a secret μ are published. Decryption reveals μ if $f(\mathbf{x}) \neq 0$; otherwise, μ is hidden.[8]

1-ABE. To hide μ conditioned on $f(\mathbf{x}) = 0$, our key idea is using IPFE to compute an AKGS garbling $\widehat{f(\mathbf{x})}_{\mu,0}$ of $f(\mathbf{x})$ with secrets $\alpha = \mu$ and $\beta = 0$. The security of AKGS guarantees that only $\mu f(\mathbf{x})$ is revealed, which information theoretically hides μ when $f(\mathbf{x}) = 0$.

The reason that it is possible to use IPFE to compute the garbling is attributed to the *affine input-encoding* property of AKGS — the labels $\ell_1, \ldots, \ell_m$ are the output of affine functions $L_1, \ldots, L_j$ of $\mathbf{x}$ as described in Equation (1). Since f, α, β are known at key generation time,

[7] Some works use "inner-product encryption" (IPE) to refer to IPFE [BJK15,DDM16,LV16,Lin17] and some others [KSW13,SSW09,OT09,OT10,OT12,CGW18] use it for inner-product *predicate* encryption.

[8] We can also handle policies of the form $f_{=0}$ so that μ is revealed if and only if $f(\mathbf{x}) = 0$. For simplicity, we focus on one case in this overview.

the ABE key can be a collection of IPFE secret keys, each encoding the coefficient vector $\mathbf{L}_j$ of one label function L_j. On the other hand, the ABE ciphertext can be an IPFE ciphertext encrypting $(1, \mathbf{x})$. When put together for decryption, they reveal exactly the labels $L_1(\mathbf{x}), \ldots, L_m(\mathbf{x})$, as described below on the left.

Honest Algorithms		Hybrid for Selective Security	
$\mathsf{ct}(\mathbf{x})$:	$\mathsf{ict}(\ (1,\mathbf{x}) \parallel \mathbf{0}\ )$	$\mathsf{ct}(\mathbf{x})$:	$\mathsf{ict}(\ (1,\mathbf{x}) \parallel 1 \parallel \mathbf{0}\ )$
$\mathsf{sk}(f,\mu)$:	$j \in [m]$: $\mathsf{isk}_j(\ \mathbf{L}_j \parallel \mathbf{0}\ )$	$\mathsf{sk}(f,\mu)$:	$j \in [m]$: $\mathsf{isk}_j(\ \mathbf{0} \parallel \ell_j \parallel \mathbf{0}\ )$

We note that the positions or slots at the right end of the vectors encoded in isk and ict are set to zero by the honest algorithms — $\mathbf{0}$ denotes a vector (of unspecified length) of zeros. These slots provide programming space in the security proof.

It is extremely simple to prove selective (or semi-adaptive) security, where the input $\mathbf{x}$ is chosen before seeing the sk. By the function-hiding property of IPFE, it is indistinguishable to switch the secret keys and the ciphertext to encode any vectors that preserve the inner products. This allows us to hardwire honestly generated labels $\widehat{f(\mathbf{x})}_{\mu,0} = \{\ell_j \leftarrow \langle \mathbf{L}_j, (1,\mathbf{x})\rangle\}_{j\in[m]}$ in the secret keys as described above on the right. The simulation security of AKGS then implies that only $\mu f(\mathbf{x})$ is revealed, i.e., nothing about μ is revealed.

Achieving Adaptive Security. When it comes to adaptive security, where the input $\mathbf{x}$ is chosen *after* seeing sk, we can no longer hardwire the honest labels $\widehat{f(\mathbf{x})}_{\mu,0}$ in the secret key, as $\mathbf{x}$ is undefined when sk is generated, and hence cannot invoke the simulation security of AKGS. Our second key idea is relying on a stronger security property of AKGS, named *piecewise security*, to hardwire simulated labels into the secret key in a piecemeal fashion.

Piecewise security of AKGS requires the following two properties: *i*) reverse sampleability — there is an efficient procedure $\mathsf{RevSamp}$ that can *perfectly* reversely sample the first label ℓ_1 given the output $\alpha f(\mathbf{x}) + \beta$ and all the other labels $\ell_2, \ldots, \ell_m$, and *ii*) marginal randomness — each ℓ_j of the following labels for $j > 1$ is uniformly distributed over $\mathbb{Z}_p$ even given all subsequent *label functions* $\mathbf{L}_{j+1}, \ldots, \mathbf{L}_m$. More formally,

$$\{\ell_1 \leftarrow \langle \mathbf{L}_1, (1,\mathbf{x})\rangle, \quad \mathbf{L}_2, \ldots, \mathbf{L}_m\} \equiv \{\ell_1' \xleftarrow{\$} \mathsf{RevSamp}(\cdots), \quad \mathbf{L}_2, \ldots, \mathbf{L}_m\}, \tag{2}$$

$$\{\ell_j \leftarrow \langle \mathbf{L}_j, (1,\mathbf{x})\rangle, \mathbf{L}_{j+1}, \ldots, \mathbf{L}_m\} \equiv \{\ell_j' \xleftarrow{\$} \mathbb{Z}_p \quad , \mathbf{L}_{j+1}, \ldots, \mathbf{L}_m\}. \tag{3}$$

In Equation (2), $\ell_1' \xleftarrow{\$} \mathsf{RevSamp}(f, \mathbf{x}, \alpha f(\mathbf{x}) + \beta, \ell_2, \ldots, \ell_m)$. These properties are natural and satisfied by existing AKGS for ABPs and arithmetic formulae [IW14,AIK11].

Adaptive Security via Piecewise Security. We are now ready to prove adaptive security of our 1-ABE. The proof strategy is to first hardwire ℓ_1 in the ciphertext and sample it reversely as $\ell_1 \xleftarrow{\$} \mathsf{RevSamp}(f, \mathbf{x}, 0, \ell_2, \ldots, \ell_m)$, where $\ell_j = \langle \mathbf{L}_j, (1,\mathbf{x})\rangle$ for $j > 1$ and $\mu f(\mathbf{x}) = 0$ by the constraint, as described in hybrid $k = 1$ below. The indistinguishability follows immediately from the function-hiding property of IPFE and the reverse sampleability of AKGS. Then, we gradually replace each remaining label function $\mathbf{L}_j$ for $j > 1$ with a randomly sampled label $\ell_j \xleftarrow{\$} \mathbb{Z}_p$ in the secret key, as described in hybrids $1 \le k \le m+1$. It is easy to observe that in the final hybrid $k = m+1$, where all labels $\ell_2, \ldots, \ell_m$ are random and ℓ_1 reversely sampled *without* μ, the value μ is information-theoretically hidden.

Hybrid $1 \le k \le m+1$	Hybrid $k:1$ or $k:2$
$\mathsf{sk}(f,\mu)$: $\mathsf{isk}_1(\ \mathbf{0}\ \|\ 1\ \|\ 0\ \|\ 0\ )$	
$1<j<k$: $\mathsf{isk}_j(\ \mathbf{0}\ \|\ 0\ \|\ \ell_j\ \|\ 0\ )$	
$\mathsf{isk}_k(\ \mathbf{L}_k\ \|\ 0\ \|\ 0\ \|\ 0\ )$	$\mathsf{isk}_k(\ \mathbf{0}\ \|\ 0\ \|\ 0\ \|\ 1\ )$
$j>k$: $\mathsf{isk}_j(\ \mathbf{L}_j\ \|\ 0\ \|\ 0\ \|\ 0\ )$	
$\mathsf{ct}(\mathbf{x})$: $\mathsf{ict}(\ (1,\mathbf{x})\ \|\ \ell_1\ \|\ 1\ \|\ 0\ )$	$\mathsf{ict}(\ (1,\mathbf{x})\ \|\ \ell_1\ \|\ 1\ \|\ \ell_k\ )$
$\ell_1 \xleftarrow{\$} \mathsf{RevSamp}(\cdots)$ and for $1<j<k$: $\ell_j \xleftarrow{\$} \mathbb{Z}_p$	$\ell_k \leftarrow \langle \mathbf{L}_k, (1,\mathbf{x})\rangle$ or $\ell_k \xleftarrow{\$} \mathbb{Z}_p$

To move from hybrid k to $k+1$, we want to switch the k^{th} IPFE secret key isk_k from encoding the label function $\mathbf{L}_k$ to a simulated label $\ell_k \xleftarrow{\$} \mathbb{Z}_p$. This is possible via two moves. First, by the function-hiding property of IPFE, we can hardwire the honest $\ell_k = \langle \mathbf{L}_k, (1,\mathbf{x})\rangle$ in the ciphertext as in hybrid $k:1$ (recall that at encryption time, $\mathbf{x}$ is known). Then, by the marginal randomness property of AKGS, we switch to sample ℓ_k as random in hybrid $k:2$. Lastly, hybrid $k:2$ is indistinguishable to hybrid $k+1$ again by the function-hiding property of IPFE.

1-ABE for ABPs. Plugging in the AKGS for ABPs by Ishai and Wee [IW14], we immediately obtain 1-ABE for ABPs based on k-Lin. The size of the garbling grows linearly with the number of vertices $|V|$ in the graph describing the ABP, i.e., $m = O(|V|)$. Combined with the fact that IPFE has linear-size secret keys and ciphertexts, our 1-ABE scheme for ABPs has secret keys of size $O(m|\mathbf{x}|) = O(|V||\mathbf{x}|)$ and ciphertexts of size $O(|\mathbf{x}|)$. This gives an efficiency improvement over previous 1-ABE or ABE schemes for ABPs [IW14,CGKW18], where the secret key size grows linearly with the number of edges $|E|$ in the ABP graph, due to that their schemes first convert ABPs into an arithmetic span program, which incurs the efficiency loss.

Discussion. Our method for constructing 1-ABE is generic and modular. In particular, it has the advantage that the proof of adaptive security is agnostic of the computation being performed and merely carries out the simulation of AKGS in a mechanic way. Indeed, if we plug in an AKGS for arithmetic formulae or any other classes of non-uniform computation, the proof remains the same. (Our 1-ABE for logspace Turing machines also follows the same blueprint, but needs additional ideas.) Furthermore, note that our method departs from the partial selectivization technique used in [KW19], which is not applicable to arithmetic computation as the security reduction cannot afford to guess even one component of the input $\mathbf{x}$. The problem is circumvented by using IPFE to hardwire the labels (i.e., ℓ_1, ℓ_k) that depend on $\mathbf{x}$ in the ciphertext.

2.2 Full-Fledged ABE via IPFE

From 1-ABE for the 1-key 1-ciphertext setting to full-fledged ABE, we need to support publishing multiple keys and make encryption public-key. It turns out that the security of our 1-ABE scheme directly extends to the many-key 1-ciphertext (still secret-key) setting via a simple hybrid argument. Consider the scenario where a ciphertext ct and multiple keys $\{\mathsf{sk}_q(f_q,\mu_q)\}_{q\in[Q]}$ that are unauthorized to decrypt the ciphertext are published. Combining the above security proof for 1-ABE with a hybrid argument, we can gradually switch each secret key sk_q from encoding honest label functions encapsulating μ_q to ones encapsulating an independent secret $\mu'_q \xleftarrow{\$} \mathbb{Z}_p$. Therefore, all the secrets $\{\mu_q\}_{q\in[Q]}$ are hidden.

The security of our 1-ABE breaks down once two ciphertexts are released. Consider publishing just a single secret key $\mathsf{sk}(f,\mu)$ and two ciphertexts $\mathsf{ct}_1(\mathbf{x}_1), \mathsf{ct}_2(\mathbf{x}_2)$. Since the label functions $L_1,\ldots,L_m$ are encoded in sk, decryption computes two AKGS garblings $\widehat{f(\mathbf{x}_1)}_{\mu,0}$ and $\widehat{f(\mathbf{x}_2)}_{\mu,0}$

generated using the *same* label functions. However, AKGS security does not apply when the label functions are reused.

What we wish is that IPFE decryption computes two garblings $\widehat{f(\mathbf{x}_1)}_{\mu,0} = (L_1(\mathbf{x}_1), \ldots, L_m(\mathbf{x}_1))$ and $\widehat{f(\mathbf{x}_2)}_{\mu,0} = (L'_1(\mathbf{x}_2), \ldots, L'_m(\mathbf{x}_2))$ using *independent* label functions. This can be achieved in a *computational* fashion relying on the fact that the IPFE scheme encodes the vectors and the decryption results in the exponent of bilinear pairing groups. Hence we can rely on computational assumptions such as SXDH or k-Lin, combined with the function-hiding property of IPFE to argue that the produced garblings are computationally independent. We modify the 1-ABE scheme as follows:

- If SXDH holds in the pairing groups, we encode in the ciphertext $(1, \mathbf{x})$ multiplied by a random scalar $s \xleftarrow{\$} \mathbb{Z}_p$. As such, decryption computes $(sL_1(\mathbf{x}), \ldots, sL_m(\mathbf{x}))$ *in the exponent*. We argue that the label functions $sL_1, \ldots, sL_m$ are computationally random *in the exponent*: By the function-hiding property of IPFE, it is indistinguishable to multiply s not with the ciphertext vector, but with the coefficient vectors in the secret key as depicted below on the right; by DDH (in G_2) and the linearity of Garble (i.e., the coefficients $\mathbf{L}_j$ depend linearly on the secrets α, β and the randomness $\mathbf{r}$ used by Garble), $s\mathbf{L}_j$ are the coefficients of pseudorandom label functions.

Algorithms based on SXDH		Hybrid	
			$\approx \mathbf{L}'_j$ (fresh)
$\mathsf{sk}(f, \mu)$:	$j \in [m]$: $\mathsf{isk}_j($ $\mathbf{L}_j$ $\| \mathbf{0})$	$\mathsf{isk}_j($ $\mathbf{L}_j$ $\|$ $s\mathbf{L}_j$ $\| \mathbf{0})$	
$\mathsf{ct}(\mathbf{x})$:	$\mathsf{ict}($ $s(1, \mathbf{x})$ $\| \mathbf{0})$	$\mathsf{ict}($ $\mathbf{0}$ $\| (1, \mathbf{x})$ $\| \mathbf{0})$	

- If k-Lin holds in the pairing groups, we encode in the secret key k independent copies of label functions $L_1^t, \ldots, L_m^t$ for $t \in [k]$, and in the ciphertexts k copies of $(1, \mathbf{x})$ multiplied with independent random scalars $\mathbf{s}[t]$ for $t \in [k]$. This way, decryption computes a random linear combination of the garblings $(\sum_{t\in[k]} \mathbf{s}[t]L_1^t(\mathbf{x}), \ldots, \sum_{t\in[k]} \mathbf{s}[t]L_m^t(\mathbf{x}))$ in the exponent, which via a similar hybrid as above corresponds to pseudorandom label functions in the exponent.

Algorithms based on k-Lin

$$\begin{array}{rrr} \mathsf{sk}(f,\mu): & j \in [m]: \mathsf{isk}_j(& \mathbf{L}_j^1 \| \cdots \| \mathbf{L}_j^k \| \mathbf{0}) \\ \mathsf{ct}(\mathbf{x}): & & \mathsf{ict}(\mathbf{s}[1](1,\mathbf{x}) \| \cdots \| \mathbf{s}[k](1,\mathbf{x}) \| \mathbf{0}) \end{array}$$

Hybrid

$$\begin{array}{rrr} & & \approx \mathbf{L}'_j \text{ (fresh)} \\ \mathsf{sk}(f,\mu): & j \in [m]: \mathsf{isk}_j(& \mathbf{L}_j^1 \| \cdots \| \mathbf{L}_j^k \| \sum_{t\in[k]} \mathbf{s}[t]\mathbf{L}_j^t \| \mathbf{0}) \\ \mathsf{ct}(\mathbf{x}): & & \mathsf{ict}(\mathbf{0} \| \cdots \| \mathbf{0} \| (1,\mathbf{x}) \| \mathbf{0}) \end{array}$$

The above modification yields a secret-key ABE secure in the many-ciphertext many-key setting. The final hurdle is how to make the scheme public-key, which we resolve using slotted IPFE.

Slotted IPFE. Proposed in [LV16], *slotted* IPFE is a hybrid between a secret-key function-hiding IPFE and a public-key IPFE. Here, a vector $\mathbf{u} \in \mathbb{Z}_p^n$ is divided into two parts $(\mathbf{u}_{\text{pub}}, \mathbf{u}_{\text{priv}})$ with $\mathbf{u}_{\text{pub}} \in \mathbb{Z}_p^{n_{\text{pub}}}$ in the public slot and $\mathbf{u}_{\text{priv}} \in \mathbb{Z}_p^{n_{\text{priv}}}$ in the private slot ($n_{\text{pub}} + n_{\text{priv}} = n$). Like a usual secret-key IPFE, the encryption algorithm $\mathsf{IPFE.Enc}$ using the master secret key msk can encrypt to both the public and private slots, i.e., encrypting any vector $\mathbf{u}$. In addition, there is an

$\mathsf{IPFE.SlotEnc}$ algorithm that uses the master public key mpk, but can only encrypt to the public slot, i.e., encrypting vectors such that $\mathbf{u}_{\text{priv}} = \mathbf{0}$. Since anyone can encrypt to the public slot, it is impossible to hide the public slot part $\mathbf{v}_{\text{pub}}$ of a secret-key vector $\mathbf{v}$. As a result, slotted IPFE guarantees function-hiding only w.r.t. the private slot, and the weaker indistinguishability security w.r.t. the public slot. Based on the construction of slotted IPFE in [Lin17], we obtain adaptively secure slotted IPFE based on k-Lin.

The aforementioned secret-key ABE scheme can be easily turned into a public-key one with slotted IPFE: The ABE encryption algorithm simply uses $\mathsf{IPFE.SlotEnc}$ and mpk to encrypt to the public slots. In the security proof, we move vectors encrypted in the public slot of the challenge ciphertext to the private slot, where function-hiding holds and the same security arguments outlined above can be carried out.

Discussion. Our method can be viewed as using IPFE to implement dual system encryption [Wat09]. We believe that IPFE provides a valuable abstraction, making it conceptually simpler to design strategies for moving information between the secret key and the ciphertext, as done in the proof of 1-ABE, and for generating independent randomness, as done in the proof of full ABE. The benefit of this abstraction is even more prominent when it comes to ABE for logspace Turing machines.

2.3 1-ABE for Logspace Turing Machines

We now present ideas for constructing 1-ABE for L, and then its extension to NL and how to handle DFA and NFA as special cases for better efficiency. Moving to full-fledged ABE follows the same ideas in the previous subsection, though slightly more complicated, which we omit in this overview.

1-ABE for L enables generating a single secret key $\mathsf{sk}(M, \mu)$ for a Turing machine M and secret μ, and a ciphertext $\mathsf{ct}(\mathbf{x}, T, S)$ specifying an input $\mathbf{x}$ of length N, a polynomial time bound $T = \text{poly}(N)$, and a logarithmic space bound $S = O(\log N)$ such that decryption reveals $\mu M|_{N,T,S}(\mathbf{x})$, where $M|_{N,T,S}(\mathbf{x})$ represents the computation of running $M(\mathbf{x})$ for T steps with a work tape of size S, which outputs 1 if and only if the computation lands in an accepting state after T steps and has *never* exceeded the space bound S. A key feature of ABE for uniform computation is that a secret key $\mathsf{sk}(M, \mu)$ can decrypt ciphertexts with inputs of unbounded lengths and unbounded time / (logarithmic) space bounds. (In contrast, for non-uniform computation, the secret key decides the input length and time/space bounds.) Our 1-ABE for L follows the same blueprint of combining AKGS with IPFE, but uses new ideas in order to implement the unique feature of ABE for uniform computation.

Notations for Turing Machines. We start with introducing notations for logspace Turing machines (TM) over the binary alphabet. A TM $M = (Q, q_{\text{acc}}, \delta)$ consists of Q states, with the initial state being 1 and an accepting state[9] $q_{\text{acc}} \in [Q]$, and a transition function δ. The computation of $M|_{N,T,S}(\mathbf{x})$ goes through a sequence of $T+1$ *configurations* $(\mathbf{x}, (i, j, \mathbf{W}, q))$, where $i \in [N]$ is the input tape pointer, $j \in [S]$ the work tape pointer, $\mathbf{W} \in \{0,1\}^S$ the content of the work tape, and $q \in [Q]$ the state. The initial *internal* configuration is thus $(i = 1, j = 1, \mathbf{W} = \mathbf{0}_S, q = 1)$, and the transition from one internal configuration $(i, j, \mathbf{W}, q)$ to the next $(i', j', \mathbf{W}', q')$ is governed by the transition function δ and the input $\mathbf{x}$. Namely, if $\delta(q, \mathbf{x}[i], \mathbf{W}[j]) = (q', w', \Delta i, \Delta j)$,

$$(i, j, \mathbf{W}, q) \to (i' = i + \Delta i,\ j' = j + \Delta j,\ \mathbf{W}' = \mathsf{overwrite}(\mathbf{W}, j, w'),\ q').$$

In other words, the transition function δ on input state q and bits $\mathbf{x}[i]$, $\mathbf{W}[j]$ on the input and work tape under scan, outputs the next state q', the new bit $w' \in \{0,1\}$ to be written to the work

[9] For simplicity, in this overview, we assume there is only one accepting state.

tape, and the directions $\Delta i, \Delta j \in \{0, \pm 1\}$ to move the input and work tape pointers. The next internal configuration is then derived by updating the current configuration accordingly, where $\mathbf{W}' = \mathsf{overwrite}(\mathbf{W}, j, w')$ is a vector obtained by overwriting the j^{th} cell of $\mathbf{W}$ with w' and keeping the other cells unchanged.

AKGS for Logspace Turing Machines. To obtain an AKGS for L, we represent the TM computation algebraically as a sequence of matrix multiplications over $\mathbb{Z}_p$, for which we design an AKGS. To do so, we represent each internal configuration as a basis vector $\mathbf{e}_{(i,j,\mathbf{W},q)}$ of dimension $NS2^S Q$ with a single 1 at position $(i, j, \mathbf{W}, q)$. We want to find a *transition matrix* $\mathbf{M}(\mathbf{x})$ (depending on δ and $\mathbf{x}$) such that moving to the next state $\mathbf{e}_{(i',j',\mathbf{W}',q')}$ simply involves (right) multiplying $\mathbf{M}(\mathbf{x})$, i.e., $\mathbf{e}^\mathsf{T}_{(i,j,\mathbf{W},q)}\mathbf{M}(\mathbf{x}) = \mathbf{e}^\mathsf{T}_{(i',j',\mathbf{W}',q')}$. It is easy to verify that the correct transition matrix is

$$\mathbf{M}(\mathbf{x})[(i,j,\mathbf{W},q),(i',j',\mathbf{W}',q')] = \mathsf{CanTransit}[(i,j,\mathbf{W}),(i',j',\mathbf{W}')] \times \mathbf{M}_{\mathbf{x}[i],\mathbf{W}[j],\mathbf{W}'[j],i'-i,j'-j}[q,q'], \tag{4}$$

$$\begin{aligned} \mathsf{CanTransit}[(i,j,\mathbf{W}),(i',j',\mathbf{W}')] = 1 \quad &\text{iff } \mathbf{W}'[\neq j] = \mathbf{W}[\neq j] \text{ and } i'-i, j'-j \in \{0,\pm 1\}, \\ \mathbf{M}_{x,w,w',\Delta i,\Delta j}[q,q'] = 1 \quad &\text{iff } \delta(q,x,w') = (q',w',\Delta i,\Delta j). \end{aligned} \tag{5}$$

Here, $\mathsf{CanTransit}[(i,j,\mathbf{W}),(i'j',\mathbf{W}')]$ indicates whether it is possible, irrespective of δ, to move from an internal configuration with $(i,j,\mathbf{W})$ to one with $(i',j',\mathbf{W}')$. If possible, then $\mathbf{M}_{\mathbf{x}[i],\mathbf{W}[j],\mathbf{W}'[j],\Delta i,\Delta j}[q,q']$ indicates whether δ permits moving from state q with current read bits $x = \mathbf{x}[i], w = \mathbf{W}[j]$ to state q' with overwriting bit $w' = \mathbf{W}'[j]$ and moving directions $\Delta i = i' - i, \Delta j = j' - j$. Armed with this, the TM computation can be done by right multiplying the matrix $\mathbf{M}(\mathbf{x})$ for T times with the initial configuration $\mathbf{e}^\mathsf{T}_{(1,1,\mathbf{0},1)}$, reaching the final configuration $\mathbf{e}^\mathsf{T}_{(i_T,j_T,\mathbf{W}_T,q_T)}$, and then testing whether $q_T = q_{\text{acc}}$. More precisely,

$$M|_{N,T,S}(\mathbf{x}) = \mathbf{e}^\mathsf{T}_{(1,1,\mathbf{0},1)}\big(\mathbf{M}(\mathbf{x})\big)^T\mathbf{t} \text{ for } \mathbf{t} = \mathbf{1}_{NS2^S} \otimes \mathbf{e}_{q_{\text{acc}}}.$$

To construct AKGS for L, it boils down to construct AKGS for matrix multiplication. Our construction is inspired by the randomized encoding for arithmetic NC^1 scheme of [AIK11] and the garbling mechanism for multiplication gates in [BGG+14]. Let us focus on garbling the computation $M|_{N,T,S}(\mathbf{x})$ with secrets $\alpha = \mu$ and $\beta = 0$ (the case needed in our 1-ABE). The garbling algorithm Garble produces the following affine label functions of $\mathbf{x}$:

$$\begin{aligned} \ell_{\text{init}} &= L_{\text{init}}(\mathbf{x}) = \mathbf{e}^\mathsf{T}_{(1,1,\mathbf{0},1)}\mathbf{r}_0, \\ t \in [T]\colon \quad \boldsymbol{\ell}_t &= (\ell_{t,z}) = \big(L_{t,z}(\mathbf{x})\big)_z = -\boxed{\mathbf{r}_{t-1}} + \mathbf{M}(\mathbf{x})\mathbf{r}_t, \\ \boldsymbol{\ell}_{T+1} &= (\ell_{T+1,z})_z = \big(L_{T+1,z}(\mathbf{x})\big)_z = -\boxed{\mathbf{r}_T} + \mu\mathbf{t}. \end{aligned}$$

Here, $z = (i,j,\mathbf{W},q)$ runs through all $NS2^SQ$ possible internal configurations and $\mathbf{r}_t \xleftarrow{\$} \mathbb{Z}_p^{[N]\times[S]\times\{0,1\}^S\times[Q]}$. The evaluation proceeds inductively, starting with $\ell_{\text{init}} = \mathbf{e}^\mathsf{T}_{(1,1,\mathbf{0},1)}\mathbf{r}_0$, going through $\mathbf{e}^\mathsf{T}_{(i_t,j_t,\mathbf{W}_t,q_t)}\mathbf{r}_t$ for every $t \in [T]$ using the identity below, and completing after T steps by combining $\mathbf{e}^\mathsf{T}_{(i_T,j_T,\mathbf{W}_T,q_T)}\mathbf{r}_T$ with $\boldsymbol{\ell}_{T+1}$ to get $\mathbf{e}^\mathsf{T}_{(i_T,j_T,\mathbf{W}_T,q_T)}\mu\mathbf{t} = \mu M|_{N,T,S}(\mathbf{x})$ as desired:

$$\mathbf{e}^\mathsf{T}_{(i_{t+1},j_{t+1},\mathbf{W}_{t+1},q_{t+1})}\mathbf{r}_{t+1} = \mathbf{e}^\mathsf{T}_{(i_t,j_t,\mathbf{W}_t,q_t)}\mathbf{r}_t + \mathbf{e}^\mathsf{T}_{(i_t,j_t,\mathbf{W}_t,q_t)}(\underbrace{-\mathbf{r}_t + \mathbf{M}(\mathbf{x})\mathbf{r}_{t+1}}_{\boldsymbol{\ell}_{t+1}}).$$

We now show that the above AKGS is *piecewise secure*. First, ℓ_{init} is reversely sampleable. Since Eval is linear in the labels and ℓ_{init} has coefficient 1, given all but the first label ℓ_{init}, one can reversely

compute ℓ_{init} – the value uniquely determined by the linear equation[10] imposed by the correctness of Eval. Second, the marginal randomness property holds because every label $\boldsymbol{\ell}_t$ is random due to the random additive term $\mathbf{r}_{t-1}$ that is not used in subsequent label functions $L_{t',z}$ for all $t' > t$ and z, nor in the non-constant terms of $L_{t,z}$'s — we call $\mathbf{r}_{t-1}$ the *randomizers* of $\boldsymbol{\ell}_t$ (highlighted in the box). Lastly, we observe that the size of the garbling is $(T+1)NS2^SQ+1$.

1-ABE for L. We now try to construct 1-ABE for L from AKGS for L, following the same blueprint of using IPFE. Yet, applying the exact same method for non-uniform computation fails for multiple reasons. In 1-ABE for non-uniform computation, the ciphertext ct contains a single IPFE ciphertext ict encoding $(1, \mathbf{x})$, and the secret key sk contains a set of IPFE secret keys isk_j encoding all the label functions. However, in the uniform setting, the secret key $\mathsf{sk}(M, \mu)$ depends only on the TM M and the secret μ, and is supposed to work with ciphertexts $\mathsf{ct}(\mathbf{x}, T, S)$ with unbounded $N = |\mathbf{x}|, T, S$. Therefore, at key generation time, the size of the AKGS garbling, $(T+1)NS2^SQ+1$, is *unknown*, let alone generating and encoding all the label functions. Moreover, we want our 1-ABE to be compact, with secret key size $|\mathsf{sk}| = O(Q)$ linear in the number Q of states and ciphertext size $|\mathsf{ct}| = O(TNS2^S)$ (ignoring polynomial factors in the security parameter). The total size of secret key and ciphertext is much smaller than the total number of label functions, i.e., $|\mathsf{sk}| + |\mathsf{ct}| \ll (T+1)NS2^SQ+1$.

To overcome these challenges, our idea is that instead of encoding the label functions in the secret key or the ciphertext (for which there is not enough space), we let the secret key and the ciphertext jointly generate the label functions. For this idea to work, the label functions cannot be generated with true randomness which cannot be "compressed", and must use pseudorandomness instead. More specifically, our 1-ABE secret key $\mathsf{sk}(M, \mu)$ contains $\sim Q$ IPFE secret keys $\{\mathsf{isk}(\mathbf{v}_j)\}_j$, while the ciphertext $\mathsf{ct}(\mathbf{x}, T, S)$ contains $\sim TNS2^S$ IPFE ciphertexts $\{\mathsf{ict}(\mathbf{u}_i)\}_i$, such that decryption computes in the exponent $\sim TNS2^SQ$ cross inner products $\langle \mathbf{u}_i, \mathbf{v}_j \rangle$ that correspond to a garbling of $M|_{N,T,S}(\mathbf{x})$ with secret μ. To achieve this, we rely crucially on the special *block structure* of the transition matrix $\mathbf{M}$ (which in turn stems from the structure of TM computation, where the same transition function is applied in every step). Furthermore, as discussed above, we replace every truly random value $\mathbf{r}_t[i, j, \mathbf{W}, q]$ with a product $\mathbf{r}_{\mathrm{x}}[t, i, j, \mathbf{W}]\mathbf{r}_{\mathrm{f}}[q]$, which can be shown pseudorandom in the exponent based on SXDH.[11]

Block Structure of the Transition Matrix. Let us examine the transition matrix again (cf. Equations (4) and (5)):

$$\mathbf{M}(\mathbf{x})[(i, j, \mathbf{W}, q), (i', j', \mathbf{W}', q')] = \mathsf{CanTransit}[(i, j, \mathbf{W}), (i', j', \mathbf{W}')] \times \mathbf{M}_{\mathbf{x}[i], \mathbf{W}[j], \mathbf{W}'[j], i'-i, j'-j}[q, q'].$$

We see that that every block $\mathbf{M}(\mathbf{x})[(i, j, \mathbf{W}, \textvisiblespace), (i', j', \mathbf{W}', \textvisiblespace)]$ either is the $Q \times Q$ zero matrix or belongs to a small set $\mathcal{T}$ of a constant number of *transition blocks*:

$$\mathcal{T} = \left\{ \mathbf{M}_{x,w,w',\Delta i,\Delta j} \,\middle|\, x, w, w' \in \{0,1\},\ \Delta i, \Delta j \in \{0, \pm 1\} \right\}.$$

Moreover, in the $\mathfrak{i} = (i, j, \mathbf{W})^{\text{th}}$ "block row", $\mathbf{M}(\mathbf{x})[(\mathfrak{i}, \textvisiblespace), (\textvisiblespace, \textvisiblespace, \textvisiblespace, \textvisiblespace)]$, each transition block $\mathbf{M}_{x,w,w',\Delta i,\Delta j}$ either does not appear at all if $x \neq \mathbf{x}[i]$ or $w' \neq \mathbf{W}[j]$, or appears once as the block $\mathbf{M}(\mathbf{x})[(\mathfrak{i}, \textvisiblespace), (\mathfrak{i}', \textvisiblespace)]$, where $\mathfrak{i}'$ is the triplet obtained by updating $\mathfrak{i}$ appropriately according to $(w', \Delta i, \Delta j)$:

$$\mathfrak{i}' \stackrel{\text{def}}{=\!=} \mathfrak{i} \boxplus (w', \Delta i, \Delta j) = (i + \Delta i,\ j + \Delta j,\ \mathbf{W}' = \mathsf{overwrite}(\mathbf{W}, j, w')),$$
$$\mathbf{M}(\mathbf{x})[(\mathfrak{i}, \textvisiblespace), (\mathfrak{i}', \textvisiblespace)] = \mathbf{M}_{\mathbf{x}[i], \mathbf{W}[j], w', \Delta i, \Delta j}.$$

[10] This means $\mathsf{RevSamp}$ is deterministic, and we can reversely compute ℓ_{init} *in the exponent* and when the randomness is not uniform, which is important for our construction.

[11] Our scheme readily extends to be based on k-Lin. However, that makes the scheme more complex to present. We choose to present this scheme using SXDH in this paper.

Thus we can "decompose" every label $\boldsymbol{\ell}_t[\mathsf{i}, q]$ as an inner product $\langle \mathbf{u}_{t,\mathsf{i}}, \mathbf{v}_q \rangle$ as

$$\begin{aligned}
\boldsymbol{\ell}_t[\mathsf{i}, q] &= -\mathbf{r}_{t-1}[\mathsf{i}, q] + \mathbf{M}(\mathbf{x})[(\mathsf{i}, q)(\sqcup, \sqcup, \sqcup, \sqcup)]\mathbf{r}_t \\
&= -\mathbf{r}_{t-1}[\mathsf{i}, q] + \sum_{w', \Delta i, \Delta j} \big(\mathbf{M}_{\mathbf{x}[i], \mathbf{W}[j], w', \Delta i, \Delta j} \mathbf{r}_t[\mathsf{i}', \sqcup]\big)[q] \qquad \big(\mathsf{i}' = \mathsf{i} \boxplus (w', \Delta i, \Delta j)\big) \\
&= -\mathbf{r}_{\mathrm{x}}[t-1, \mathsf{i}]\mathbf{r}_{\mathrm{f}}[q] + \sum_{w', \Delta i, \Delta j} \mathbf{r}_{\mathrm{x}}[t, \mathsf{i}'] \big(\mathbf{M}_{\mathbf{x}[i], \mathbf{W}[j], w', \Delta i, \Delta j} \mathbf{r}_{\mathrm{f}}\big)[q] \\
&= \langle \mathbf{u}_{t,\mathsf{i}}, \mathbf{v}_q \rangle, \qquad \big(\mathbf{r}_{t''}[\mathsf{i}'', q''] = \mathbf{r}_{\mathrm{x}}[t'', \mathsf{i}'']\mathbf{r}_{\mathrm{f}}[q'']\big)
\end{aligned}$$

where vectors $\mathbf{u}_{t,\mathsf{i}}$ and $\mathbf{v}_q$ are as follows, with $\mathbb{1}\{\cdots\}$ indicating if the conditions (its argument) are true:

$$\begin{aligned}
\mathbf{u}_{t,\mathsf{i}} &= (\ \mathbf{r}_{\mathrm{x}}[t-1, \mathsf{i}] \parallel \cdots \parallel \mathbf{r}_{\mathrm{x}}[t, \mathsf{i}'] \cdot \mathbb{1}\{x = \mathbf{x}[i], w = \mathbf{W}[j]\} \parallel \cdots \parallel \mathbf{0}), \\
\mathbf{v}_q &= (\quad -\mathbf{r}_{\mathrm{f}}[q] \quad \parallel \cdots \parallel \quad (\mathbf{M}_{x,w,w',\Delta i,\Delta j}\mathbf{r}_{\mathrm{f}})[q] \quad \parallel \cdots \parallel \mathbf{0}).
\end{aligned}$$

Similarly, we can "decompose" $\ell_{\mathrm{init}} = \mathbf{e}_{1,1,\mathbf{0},1}^{\mathsf{T}} \mathbf{r}_0$ as $\langle \mathbf{r}_{\mathrm{x}}[0,1,1,\mathbf{0}], \mathbf{r}_{\mathrm{f}}[1] \rangle$. (For simplicity in the discussion below, we omit details on how to handle $\boldsymbol{\ell}_{T+1}$.) Given such decomposition, our semi-compact 1-ABE scheme follows immediately by using IPFE to compute the garbling:

Honest Algorithms

$$\begin{aligned}
\mathsf{sk}(M, \mu)\text{: } \mathsf{isk}_{\mathrm{init}}(\quad \mathbf{r}_{\mathrm{f}}[1] \quad \parallel \mathbf{0}\ ), \quad \forall q\text{: } \quad \mathsf{isk}_q(\ \mathbf{u}_{t,\mathsf{i}} \parallel \mathbf{0}\ ) \\
\mathsf{ct}(\mathbf{x}, T, S)\text{: } \mathsf{ict}_{\mathrm{init}}(\ \mathbf{r}_{\mathrm{x}}[0,1,1,\mathbf{0}] \parallel \mathbf{0}\ ), \quad \forall t, \mathsf{i}\text{: } \mathsf{ict}_{t,\mathsf{i}}(\ \mathbf{v}_q \parallel \mathbf{0}\ )
\end{aligned}$$

Decrypting the pair $\mathsf{isk}_{\mathrm{init}}, \mathsf{ict}_{\mathrm{init}}$ (generated using one master secret key) gives exactly the first label ℓ_{init}, while decrypting $\mathsf{isk}_q, \mathsf{ict}_{t,\mathsf{i}}$ (generated using another master secret key) gives the label $\boldsymbol{\ell}_t[\mathsf{i}, q]$ in the exponent, generated using pseudorandomness $\mathbf{r}_t[\mathsf{i}, q] = \mathbf{r}_{\mathrm{x}}[t, \mathsf{i}]\mathbf{r}_{\mathrm{f}}[q]$. Note that the honest algorithms encode $\mathbf{r}_{\mathrm{f}}[q]$ (in $\mathbf{v}_q$) and $\mathbf{r}_{\mathrm{x}}[t, \mathsf{i}]$ (in $\mathbf{u}_{t,\mathsf{i}}$) in IPFE secret keys and ciphertexts that use the *two* source groups G_1 and G_2 respectively. As such, we cannot directly use the SXDH assumption to argue the pseudorandomness of $\mathbf{r}_t[\mathsf{i}, q]$. In the security proof, we will use the function-hiding property of IPFE to move both $\mathbf{r}_{\mathrm{x}}[t, \mathsf{i}]$ and $\mathbf{r}_{\mathrm{f}}[q]$ into the same source group before invoking SXDH.

Adaptive Security. To show adaptive security, we follow the same blueprint of going through a sequence of hybrids, where we first hardcode ℓ_{init} and sample it reversely using $\mathsf{RevSamp}$, and next simulate the other labels $\boldsymbol{\ell}_t[\mathsf{i}, q]$ one by one. Hardwiring ℓ_{init} is easy by relying on the function-hiding property of IPFE. However, it is now more difficult to simulate $\boldsymbol{\ell}_t[\mathsf{i}, q]$ because *i)* before simulating $\boldsymbol{\ell}_t[\mathsf{i}, q]$, we need to switch its *randomizer* $\mathbf{r}_{t-1}[\mathsf{i}, q] = \mathbf{r}_{\mathrm{x}}[t-1, \mathsf{i}]\mathbf{r}_{\mathrm{f}}[q]$ to truly random $\mathbf{r}_{t-1}[\mathsf{i}, q] \xleftarrow{\$} \mathbb{Z}_p$, which enables us to simulate the label $\boldsymbol{\ell}_t[\mathsf{i}, q]$ as random; and *ii)* to keep simulation progressing, we need to switch the random $\boldsymbol{\ell}_t[\mathsf{i}, q]$ back to a pseudorandom value $\boldsymbol{\ell}_t[\mathsf{i}, q] = \mathbf{s}_{\mathrm{x}}[t, \mathsf{i}]\mathbf{s}_{\mathrm{f}}[q]$, as otherwise, there is not enough space to store all $\sim TNS2^S Q$ random labels $\boldsymbol{\ell}_t[\mathsf{i}, q]$.

We illustrate how to carry out above proof steps in the simpler case where the the adversary queries for the ciphertext first and the secret key second. The other case where the secret key is queried first is handled using similar ideas, but the technicality becomes much more delicate.

In hybrid (t, i), the first label ℓ_{init} is reversely sampled and hardcoded in the secret key $\mathsf{isk}_{\mathrm{init}}$, i.e., $\mathsf{ict}_{\mathrm{init}}$ encrypts $(1 \parallel 0)$ and $\mathsf{isk}_{\mathrm{init}}$ encrypts $(\ell_{\mathrm{init}} \parallel 0)$ with $\ell_{\mathrm{init}} \leftarrow \mathsf{RevSamp}(\cdots)$. All labels $\boldsymbol{\ell}_{t'}[\mathsf{i}', q]$ with $(t', \mathsf{i}') < (t, \mathsf{i})$ have been simulated as $\mathbf{s}_{\mathrm{x}}[t', \mathsf{i}']\mathbf{s}_{\mathrm{f}}[q]$ — observe that the ciphertext $\mathsf{ict}_{t',\mathsf{i}'}$ encodes only $\mathbf{s}_{\mathrm{f}}[t', \mathsf{i}']$ in the second slot, which is multiplied by $\mathbf{s}_{\mathrm{f}}[q]$ in the second slot of isk_q. On the other hand, all labels $\boldsymbol{\ell}_{t'}[\mathsf{i}', q]$ with $(t', \mathsf{i}') \geq (t, \mathsf{i})$ are generated honestly as the honest algorithms do.

$$
\begin{array}{rrl}
& \text{HYBRID } (t,\mathfrak{i}),\ \boxed{(t,\mathfrak{i}):1}\text{, AND } \boxed{(t,\mathfrak{i})+1} & \\
\mathsf{ct}(\mathbf{x},T,S)\text{:} & (t',\mathfrak{i}') < (t,\mathfrak{i})\text{:} & \mathsf{ict}_{t',\mathfrak{i}'}\big(\quad \mathbf{0} \quad \| \quad \mathbf{s}_{\mathrm{x}}[t',\mathfrak{i}'] \quad \| \quad 0 \quad \big) \\
& (t',\mathfrak{i}') = (t,\mathfrak{i})\text{:} & \mathsf{ict}_{t,\mathfrak{i}}\big(\mathbf{u}_{t,\mathfrak{i}} \ \boxed{\mathbf{0}}\ \boxed{\mathbf{0}} \| 0\ \boxed{0}\ \boxed{\mathbf{s}_{\mathrm{x}}[t,\mathfrak{i}]} \| 0 \quad \boxed{1} \quad \boxed{0} \big) \\
& (t',\mathfrak{i}') > (t,\mathfrak{i})\text{:} & \mathsf{ict}_{t',\mathfrak{i}'}\big(\quad \mathbf{u}_{t',\mathfrak{i}'} \quad \| \quad 0 \quad \| \quad 0 \quad \big) \\
\mathsf{sk}(M,\mu)\text{:} & q \in [Q]\text{:} & \mathsf{isk}_q\big(\quad \mathbf{v}_q \quad \| \quad \mathbf{s}_{\mathrm{f}}[q] \quad \| 0\ \boxed{\boldsymbol{\ell}_t[\mathfrak{i},q]}\ \boxed{0} \big)
\end{array}
$$

Moving from hybrid $(t,\mathfrak{i})$ to its successor $(t,\mathfrak{i})+1$, the only difference is that labels $\boldsymbol{\ell}_t[\mathfrak{i},q]$ are switched from being honestly generated $\langle \mathbf{u}_{t,\mathfrak{i}}, \mathbf{v}_q \rangle$ to pseudorandom $\mathbf{s}_{\mathrm{x}}[t,\mathfrak{i}]\mathbf{s}_{\mathrm{f}}[q]$, as depicted above with values in the solid line box (the rest of the hybrid is identical to hybrid $(t,\mathfrak{i})$). The transition can be done via an intermediate hybrid $(t,\mathfrak{i}):1$ with values in the dash line box. In this hybrid, all labels $\boldsymbol{\ell}_t[\mathfrak{i},q]$ produced as inner products of all $\mathbf{v}_q$'s and $\mathbf{u}_{t,\mathfrak{i}}$ are temporarily hardcoded in the secret keys isk_q, using the third slot (which is zeroed out in all the other $\mathbf{u}_{(t',\mathfrak{i}')\neq(t,\mathfrak{i})}$'s). Furthermore, $\mathbf{u}_{t,\mathfrak{i}}$ is removed from $\mathsf{ict}_{t,\mathfrak{i}}$. As such, the random scalar $\mathbf{r}_{\mathrm{x}}[t-1,\mathfrak{i}]$ (formerly embedded in $\mathbf{u}_{t,\mathfrak{i}}$) no longer appears in the exponent of group G_1, and $\ell_{\mathrm{init}} \leftarrow \mathsf{RevSamp}(\cdots)$ can be performed using $\mathbf{r}_{\mathrm{x}}[t-1,\mathfrak{i}]$, $\mathbf{r}_{\mathrm{f}}[q], \mathbf{r}_{t-1}[\mathfrak{i},q]$ in the exponent of G_2. Therefore, we can invoke the SXDH assumption in G_2 to switch the randomizers $\mathbf{r}_{t-1}[\mathfrak{i},q] = \mathbf{r}_{\mathrm{x}}[t-1,\mathfrak{i}]\mathbf{r}_{\mathrm{f}}[q]$ to be truly random, and hence so are the labels $\boldsymbol{\ell}_t[\mathfrak{i},q] \xleftarrow{\$} \mathbb{Z}_p$. By a similar argument, this intermediate hybrid $(t,\mathfrak{i}):1$ is also indistinguishable to $(t,\mathfrak{i})+1$, as the random $\boldsymbol{\ell}_t[\mathfrak{i},q]$ can be switched to $\mathbf{s}_{\mathrm{x}}[t,\mathfrak{i}]\mathbf{s}_{\mathrm{f}}[q]$ in hybrid $(t,\mathfrak{i})+1$, relying again on SXDH and the function-hiding property of IPFE. This concludes our argument of security in the simpler case where the ciphertext is queried first.

AKGS and 1-ABE for NL. Our construction of AKGS and 1-ABE essentially works for NL without modification, because the computation of a non-deterministic logspace Turing machine $M = ([Q], q_{\mathrm{acc}}, \delta)$ on an input $\mathbf{x}$ can also be represented as a sequence of matrix multiplications. We briefly describe how by pointing out the difference from L. The transition function δ of a non-deterministic TM dost not instruct a unique transition, but rather specifies a set of legitimate transitions. Following one internal configuration $(i,j,\mathbf{W},q)$, there are potentially many legitimate successors:

$$
(i,j,\mathbf{W},q) \to \big\{ (i'=i+\Delta i, j'=j+\Delta j, \mathbf{W}' = \mathsf{overwrite}(\mathbf{W},j,w'), q') \,\big|\, (q',w',\Delta i,\Delta j) \in \delta(q,\mathbf{x}[i],\mathbf{W}[j]) \big\}.
$$

The computation is accepting if and only if *there exists* a path with T legitimate transitions starting from $(1,1,\mathbf{0},1)$, through $(i_t,j_t,\mathbf{W}_t,q_t)$ for $t \in [T]$, and landing at $q_T = q_{\mathrm{acc}}$.

Naturally, we modify the transition matrix as below to reflect all legitimate transitions. The only difference is that each transition block determined by δ may map a state q to multiple states q', as highlighted in the solid line box:

$$
\begin{aligned}
\mathbf{M}(\mathbf{x})[(i,j,\mathbf{W},q),(i',j',\mathbf{W}',q')] &= \mathsf{CanTransit}[(i,j,\mathbf{W}),(i',j',\mathbf{W}')] \times \mathbf{M}_{\mathbf{x}[i],\mathbf{W}[j],\mathbf{W}'[j],i'-i,j'-j}[q,q'], \\
\mathbf{M}_{x,w,w',\Delta i,\Delta j}[q,q'] &= 1 \quad \text{iff } (q',w',\Delta i,\Delta j) \ \boxed{\in}\ \delta(q,x,w').
\end{aligned}
$$

Let us observe the effect of right multiplying $\mathbf{M}(\mathbf{x})$ to an $\mathbf{e}_{\mathfrak{i},q}$ indicating configuration $(\mathfrak{i},q)$: $\mathbf{e}_{\mathfrak{i},q}^{\mathsf{T}}\mathbf{M}(\mathbf{x})$ gives a vector $\mathbf{c}_1$ such that $\mathbf{c}_1[\mathfrak{i}',q'] = 1$ if and only if $(\mathfrak{i}',q')$ is a legitimate next configuration. Multiplying $\mathbf{M}(\mathbf{x})$ one more time, $\mathbf{e}_{\mathfrak{i},q}^{\mathsf{T}}\big(\mathbf{M}(\mathbf{x})\big)^2$ gives $\mathbf{c}_2$ where $\mathbf{c}_2[\mathfrak{i}',q']$ is the number of length-2 paths of legitimate transitions from $(\mathfrak{i},q)$ to $(\mathfrak{i}',q')$. Inductively, $\mathbf{e}_{\mathfrak{i},q}^{\mathsf{T}}\big(\mathbf{M}(\mathbf{x})\big)^t$ yields $\mathbf{c}_t$ that counts the number of length-t paths from $(\mathfrak{i},q)$ to any other internal configuration $(\mathfrak{i}',q')$. Therefore, we can *arithmetize* the computation of M on $\mathbf{x}$ as

$$
M|_{N,T,S}(\mathbf{x}) = \mathbf{e}_{(1,1,\mathbf{0},1)}^{\mathsf{T}}\big(\mathbf{M}(\mathbf{x})\big)^T\mathbf{t} \text{ for } \mathbf{t} = \mathbf{1}_{NS2^S} \otimes \mathbf{e}_{q_{\mathrm{acc}}}. \tag{6}
$$

Right multiplying $\mathbf{t}$ in the end sums up the number of paths to $(\mathsf{i}, q_{\mathrm{acc}})$ for all i in $\mathbf{c}_T$ (i.e., accepting paths).

If the computation is not accepting — there is no path to any $(\mathsf{i}, q_{\mathrm{acc}})$ — the final sum would be 0 as desired. If the computation is accepting — there is a path to some $(\mathsf{i}, q_{\mathrm{acc}})$ — then the sum should be non-zero (up to the following technicality). Now that we have represented NL computation as matrix multiplication, we immediately obtain AKGS and 1-ABE for NL using the same construction for L.

A Technicality in the Correctness for NL. The correctness of our scheme relies on the fact that when the computation is accepting, the matrix multiplication formula (Equation (6)) counts correctly the total number of length-T accepting paths. However, a subtle issue is that in our 1-ABE, the matrix multiplications are carried out over $\mathbb{Z}_p$, where p is the order of the bilinear pairing groups. This means if the total number of accepting paths happens to be a multiple of p, the sequence of matrix multiplications mod p carried out in 1-ABE would return 0, while the correct output should be non-zero. This technicality can be circumvented if p is entropic with $\omega(\log n)$ bits of entropy and the computation $(M, \mathbf{x}, T, S)$ is independent of p. In that case, the probability that the number of accepting paths is a multiple of p is negligible. We can achieve this by letting the setup algorithm of 1-ABE sample the bilinear pairing groups from a distribution with entropic order. Then, we have statistical correctness for computations $(M, \mathbf{x}, T, S)$ chosen statically ahead of time (independent of p). We believe such *static* correctness is sufficient for most applications where correctness is meant for non-adversarial behaviors. However, if the computation $(M, \mathbf{x}, T, S)$ is chosen *adaptively* to make the number of accepting paths a multiple of p, then an accepting computation will be mistakenly rejected. We stress that security is unaffected since if an adversary chooses M and $(\mathbf{x}, T, S)$ as such, it only learns less information.

The Special Cases of DFA and NFA. DFA and NFA are special cases of L and NL, respectively, as they can be represented as Turing machines with a work tape of size $S = 1$ that always runs in time $T = N$, and the transition function δ always moves the input tape pointer to the right. Therefore, the internal configuration of a finite automaton contains only the state q, and the transition matrix $\mathbf{M}(x)$ is determined by δ and the current input bit x under scan. Different from the case of L and NL, here the transition matrix no longer keeps track of the input tape pointer since its move is fixed — the t^{th} step uses the transition matrix $\mathbf{M}(\mathbf{x}[t])$ depending on $\mathbf{x}[t]$. Thus, the computation can be represented as follows:

$$M(\mathbf{x}) = \mathbf{e}_1^\top \prod_{t=1}^{N} \mathbf{M}(\mathbf{x}[t]) \cdot \mathbf{e}_{q_{\mathrm{acc}}} = \mathbf{e}_1^\top \prod_{t=1}^{N} \big(\mathbf{M}_0(1-\mathbf{x}[t]) + \mathbf{M}_1\mathbf{x}[t]\big) \cdot \mathbf{e}_{q_{\mathrm{acc}}},$$

where $\mathbf{M}_b[q, q'] = \mathbb{1}\{\delta(q, b) = q'\}$. Our construction of AKGS directly applies:

$$\begin{aligned} \ell_{\mathrm{init}} &= L_{\mathrm{init}}(\mathbf{x}) = \mathbf{e}_1^\top \mathbf{r}_0, \\ t \in [N]: \qquad \boldsymbol{\ell}_t &= \big(L_{t,q}(\mathbf{x})\big)_{q\in[Q]} = -\mathbf{r}_{t-1} + \boxed{\mathbf{M}(\mathbf{x}[t])}\,\mathbf{r}_t, \qquad \big(\mathbf{r}_{t-1}, \mathbf{r}_t \xleftarrow{\$} \mathbb{Z}_p^Q\big) \\ \boldsymbol{\ell}_{N+1} &= \big(L_{N+1,q}(\mathbf{x})\big)_{q\in[Q]} = -\mathbf{r}_N + \mu\mathbf{e}_{q_{\mathrm{acc}}}. \end{aligned}$$

When using pseudorandomness $\mathbf{r}_t[q] = \mathbf{r}_{\mathrm{f}}[q]\mathbf{r}_{\mathrm{x}}[t]$, the labels $\boldsymbol{\ell}_t[q]$ can be computed as $\langle \mathbf{u}_t, \mathbf{v}_q \rangle$ with

$$\begin{aligned} \mathbf{v}_q &= (\ \ -\mathbf{r}_{\mathrm{f}}[q] \ \ \|\ \ (\mathbf{M}_0\mathbf{r}_{\mathrm{f}})[q] \ \ \|\ (\mathbf{M}_1\mathbf{r}_{\mathrm{f}})[q] \,\|\, \mathbf{0}\ ), \\ \mathbf{u}_t &= (\ \mathbf{r}_{\mathrm{x}}[t-1] \,\|\, (1-\mathbf{x}[t])\mathbf{r}_{\mathrm{x}}[t] \,\|\ \ \mathbf{x}[t]\mathbf{r}_{\mathrm{x}}[t] \ \ \|\, \mathbf{0}\ ). \end{aligned}$$

Applying our 1-ABE construction with respect to such "decomposition" gives *compact* 1-ABE for DFA and NFA with secret keys of size $O(Q)$ and ciphertexts of size $O(N)$.

Discussion. Prior to our work, there have been constructions of ABE for DFA based on pairing [Wat12,Att14,Att16,AC17,GWW19,AMY19b] and ABE for NFA based on LWE [AMY19a]. However, no previous scheme achieves adaptive security unless based on q-type assumptions [Att14, Att16]. The work of [BL15] constructed ABE for DFA, and that of [AFS18] for random access machines, both based on LWE, but they only support inputs of bounded length, giving up the important advantage of uniform computation of handling unbounded-length inputs. There are also constructions of ABE (and even the stronger generalization, functional encryption) for Turing machines [GKP+13b,AS16,AM18,KNTY19] based on strong primitives such as multilinear map, extractable witness encryption, and indistinguishability obfuscation. However, these primitives are non-standard and currently not well-understood.

In terms of techniques, our work is most related to previous pairing-based ABE for DFA, in particular, the recent construction based on k-Lin [GWW19]. These ABE schemes for DFA use a linear secret-sharing scheme for DFA first proposed in [Wat12], and combining the secret key and ciphertext produces a secret-sharing in the exponent, which reveals the secret if and only if the DFA computation is accepting. Proving (even selective) security is complicated. Roughly speaking, the work of [GWW19] relies on an *entropy propagation technique* to trace the DFA computation and propagate a few random masks "down" the computation path, with which they can argue that secret information related to states that are *backward reachable* from the final accepting states is hidden. The technique is implemented using the "nested two-slot" dual system encryption [CGKW18, HKS15,LW10,LW11,OT12,Wat09] combined with a combinatorial mechanism for propagation.

Our AKGS is a generalization of Waters' secret-sharing scheme to L and NL, and the optimized version for DFA is identical to Waters' secret-sharing scheme. Furthermore, our 1-ABE scheme from AKGS and IPFE is more modular. In particular, our proof (similar to our 1-ABE for non-uniform computation) does not reason about or trace the computation, and simply relies on the structure of AKGS. Using IPFE enables us to design sophisticated sequences of hybrids without getting lost in the algebra, as IPFE helps separating the logic of changes in different hybrids from how to implement the changes. For instance, we can easily manage multiple slots in the vectors encoded in IPFE for holding temporary values and generating pseudorandomness.

3 Preliminaries

3.1 Notational Conventions

Let λ be the security parameter that runs through $\mathbb{N}$, the set of natural numbers. Except in the definitions, we suppress the security parameter for convenience. A function $\varepsilon : \mathbb{N} \to \mathbb{R}$ is negligible if $\varepsilon(\lambda) = O(\lambda^{-k})$ as $\lambda \to \infty$ for all $k \in \mathbb{N}$. Throughout the paper, let p be a prime number.

Efficient algorithms in the schemes are always probabilistic polynomial-time (PPT) Turing machines. We stress that the running time is polynomial in the total length of its input, not just the security parameter. Efficient adversaries are families of polynomial-sized probabilistic interactive circuits. Adversaries always and only receive the security parameter in unary as its input, so its size (thus running time) is polynomial in the security parameter.

Let Exp be an interactive experiment that interacts with an adversary, varies by the security parameter and has binary outcome (we refer to such object as "experiments" or "hybrids" thereafter) and let $\mathcal{A}$ be some adversary. We define "$\mathsf{Exp}^{\mathcal{A}}(1^\lambda) \to 1$" to be the event that the outcome of running Exp with $\mathcal{A}$ and security parameter λ is 1. Let $\mathsf{Exp}^0, \mathsf{Exp}^1$ be two experiments. The

distinguishing advantage of $\mathcal{A}$ against $\mathsf{Exp}^0, \mathsf{Exp}^1$ is defined as

$$\mathsf{Adv}^{\mathcal{A}}_{\mathsf{Exp}^0,\mathsf{Exp}^1}(\lambda) = \Pr[\mathsf{Exp}^{\mathcal{A},0}(1^\lambda) \to 1] - \Pr[\mathsf{Exp}^{\mathcal{A},1}(1^\lambda) \to 1].$$

We write $\mathsf{Exp}^0 \approx \mathsf{Exp}^1$ if they are *computationally indistinguishable* (or simply *indistinguishable*): For all efficient adversary $\mathcal{A}$, its distinguishing advantage against them is negligible. We write $\mathsf{Exp}^0 \approx_s \mathsf{Exp}^1$ if they are *statistically indistinguishable*: For all (even unbounded) adversary $\mathcal{A}$, its distinguishing advantage against them is negligible. We write $\mathsf{Exp}^0 \equiv \mathsf{Exp}^1$ if they are *identically distributed* (or simply *identical*): For all (even unbounded) adversary $\mathcal{A}$, its distinguishing advantage against them is 0.

The same notations can be used for sequences of distributions. Let $D^0 = \{D^0_\lambda\}_{\lambda\in\mathbb{N}}, D^1 = \{D^1_\lambda\}_{\lambda\in\mathbb{N}}$ be two sequences of distributions indexed by the security parameter. $\mathsf{Exp}^{\mathcal{A},b}(1^\lambda)$ works by sampling $x \xleftarrow{\$} D^b_\lambda$, running $\mathcal{A}(1^\lambda, x)$, and using the output bit of $\mathcal{A}$ as the outcome. We write $D_0 \approx D_1$ (resp. $D_0 \approx_s D_1$, $D_0 \equiv D_1$) if $\mathsf{Exp}^0 \approx \mathsf{Exp}^1$ (resp. $\mathsf{Exp}^0 \approx_s \mathsf{Exp}^1$, $\mathsf{Exp}^0 \equiv \mathsf{Exp}^1$).

Sets and Indexing. For $m, n \in \mathbb{N}$, denote by $[m..n]$ the set $\{m, \ldots, n\}$ (when $m > n$, this is the empty set) and $[n]$ the set $[1..n]$. Let $\mathbb{Z}_p$ be the finite field of integers modulo p. We use $\mathcal{I}, \mathfrak{s}$ to denote finite sets of indices from some index universe specified in context.

In this paper, objects might be indexed with many indices. To avoid confusion, we always write vectors and matrices in boldface. To index into a vector or a matrix, we write $\mathbf{v}[i]$ or $\mathbf{A}[i, j]$. In contrast, to index into some collection of objects that is not regarded as a vector or a matrix, we use subscripts or superscripts. As an example, the notation $\mathbf{v}_t$ represents a vector, not a component of some vector. If t runs through $[k]$, this means there are k vectors, $\mathbf{v}_1, \ldots, \mathbf{v}_k$. If the k objects are instead scalars, we will write $v_1, \ldots, v_k$.

For ease of exposition, we might use non-integer indices for vectors. Let S be any set, we write $S^{\mathcal{I}}$ for the set of vectors whose entries are in S and indexed by $\mathcal{I}$, i.e., $S^{\mathcal{I}} = \{(\mathbf{v}[i])_{i\in\mathcal{I}} \mid \mathbf{v}[i] \in S\}$. For example, let $\mathcal{I} = [n]$ and $S = \mathbb{Z}_p$, then $\mathbb{Z}_p^{[n]}$ is just $\mathbb{Z}_p^n$. Suppose $\mathfrak{s}_1, \mathfrak{s}_2$ are two index sets with $\mathfrak{s}_1 \subseteq \mathfrak{s}_2$. For any vector $\mathbf{v} \in \mathbb{Z}_p^{\mathfrak{s}_1}$, we write $\mathbf{u} = \mathbf{v}|^{\mathfrak{s}_2}$ for its zero-extension into $\mathbb{Z}_p^{\mathfrak{s}_2}$, i.e., $\mathbf{u} \in \mathbb{Z}_p^{\mathfrak{s}_2}$ and

$$\mathbf{u}[i] = \begin{cases} \mathbf{v}[i], & i \in \mathfrak{s}_1; \\ 0, & \text{otherwise.} \end{cases}$$

Conversely, for any vector $\mathbf{v} \in \mathbb{Z}_p^{\mathfrak{s}_2}$, we write $\mathbf{u} = \mathbf{v}|_{\mathfrak{s}_1}$ for its canonical projection onto $\mathbb{Z}_p^{\mathfrak{s}_1}$, i.e., $\mathbf{u} \in \mathbb{Z}_p^{\mathfrak{s}_1}$ and $\mathbf{u}[i] = \mathbf{v}[i]$ for $i \in \mathfrak{s}_1$. Lastly, let $\mathbf{u}, \mathbf{v} \in \mathbb{Z}_p^{\mathfrak{s}}$, denote by $\langle \mathbf{u}, \mathbf{v} \rangle$ their inner product, i.e., $\sum_{i\in\mathfrak{s}} \mathbf{u}[i]\mathbf{v}[i]$.

Similarly, non-integer indices can be used for matrices. Let S be a set and $\mathcal{I}_1, \mathcal{I}_2$ two index sets, we write $S^{\mathcal{I}_1\times\mathcal{I}_2}$ for the set of matrices whose entries are in S and indexed by $\mathcal{I}_1$ in the row and $\mathcal{I}_2$ in the column. Matrix multiplication and multiplication between matrices and vectors take their natural semantics so that the inner (common) index set is summed over. For example, a matrix $\mathbf{M} \in \mathbb{Z}_p^{\mathcal{I}_1\times\mathcal{I}_2}$ can multiply a vector in $\mathbb{Z}_p^{\mathcal{I}_2}$ to the left yielding a vector in $\mathbb{Z}_p^{\mathcal{I}_1}$ and any matrix in $\mathbb{Z}_p^{\mathcal{I}_2\times\mathcal{I}_3}$ to the left yielding a matrix in $\mathbb{Z}_p^{\mathcal{I}_1\times\mathcal{I}_3}$.

We use $\mathbf{0}_{\mathcal{I}}$ (resp. $\mathbf{0}_{\mathcal{I}_1\times\mathcal{I}_2}$) to represent the zero vector (resp. matrix) indexed by $\mathcal{I}$ (resp. $\mathcal{I}_1$ and $\mathcal{I}_2$), and $\mathbf{1}_{\mathcal{I}}$ for the vector whose entries are indexed by $\mathcal{I}$ and are all 1. The index sets $\mathcal{I}$ and $\mathcal{I}_1, \mathcal{I}_2$ can be suppressed if they are clear from the context.

To make transposition of vectors meaningful, we equate $S^{\mathcal{I}}$ with $S^{\mathcal{I}\times\{0\}}$ and S with $S^{\{0\}}$ (here $\{0\}$ can be any singleton set). For example, suppose $\mathbf{u} \in \mathbb{Z}_p^{\mathcal{I}_1}, \mathbf{v} \in \mathbb{Z}_p^{\mathcal{I}_2}$ and $\mathbf{M} \in \mathbb{Z}_p^{\mathcal{I}_1\times\mathcal{I}_2}$, then $\mathbf{u}, \mathbf{v}$ are also matrices of shape $\mathcal{I}_1 \times \{0\}$ and $\mathcal{I}_2 \times \{0\}$, respectively. The expression $\mathbf{u}^\intercal\mathbf{M}\mathbf{v}$ is a chained multiplication of matrices of shapes $\{0\} \times \mathcal{I}_1, \mathcal{I}_1 \times \mathcal{I}_2, \mathcal{I}_2 \times \{0\}$, so the value of the expression is a

matrix in $\mathbb{Z}_p^{\{0\}\times\{0\}}$, which is just a $\mathbb{Z}_p$ element, the same as in the case where $\mathbf{u}, \mathbf{v}, \mathbf{M}$ are vectors and matrices indexed by integers.

Tensor Product. Let $\mathbf{v}_1 \in \mathbb{Z}_p^{\mathfrak{s}_1}$ and $\mathbf{v}_2 \in \mathbb{Z}_p^{\mathfrak{s}_2}$ be two vectors, their tensor product $\mathbf{v} = \mathbf{v}_1 \otimes \mathbf{v}_2$ is a vector in $\mathbb{Z}_p^{\mathfrak{s}_1\times\mathfrak{s}_2}$ (not a matrix) with entries defined by $\mathbf{v}[(s_1, s_2)] = \mathbf{v}_1[s_1]\mathbf{v}_2[s_2]$. For two matrices $\mathbf{M}_1 \in \mathbb{Z}_p^{\mathcal{I}_1\times\mathcal{I}_1'}$ and $\mathbf{M}_2 \in \mathbb{Z}_p^{\mathcal{I}_2\times\mathcal{I}_2'}$, their tensor product $\mathbf{M} = \mathbf{M}_1 \otimes \mathbf{M}_2$ is a matrix in $\mathbb{Z}_p^{(\mathcal{I}_1\times\mathcal{I}_2)\times(\mathcal{I}_1'\times\mathcal{I}_2')}$ with entries defined by $\mathbf{M}[(i_1, i_2), (j_1, j_2)] = \mathbf{M}_1[i_1, j_1]\mathbf{M}_2[i_2, j_2]$.

Currying. Currying is the procedure of partially applying a function or specifying part of the indices of a vector/matrix, which yields another function with fewer arguments or another vector/matrix with fewer indices. We use the usual syntax for evaluating a function or indexing into a vector/matrix, except that unspecified variables are represented by "$\sqcup$". For example, let $\mathbf{M} \in \mathbb{Z}_p^{([A]\times[B])\times([C]\times[D])}$ and $a \in [A], d \in [D]$, then $\mathbf{M}[(a, \sqcup), (\sqcup, d)]$ is a matrix $\mathbf{N} \in \mathbb{Z}_p^{[B]\times[C]}$ such that $\mathbf{N}[b, c] = \mathbf{M}[(a, b), (c, d)]$ for all $b \in [B], c \in [C]$.

Coefficient Vector. An affine function $f : \mathbb{Z}_p^{\mathcal{I}} \to \mathbb{Z}_p$ is conveniently associated with its coefficient vector $\mathbf{f} \in \mathbb{Z}_p^{\mathfrak{s}}$ (written as the same letter in boldface) for $\mathfrak{s} = \{\mathsf{const}\} \cup \{\mathsf{coef}_i \mid i \in \mathcal{I}\}$ such that

$$f(\mathbf{x}) = \mathbf{f}[\mathsf{const}] + \sum_{i\in\mathcal{I}} \mathbf{f}[\mathsf{coef}_i]\mathbf{x}[i].$$

3.2 Bilinear Pairing and Matrix Diffie-Hellman Assumption

Throughout the paper, we use a sequence of bilinear pairing groups

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

where $G_{\lambda,1}, G_{\lambda,2}, G_{\lambda,\mathrm{T}}$ 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 $e_\lambda : G_{\lambda,1} \times G_{\lambda,2} \to G_{\lambda,\mathrm{T}}$ are

- *bilinear:* $e_\lambda(g_{\lambda,1}^a, g_{\lambda,2}^b) = \big(e_\lambda(g_{\lambda,1}, g_{\lambda,2})\big)^{ab}$ for all a, b; and
- *non-degenerate:* $e_\lambda(g_{\lambda,1}, g_{\lambda,2})$ generates $G_{\lambda,\mathrm{T}}$.

Implicitly, we set $g_{\lambda,\mathrm{T}} = e(g_{\lambda,1}, g_{\lambda,2})$. We require the group operations as well as the bilinear maps be efficiently computable.

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

Matrix Diffie-Hellman Assumption. In this work, we rely on the MDDH assumptions defined in [EHK+13].

Definition 1 ($\mathrm{MDDH}_{k,\ell}^q$ [EHK+13])**.** *Let $k \geq 1$ be an integer constant and $\ell = \ell(\lambda), q = q(\lambda)$ some polynomials. For a sequence of pairing groups $\mathcal{G}$ of order $p(\lambda)$, $\mathrm{MDDH}_{k,\ell}^q$ holds in G_i ($i = 1, 2, \mathrm{T}$) if*

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

When ℓ is omitted, it is implicit that $\ell(\lambda) = k+1$, and when q is omitted, it is implicit that $q(\lambda) = 1$. It is known that the k-Lin assumption implies MDDH_k, which will be the main assumption we base our schemes on. Moreover, MDDH_k implies $\mathrm{MDDH}^q_{k,\ell}$ for any ℓ, q:

Lemma 2 ([EHK+13])**.** *For any polynomial $\ell = \ell(\lambda)$ and $q = q(\lambda)$, the $\mathrm{MDDH}^q_{k,\ell}$ assumption in G_i ($i = 1, 2, \mathrm{T}$) is reducible to MDDH_k with a multiplicative security loss $\min\{\ell, q\}$ plus an additive security loss $O(1/p)$.*

We choose to present more complex schemes using the SXDH assumption (DDH assumption in both G_1, G_2).

Definition 3 (DDH)**.** *For a sequence of pairing groups $\mathcal{G}$ of order $p(\lambda)$,* DDH holds in G_i *($i = 1, 2, \mathrm{T}$) if $\{[\![a, b, ab]\!]_i\}_{\lambda\in\mathbb{N}} \approx \{[\![a, b, c]\!]_i\}_{\lambda\in\mathbb{N}}$ for $a, b, c \xleftarrow{\$} \mathbb{Z}_{p(\lambda)}$.*

Remarks. The above definition is DDH with respect to the fixed generator and MDDH_1 is DDH with respect to a random generator. The latter is implied by the former [EHK+13], and the recent work of [BMZ19] separates the two in the generic group model. Our schemes do not rely on the generator being fixed (they generalize to MDDH_k), yet fixing the generator helps focusing on the high-level ideas.

3.3 Attribute-Based Encryption

Definition 4. *Let $\mathcal{M} = \{M_\lambda\}_{\lambda\in\mathbb{N}}$ be a sequence of message sets. Let $\mathcal{P} = \{\mathcal{P}_\lambda\}_{\lambda\in\mathbb{N}}$ be a sequence of families of predicates, where $\mathcal{P}_\lambda = \{P : X_P \times Y_P \to \{0, 1\}\}$. An* attribute-based encryption (ABE) scheme *for message space $\mathcal{M}$ and predicate space $\mathcal{P}$ consists of 4 efficient algorithms:*

- $\mathsf{Setup}(1^\lambda, P \in \mathcal{P}_\lambda)$ *generates a pair of master public/secret key* $(\mathsf{mpk}, \mathsf{msk})$.
- $\mathsf{KeyGen}(1^\lambda, \mathsf{msk}, y \in Y_P)$ *generates a secret key* sk_y *associated with* y.
- $\mathsf{Enc}(1^\lambda, \mathsf{mpk}, x \in X_P, g \in M_\lambda)$ *generates a ciphertext* $\mathsf{ct}_{x,g}$ *for* g *associated with* x.
- $\mathsf{Dec}(1^\lambda, \mathsf{sk}, \mathsf{ct})$ *outputs either $\perp$ or a message in M_λ.*

Correctness *requires that for all $\lambda \in \mathbb{N}$, all $P \in \mathcal{P}_\lambda, g \in M_\lambda$, and all $y \in Y_P, x \in X_P$ such that $P(x, y) = 1$,*

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

The basic security requirement of an ABE scheme is IND-CPA security, which stipulates that no information about the message can be inferred as long as each individual secret key the adversary receives does not allow decryption.

Definition 5 (IND-CPA for ABE)**.** *Let* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *be an ABE for message space $\mathcal{M}$ and predicate space $\mathcal{P}$. The scheme is* IND-CPA secure *if $\mathsf{Exp}^0_{\mathrm{CPA}} \approx \mathsf{Exp}^1_{\mathrm{CPA}}$, where $\mathsf{Exp}^b_{\mathrm{CPA}}$ for $b \in \{0, 1\}$ is defined as follows:*

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

- **Query I.** Repeat the following for arbitrarily many rounds determined by $\mathcal{A}$: In each round, $\mathcal{A}$ submits $y_q \in Y_P$ for a secret key. Upon this query, run $\mathsf{sk}_q \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, \mathsf{msk}, y_q)$ and return sk_q to $\mathcal{A}$.

- **Challenge.** $\mathcal{A}$ submits a challenge attribute $x \in X_P$ and two messages $g_0, g_1 \in M_\lambda$. Run $\mathsf{ct} \xleftarrow{\$} \mathsf{Enc}(1^\lambda, \mathsf{mpk}, x, g_b)$ and return ct to $\mathcal{A}$.

- **Query II.** Same as Query I.

- **Guess.** $\mathcal{A}$ outputs a bit b'. Suppose $\mathcal{A}$ makes Q key queries in total. The outcome of the experiment is b' if $P(x, y_q) = 0$ for all $q \in [Q]$. Otherwise, the outcome is 0.

3.4 Function-Hiding Slotted Inner-Product Functional Encryption

In this paper, we rely on a hybrid variant between secret-key and public-key inner-product functional encryption (IPFE) schemes, called *slotted IPFE* [LV16,Lin17]. Here, the index set $\mathfrak{s}$ of vectors is partitioned into two subsets, $\mathfrak{s}_{\mathrm{pub}}$ and $\mathfrak{s}_{\mathrm{priv}}$, referred to as the public/private slot, respectively. Like in a secret-key IPFE, using the master *secret* key, one can generate ciphertexts and keys encoding any vectors. In addition, similar to a public-key IPFE, using the master *public* key, one can generate ciphertexts, however, only for vectors $\mathbf{u}$ with values set to zero in the private slot ($\mathbf{u}|_{\mathfrak{s}_{\mathrm{priv}}} = \mathbf{0}$). In other words, master public key allows for encrypting into the public slot only.

We also incorporate the property that the scheme encodes vectors and inner products in the exponent of pairing groups into the definition. As we shall see later, pairing-based slotted IPFE allows us to employ techniques akin to dual system encryption [Wat09,LW10,Att16].

Definition 6 (pairing-based slotted IPFE)**.** *Let $\mathcal{G}$ be a sequence of pairing groups of order $p(\lambda)$. A* slotted inner-product functional encryption (IPFE) scheme based on $\mathcal{G}$ *consists of 5 efficient algorithms:*

- $\mathsf{Setup}(1^\lambda, \mathfrak{s}_{\mathrm{pub}}, \mathfrak{s}_{\mathrm{priv}})$ *takes as input two disjoint index sets, the public slot $\mathfrak{s}_{\mathrm{pub}}$ and the private slot $\mathfrak{s}_{\mathrm{priv}}$, and outputs a pair of master public key and master secret key* $(\mathsf{mpk}, \mathsf{msk})$*. The whole index set $\mathfrak{s}$ is $\mathfrak{s}_{\mathrm{pub}} \cup \mathfrak{s}_{\mathrm{priv}}$.*

- $\mathsf{KeyGen}(1^\lambda, \mathsf{msk}, [\![\mathbf{v}]\!]_2)$ *generates a secret key $\mathsf{sk}_{\mathbf{v}}$ for $\mathbf{v} \in \mathbb{Z}_{p(\lambda)}^{\mathfrak{s}}$.*

- $\mathsf{Enc}(1^\lambda, \mathsf{msk}, [\![\mathbf{u}]\!]_1)$ *generates a ciphertext $\mathsf{ct}_{\mathbf{u}}$ for $\mathbf{u} \in \mathbb{Z}_{p(\lambda)}^{\mathfrak{s}}$ using the master* secret *key.*

- $\mathsf{Dec}(1^\lambda, \mathsf{sk}_{\mathbf{v}}, \mathsf{ct}_{\mathbf{u}})$ *is supposed to compute $[\![\langle \mathbf{u}, \mathbf{v} \rangle]\!]_{\mathrm{T}}$.*

- $\mathsf{SlotEnc}(1^\lambda, \mathsf{mpk}, [\![\mathbf{u}]\!]_1)$ *generates a ciphertext ct for $\mathbf{u}|^{\mathfrak{s}}$ when given input $\mathbf{u} \in \mathbb{Z}_{p(\lambda)}^{\mathfrak{s}_{\mathrm{pub}}}$ using the master* public *key.*

Decryption correctness *requires that for all $\lambda \in \mathbb{N}$, all index set $\mathfrak{s}$, and all vectors $\mathbf{u}, \mathbf{v} \in \mathbb{Z}_{p(\lambda)}^{\mathfrak{s}}$,*

$$\Pr\left[\begin{array}{l} \mathsf{msk} \xleftarrow{\$} \mathsf{Setup}(1^\lambda, \mathfrak{s}) \\ \quad \mathsf{sk} \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, \mathsf{msk}, [\![\mathbf{v}]\!]_2) \ : \ \mathsf{Dec}(1^\lambda, \mathsf{sk}, \mathsf{ct}) = [\![\langle \mathbf{u}, \mathbf{v} \rangle]\!]_{\mathrm{T}} \\ \quad \mathsf{ct} \xleftarrow{\$} \mathsf{Enc}(1^\lambda, \mathsf{msk}, [\![\mathbf{u}]\!]_1) \end{array}\right] = 1.$$

Slot-mode correctness *requires that for all* $\lambda \in \mathbb{N}$*, all disjoint index sets* $\mathfrak{s}_{\text{pub}}, \mathfrak{s}_{\text{priv}}$*, and all vector* $\mathbf{u} \in \mathbb{Z}_{p(\lambda)}^{\mathfrak{s}_{\text{pub}}}$*, the following distributions should be identical:*

$$\left\{ \begin{array}{r} (\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, \mathfrak{s}_{\text{pub}}, \mathfrak{s}_{\text{priv}}) \\ \mathsf{ct} \xleftarrow{\$} \mathsf{Enc}(1^\lambda, \mathsf{msk}, [\![\mathbf{u}|^{\mathfrak{s}}]\!]_1) \end{array} : (\mathsf{mpk}, \mathsf{msk}, \mathsf{ct}) \right\},$$

$$\left\{ \begin{array}{r} (\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, \mathfrak{s}_{\text{pub}}, \mathfrak{s}_{\text{priv}}) \\ \mathsf{ct} \xleftarrow{\$} \mathsf{SlotEnc}(1^\lambda, \mathsf{mpk}, [\![\mathbf{u}]\!]_1) \end{array} : (\mathsf{mpk}, \mathsf{msk}, \mathsf{ct}) \right\}.$$

Slotted IPFE generalizes both secret-key and public-key IPFEs: A secret-key IPFE can be obtained by setting $\mathfrak{s}_{\text{pub}} = \varnothing$ and $\mathfrak{s}_{\text{priv}} = \mathfrak{s}$; a public-key IPFE can be obtained by setting $\mathfrak{s}_{\text{pub}} = \mathfrak{s}$ and $\mathfrak{s}_{\text{priv}} = \varnothing$.

We now define the adaptive *function-hiding* property.

Definition 7 (function-hiding slotted IPFE)**.** *Let* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{SlotEnc})$ *be a slotted IPFE. The scheme is* function-hiding *if* $\mathsf{Exp}_{\text{FH}}^0 \approx \mathsf{Exp}_{\text{FH}}^1$*, where* $\mathsf{Exp}_{\text{FH}}^b$ *for* $b \in \{0,1\}$ *is defined as follows:*

- **Setup.** Run the adversary $\mathcal{A}(1^\lambda)$ and receive two disjoint index sets $\mathfrak{s}_{\text{pub}}, \mathfrak{s}_{\text{priv}}$ from $\mathcal{A}$. Let $\mathfrak{s} = \mathfrak{s}_{\text{pub}} \cup \mathfrak{s}_{\text{priv}}$. Run $(\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, \mathfrak{s}_{\text{pub}}, \mathfrak{s}_{\text{priv}})$ and return mpk to $\mathcal{A}$.
- **Challenge.** Repeat the following for arbitrarily many rounds determined by $\mathcal{A}$: In each round, $\mathcal{A}$ has 2 options.
  - $\mathcal{A}$ can submit $[\![\mathbf{v}_j^0]\!]_2, [\![\mathbf{v}_j^1]\!]_2$ for a secret key, where $\mathbf{v}_j^0, \mathbf{v}_j^1 \in \mathbb{Z}_p^{\mathfrak{s}}$. Upon this query, run $\mathsf{sk}_j \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, \mathsf{msk}, [\![\mathbf{v}_j^b]\!]_2)$ and return sk_j to $\mathcal{A}$.
  - $\mathcal{A}$ can submit $[\![\mathbf{u}_i^0]\!]_1, [\![\mathbf{u}_i^1]\!]_1$ for a ciphertext, where $\mathbf{u}_i^0, \mathbf{u}_i^1 \in \mathbb{Z}_p^{\mathfrak{s}}$. Upon this query, run $\mathsf{ct}_i \xleftarrow{\$} \mathsf{Enc}(1^\lambda, \mathsf{msk}, [\![\mathbf{u}_i^b]\!]_1)$ and return ct_i to $\mathcal{A}$.
- **Guess.** $\mathcal{A}$ outputs a bit b'. The outcome is b' if $\mathbf{v}_j^0|_{\mathfrak{s}_{\text{pub}}} = \mathbf{v}_j^1|_{\mathfrak{s}_{\text{pub}}}$ for all j and $\langle \mathbf{u}_i^0, \mathbf{v}_j^0 \rangle = \langle \mathbf{u}_i^1, \mathbf{v}_j^1 \rangle$ for all i, j. Otherwise, the outcome is 0.

Lemma 8 ([ALS16,Wee17,LV16,Lin17])**.** *Let* $\mathcal{G}$ *be a sequence of pairing groups and* $k \geq 1$ *an integer constant. If* MDDH_k *holds in both* G_1, G_2*, then there is an (adaptively) function-hiding slotted IPFE scheme based on* $\mathcal{G}$*.*

We present the construction in Appendix A, which is inspired by the selectively secure slotted IPFE scheme from SXDH in [LV16,Lin17].

4 Computation Models

We will be considering computation models computing arithmetic functions, i.e., functions with domain $\mathbb{Z}_p^{\mathcal{I}}$ for some prime p and index set $\mathcal{I}$ and with codomain $\mathbb{Z}_p$. Since we will construct ABE schemes for certain arithmetic functions, we first associate them with predicates. There are two natural ways of associating arithmetic functions with predicates — whether or not the output is zero, and whether or not the output is non-zero.

Definition 9 (zero-test predicates)**.** *Let* $f : \mathbb{Z}_p^{\mathcal{I}} \to \mathbb{Z}_p$ *be a function, define the* zero-test predicates *for* f *as*

$$f_{\neq 0} : \mathbb{Z}_p^{\mathcal{I}} \to \{0,1\}, \mathbf{x} \mapsto \begin{cases} 0, & f(\mathbf{x}) = 0; \\ 1, & f(\mathbf{x}) \neq 0; \end{cases} \qquad f_{=0} : \mathbb{Z}_p^{\mathcal{I}} \to \{0,1\}, \mathbf{x} \mapsto \neg f_{\neq 0}(\mathbf{x}).$$

4.1 Arithmetic Branching Programs

Arithmetic branching program[12] (ABP) is a computation model introduced by Nisan [Nis91] and later studied in [BG98,IK97,IK00,IK02,IW14]. We focus on the variant over $\mathbb{Z}_p$ in this paper.

Definition 10. *An* arithmetic branching program (ABP) *over $\mathbb{Z}_p^{\mathcal{I}}$ is a weighted directed acyclic graph (V, E, w, s, t) with two distinguished vertices, where V is the set of vertices, E is the set of edges, $w : E \to (\mathbb{Z}_p^{\mathcal{I}} \to \mathbb{Z}_p)$ specifies an affine weight function for each edge, and $s, t \in V$ are the distinguished vertices. The in-degree of s and the out-degree of t are* 0*. Such a program* computes *a function $f : \mathbb{Z}_p^{\mathcal{I}} \to \mathbb{Z}_p$ defined by*

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

The size *of the ABP is $|V|$, the number of vertices. Lastly, denote by* ABP *the class of all ABPs:*

$$\mathsf{ABP} = \left\{ f \;\middle|\; f \text{ is an ABP over } \mathbb{Z}_p^{\mathcal{I}} \text{ for some prime } p \text{ and index set } \mathcal{I} \right\}.$$

When there is no ambiguity, we use f to represent both the ABP and the function it computes, and ABP both the class of all ABPs and the class of all functions computed by some ABP.

ABP as Restricted Multiplication. Alternatively, one can think of an ABP as a straight-line program with addition and restricted multiplication where one multiplicand must be affine in the input. Let $f = (V, E, w, s, t)$ be some ABP of size m. Let $s = v_0, v_1, \ldots, v_{m-1} = t$ be the vertices sorted topologically. The function $f(\mathbf{x})$ can be computed by the following recurrence:

$$y_0 = 1, \qquad y_j = \sum_{(v_i, v_j) \in E} w(v_i, v_j)(\mathbf{x}) \cdot y_i, \qquad f(\mathbf{x}) = y_{m-1}.$$

Note that in the recurrence relation, we have $i < j$ by the sorting.

ABP as Determinant. It is known [IK97,IK02] that one can compute an ABP by computing the determinant of a special matrix:

Lemma 11 (Lemma 1 in [IK02])**.** *Let $f = (V, E, w, s, t)$ be an ABP of size m and $s = v_0, v_1, \ldots, v_{m-1} = t$ be sorted topologically. Let $\mathbf{M}$ be an $(m-1) \times (m-1)$ matrix whose entries are*

$$\mathbf{M}[i+1, j] = \begin{cases} 0, & i > j; \\ -1, & i = j; \\ 0, & i < j, (v_i, v_j) \notin E; \\ w(v_i, v_j)(\mathbf{x}), & i < j, (v_i, v_j) \in E. \end{cases}$$

Then the entries of $\mathbf{M}$ are affine in $\mathbf{x}$ and $f(\mathbf{x}) = \det \mathbf{M}$.

[12] The model is also known as algebraic branching programs or mod-p counting branching programs, though the precise definitions vary.

4.2 Turing Machines

The other computation model we consider in this work is Turing machines with a read-only input tape and a read-write work tape. This type of Turing machine is used to define space-bounded complexity classes of decision problems, and in this work, we construct an ABE scheme for uniform logspace predicates. One tricky issue with handling Turing machines is that the time and space complexity of a Turing machine is uncomputable. However, our ABE construction can only handle computation with polynomial time complexity and logarithmic space complexity. Below we define time/space bounded computation of a Turing machine. Correspondingly, in our ABE scheme, the attribute specifies, in addition to an input to the Turing machine, upper bounds on the concrete time complexity T and space complexity S, and decryption succeeds if and only if the machine accepts the input within this time/space bound. We will also confine ourselves to the binary alphabet. The definition below captures these features explicitly.

Definition 12 (Turing machine & time/space bounded computation)**.** *A (deterministic)* Turing machine *(over $\{0,1\}$) is a triplet $M = (Q, \mathbf{y}_{\mathrm{acc}}, \delta)$, where $Q \geq 1$ is the number of states (we use $[Q]$ as the set of states and 1 as the initial state), $\mathbf{y}_{\mathrm{acc}} \in \{0,1\}^Q$ indicates whether each state is accepting, and*

$$\begin{aligned}\delta : [Q] \times \{0,1\} \times \{0,1\} &\to [Q] \times \{0,1\} \times \{0,\pm 1\} \times \{0,\pm 1\},\\ (q, x, w) &\mapsto (q', w', \Delta i, \Delta j)\end{aligned}$$

is the state transition function, which, given the current state q, the symbol x on the input tape under scan, and the symbol w on the work tape under scan, specifies the new state q', the symbol w' overwriting w, the direction Δi to which the input tape pointer moves, and the direction Δj to which the work tape pointer moves. The machine is required to hang (instead of halting) *once it reaches an accepting state, i.e., for all $q \in [Q]$ such that $\mathbf{y}_{\mathrm{acc}}[q] = 1$ and all $x, w \in \{0,1\}$, it holds that $\delta(q, x, w) = (q, w, 0, 0)$.*

For input length $N \geq 1$ and space complexity bound $S \geq 1$, the set of internal configurations *of M is*

$$\mathcal{C}_{M,N,S} = [N] \times [S] \times \{0,1\}^S \times [Q],$$

where $(i, j, \mathbf{W}, q) \in \mathcal{C}_{M,N,S}$ specifies the input tape pointer $i \in [N]$, the work tape pointer $j \in [S]$, the content of the work tape $\mathbf{W} \in \{0,1\}^S$ and the machine state $q \in [Q]$.

For any bit-string $\mathbf{x} \in \{0,1\}^N$ for $N \geq 1$ and time/space complexity bounds $T, S \geq 1$, the machine M accepts $\mathbf{x}$ within time T and space S *if there exists a sequence of internal configurations (*computation path *of T steps) $c_0, \ldots, c_T \in \mathcal{C}_{M,N,S}$ with $c_t = (i_t, j_t, \mathbf{W}_t, q_t)$ such that*

$$\begin{aligned}&i_0 = 1,\ j_0 = 1,\ \mathbf{W}_0 = \mathbf{0}_S,\ q_0 = 1 \quad \text{(initial configuration)};\\ \text{for } 0 \leq t < T\text{:}\quad &\begin{cases}\delta(q_t, \mathbf{x}[i_t], \mathbf{W}_t[j_t]) = (q_{t+1}, \mathbf{W}_{t+1}[j_t], i_{t+1} - i_t, j_{t+1} - j_t),\\ \mathbf{W}_{t+1}[j] = \mathbf{W}_t[j] \text{ for all } j \neq j_t \quad \text{(valid transitions)};\end{cases}\\ &\mathbf{y}_{\mathrm{acc}}[q_T] = 1 \quad \text{(accepting)}.\end{aligned}$$

Denote by $M|_{N,T,S}$ the function $\{0,1\}^N \to \{0,1\}$ mapping $\mathbf{x}$ to whether M accepts $\mathbf{x}$ in time T and space S. Define $\mathsf{TM} = \{M \mid M \text{ is a Turing machine}\}$ to be the set of all Turing machines.

Remarks. The above definition disallows moving off the input/work tape. For example, if δ specifies moving the input tape pointer to the left when it is already at the leftmost position, there is no valid next internal configuration.

We can avoid the problem of moving off the input tape by encoding the input string $ab\cdots cd$ as $1a0b0\cdots0c0d1$. Here, the 1's at the ends indicate moving to one direction (the last direction to which the input tape pointer moved, or left initially) should be avoided, and the interleaving 0's indicate that both directions are allowed. The machine uses 2 additional bits in its state to remember

- whether it is reading an indicator bit or an input bit (initially "indicator"), and
- to which direction the input tape pointer last moved (initially "left").

The machine can maintain those bits and avoid moving off the input tape accordingly.

We can also eliminate the problem of moving off the work tape *to the left* by encoding $ab\cdots cd$ on the work tape as $1a0b0\cdots0c0d000\cdots$, where the leading 1 indicates moving to the left should be avoided, and the interleaving 0's indicate both directions are allowed. The machine uses one additional bit in its state and avoids moving off the work tape to the left accordingly.

The problem of moving off the work tape *to the right* is undetectable by the machine, and this is intended. When the space bound is violated, the input is *silently* rejected.

Relations Among Computation Models. A Boolean (resp. arithmetic) formula can be converted to a Boolean (resp. arithmetic) branching program in linear time, and a Boolean branching program (BP) can be trivially converted into an ABP. Therefore, ABP is a model that generalizes all the three without degrade in efficiency. Moreover, if a non-deterministic BP is unambiguous (i.e., there is at most one accepting path), it can be trivially converted to an ABP; otherwise, it can be approximated by an ABP of the same size with random weights. Lastly, a family of polynomial-sized ABPs can decide languages in $\mathsf{NL/poly}$ [BG98]. Note that this does not mean ABE for ABP trivially implies ABE for NL. In the former, each secret key is associated with a specific ABP instance thus only works for a specific input length, whereas in the latter, each secret key is associated with a Turing machine and can work for any input length.

5 Arithmetic Key Garbling Scheme

Arithmetic key garbling scheme (AKGS) is an information-theoretic primitive related to randomized encodings [AIK11] and partial garbling schemes [IW14]. It is the information-theoretic core in our construction of one-key one-ciphertext ABE (more precisely 1-ABE constructed in Section 6.2). Given a function $f : \mathbb{Z}_p^{\mathcal{I}} \to \mathbb{Z}_p$ and two secrets $\alpha, \beta \in \mathbb{Z}_p$, an AKGS produces label functions $L_1, \ldots, L_m : \mathbb{Z}_p^{\mathcal{I}} \to \mathbb{Z}_p$ that are affine in $\mathbf{x}$. For any $\mathbf{x}$, one can compute $\alpha f(\mathbf{x}) + \beta$ from $L_1(\mathbf{x}), \ldots, L_m(\mathbf{x})$ together with f and $\mathbf{x}$, while all other information about α, β are hidden.

Definition 13 (AKGS, adopted from Definition 1 in [IW14])**.** *An* arithmetic key garbling scheme (AKGS) *for a function class $\mathcal{F} = \{f\}$, where $f : \mathbb{Z}_p^{\mathcal{I}} \to \mathbb{Z}_p$ for some $p, \mathcal{I}$ specified by f, consists of two efficient algorithms:*

- $\mathsf{Garble}(f \in \mathcal{F}, \alpha \in \mathbb{Z}_p, \beta \in \mathbb{Z}_p)$ *is randomized and outputs m affine functions $L_1, \ldots, L_m : \mathbb{Z}_p^{\mathcal{I}} \to \mathbb{Z}_p$ (called* label functions*, which specifies how input is encoded as labels). Pragmatically, it outputs the coefficient vectors $\mathbf{L}_1, \ldots, \mathbf{L}_m$.*
- $\mathsf{Eval}(f \in \mathcal{F}, \mathbf{x} \in \mathbb{Z}_p^{\mathcal{I}}, \ell_1 \in \mathbb{Z}_p, \ldots, \ell_m \in \mathbb{Z}_p)$ *is deterministic and outputs a value in $\mathbb{Z}_p$ (the input $\ell_1, \ldots, \ell_m$ are called* labels*, which are supposed to be the values of the label functions at $\mathbf{x}$).*

Correctness *requires that for all* $f : \mathbb{Z}_p^{\mathcal{I}} \to \mathbb{Z}_p \in \mathcal{F}, \alpha, \beta \in \mathbb{Z}_p, \mathbf{x} \in \mathbb{Z}_p^{\mathcal{I}}$,

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

We also require that the scheme have deterministic shape*, meaning that* m *is determined solely by* f*, independent of* α, β *and the randomness in* Garble*. The number of label functions,* m*, is called the* garbling size *of* f *under this scheme.*

Definition 14 (linear AKGS)**.** *An AKGS* (Garble, Eval) *for* $\mathcal{F}$ *is* linear *if the following conditions hold:*

- $\mathsf{Garble}(f, \alpha, \beta)$ *uses a* **uniformly random** *vector* $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_p^{m'}$ *as its randomness, where* m' *is determined solely by* f*, independent of* α, β*.*
- *The coefficient vectors* $\mathbf{L}_1, \ldots, \mathbf{L}_m$ *produced by* $\mathsf{Garble}(f, \alpha, \beta; \mathbf{r})$ *are linear in* $(\alpha, \beta, \mathbf{r})$*.*
- $\mathsf{Eval}(f, \mathbf{x}, \ell_1, \ldots, \ell_m)$ *is linear in* $(\ell_1, \ldots, \ell_m)$*.*

Later in this paper, AKGS refers to linear AKGS by default.

5.1 Security Notions of AKGS

The basic security notion of AKGS requires the existence of an efficient simulator that draws a sample from the real labels' distribution given $f, \mathbf{x}, \alpha f(\mathbf{x}) + \beta$. We emphasize, as it's the same case in [IW14], that AKGS does *not* hide $\mathbf{x}$ and hides all other information about α, β except the value $\alpha f(\mathbf{x}) + \beta$.

Definition 15 ((usual) simulation security, Definition 1 in [IW14])**.** *An AKGS* (Garble, Eval) *for* $\mathcal{F}$ *is* secure *if there exists an efficient algorithm* Sim *such that for all* $f : \mathbb{Z}_p^{\mathcal{I}} \to \mathbb{Z}_p \in \mathcal{F}, \alpha, \beta \in \mathbb{Z}_p, \mathbf{x} \in \mathbb{Z}_p^{\mathcal{I}}$*, the following distributions are identical:*

$$\left\{\begin{array}{r}(\mathbf{L}_1, \ldots, \mathbf{L}_m) \xleftarrow{\$} \mathsf{Garble}(f, \alpha, \beta) \\ \ell_j \leftarrow L_j(\mathbf{x}) \text{ for } j \in [m]\end{array} : (\ell_1, \ldots, \ell_m)\right\},$$
$$\{\ (\ell_1, \ldots, \ell_m) \xleftarrow{\$} \mathsf{Sim}(f, \mathbf{x}, \alpha f(\mathbf{x}) + \beta) : (\ell_1, \ldots, \ell_m)\}.$$

As discussed in Section 2.1, the usual simulation security suffices for selective (or semi-adaptive) security. To achieve adaptive security, we need the following stronger property.

Definition 16 (piecewise security)**.** *An AKGS* (Garble, Eval) *for* $\mathcal{F}$ *is* piecewise secure *if the following conditions hold:*

- *The first label is* reversely sampleable *from the other* labels *together with* f *and* $\mathbf{x}$*. This reconstruction is perfect even given all the other* label functions*. Formally, there exists an efficient algorithm* RevSamp *such that for all* $f : \mathbb{Z}_p^{\mathcal{I}} \to \mathbb{Z}_p \in \mathcal{F}, \alpha, \beta \in \mathbb{Z}_p, \mathbf{x} \in \mathbb{Z}_p^{\mathcal{I}}$*, the following distributions are identical:*

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

- *For the other* labels*, each is* marginally random *even given all the* label functions *after it. Formally, this means for all* $f : \mathbb{Z}_p^{\mathcal{I}} \to \mathbb{Z}_p \in \mathcal{F}, \alpha, \beta \in \mathbb{Z}_p, \mathbf{x} \in \mathbb{Z}_p^{\mathcal{I}}$ *and all* $j \in [m], j > 1$*, the following distributions are identical:*

$$\left\{ \begin{array}{c} (\mathbf{L}_1, \ldots, \mathbf{L}_m) \xleftarrow{\$} \mathsf{Garble}(f, \alpha, \beta) \\ \ell_j \leftarrow L_j(\mathbf{x}) \end{array} : (\ell_j, \mathbf{L}_{j+1}, \ldots, \mathbf{L}_m) \right\},$$
$$\left\{ \begin{array}{c} (\mathbf{L}_1, \ldots, \mathbf{L}_m) \xleftarrow{\$} \mathsf{Garble}(f, \alpha, \beta) \\ \ell_j \xleftarrow{\$} \mathbb{Z}_p \end{array} : (\ell_j, \mathbf{L}_{j+1}, \ldots, \mathbf{L}_m) \right\}.$$

Lemma 17. *A piecewise secure AKGS* $(\mathsf{Garble}, \mathsf{Eval})$ *for some function class* $\mathcal{F}$ *is also secure.*

Proof. Assuming the AKGS is piecewise secure, we let the simulator $\mathsf{Sim}(f, \mathbf{x}, \gamma)$ sample $\ell_j \xleftarrow{\$} \mathbb{Z}_p$ for $1 < j \leq m$, reversely sample $\ell_1 \xleftarrow{\$} \mathsf{RevSamp}(f, \mathbf{x}, \gamma, \ell_2, \ldots, \ell_m)$, and output $(\ell_1, \ldots, \ell_m)$. It is readily verified that Sim efficiently and perfectly simulates the labels. Therefore, the AKGS is also secure. □

Special Piecewise Security. We identify a special structural form of AKGS, referred to as special piecewise security. It is equivalent to piecewise security, and has the advantage of being easier to verify. In particular, all AKGS schemes considered in this work have this form.

Definition 18. *An AKGS* $(\mathsf{Garble}, \mathsf{Eval})$ *for* $\mathcal{F}$ *is* special piecewise secure *if it has the following special form:*

- *For all* $f : \mathbb{Z}_p^{\mathcal{I}} \to \mathbb{Z}_p$ *in* $\mathcal{F}$ *and all* $\mathbf{x} \in \mathbb{Z}_p^{\mathcal{I}}$, $\mathsf{Eval}(f, \mathbf{x}, \overset{\ell_1}{\downarrow} 1, \overset{\ell_2}{\downarrow} 0, \ldots, \overset{\ell_m}{\downarrow} 0) \neq 0$.
- *For all* $f : \mathbb{Z}_p^{\mathcal{I}} \to \mathbb{Z}_p \in \mathcal{F}, \alpha, \beta \in \mathbb{Z}_p$*, let* $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_p^{m'}$ *be the randomness* $\mathsf{Garble}(f, \alpha, \beta)$ *uses. For all* $j \in [m], j > 1$*, the label function* L_j *produced by* $\mathsf{Garble}(f, \alpha, \beta; \mathbf{r})$ *can be written as*

$$L_j(\mathbf{x}) = k_j \mathbf{r}[j-1] + L'_j(\mathbf{x}; \alpha, \beta, \mathbf{r}[\geq j]),$$

  where $k_j \in \mathbb{Z}_p$ *is a* non-zero constant *(not depending on* $\mathbf{x}, \alpha, \beta, \mathbf{r}$*) and* L'_j *is an affine function of* $\mathbf{x}$ *whose coefficient vector is linear in* $(\alpha, \beta, \mathbf{r}[j], \mathbf{r}[j+1], \ldots)$*. The component* $\mathbf{r}[j-1]$ *is called the* randomizer *of* L_j *and* ℓ_j*.*

We note that the first requirement of special piecewise security, on top of Eval being linear in the labels (always required as a linear AKGS), simply says the coefficient of ℓ_1 is always non-zero. An AKGS with the special form is clearly piecewise secure.

Lemma 19. *A special piecewise secure AKGS* $(\mathsf{Garble}, \mathsf{Eval})$ *for some function class* $\mathcal{F}$ *is also piecewise secure. Moreover, the* $\mathsf{RevSamp}$ *algorithm (required in piecewise security) obtained for a special piecewise secure AKGS is linear in* $\gamma, \ell_2, \ldots, \ell_m$ *and perfectly recovers* ℓ_1 *even if the randomness of* Garble *is not uniformly sampled.*

Proof. Assuming the AKGS is special piecewise secure, we first show that the first label is reversely sampleable. Since Eval is linear in the labels, we have

$$\mathsf{Eval}(f, \mathbf{x}, \ell_1, \ell_2, \ldots, \ell_m) = \ell_1 \mathsf{Eval}(f, \mathbf{x}, 1, 0, \ldots, 0) + \mathsf{Eval}(f, \mathbf{x}, 0, \ell_2, \ldots, \ell_m).$$

Define the reverse sampling algorithm as

$$\mathsf{RevSamp}(f, \mathbf{x}, \gamma, \ell_2, \ldots, \ell_m) = \big(\mathsf{Eval}(f, \mathbf{x}, 1, 0, \ldots, 0)\big)^{-1} \big(\gamma - \mathsf{Eval}(f, \mathbf{x}, 0, \ell_2, \ldots, \ell_m)\big),$$

which is clearly efficient and linear in $\gamma, \ell_2, \ldots, \ell_m$. Note that given $f, \mathbf{x}, \gamma = \alpha f(\mathbf{x}) + \beta, \ell_2, \ldots, \ell_m$, the first label ℓ_1 is uniquely determined due to the requirement of Eval in special piecewise security, and the algorithm $\mathsf{RevSamp}$ defined above computes this value. Since ℓ_1 is uniquely determined, it does not matter if the label functions $\mathbf{L}_2, \ldots, \mathbf{L}_m$ are revealed, nor if the garbling is not generated using uniform randomness.

For marginal randomness property, for all $1 < j \leq m$, given the label functions $\mathbf{L}_{j+1}, \ldots, \mathbf{L}_m$, the label $\ell_j = k_j \mathbf{r}[j-1] + L'_j(\mathbf{x})$ is uniformly random since $\mathbf{r}[j-1]$ is a uniformly random value independent of $\mathbf{L}'_j, \mathbf{L}_{j+1}, \ldots, \mathbf{L}_m$. □

A piecewise secure AKGS can be converted into a special piecewise secure one. We present the proof of the following theorem in Appendix C:

Theorem 20. *A piecewise secure AKGS is also special piecewise secure after an appropriate change of variable for the randomness used by* Garble.

6 ABE for ABPs

In this section, we show how to construct a KP-ABE for ABPs in three steps: *i*) construct a piecewise secure AKGS for ABPs; *ii*) build a 1-key 1-ciphertext secure secret-key ABE (1-ABE) using a piecewise secure AKGS and a function-hiding secret-key IPFE; *iii*) lift the 1-ABE into a full-fledged ABE using slotted IPFE.

The first step is specific to the class of predicates for which we want to construct an ABE. The second and third steps are independent of the predicates as long as all the predicates share the same domain.

6.1 AKGS for ABPs

Our piecewise secure AKGS for ABPs is a special case of the partial garbling scheme for ABPs in [IW14], which is in turn built upon randomizing polynomials [IK00,IK02]. In a partial garbling scheme (PGS), the input of a function $f : \mathbb{Z}_p^{\mathcal{I}_x} \times \mathbb{Z}_p^{\mathcal{I}_z} \to \mathbb{Z}_p$ is divided into two parts, the public part $\mathbf{x} \in \mathbb{Z}_p^{\mathcal{I}_x}$ and the private part $\mathbf{z} \in \mathbb{Z}_p^{\mathcal{I}_z}$. The labels for f with input $\mathbf{x}, \mathbf{z}$ in a secure PGS contain no other information about $\mathbf{z}$ except the value $f(\mathbf{x}, \mathbf{z})$, but might otherwise completely reveal $\mathbf{x}$.

More specifically, our AKGS garbling for computation $f(x)$ with secrets α, β is essentially a PGS garbling for the computation $\alpha f(x) + \beta$, where x is the the public input and α, β are the private inputs. In this special case (slightly more generally, when the computation has degree 1 in the private inputs), the PGS garbling of [IW14] satisfies the linearity structure and the (usual) simulation security of AKGS. We further show that it also satisfies the *piecewise security*.

AKGS and PGS have slightly different syntax. In an AKGS, the label functions L_j's are regarded as affine functions of $\mathbf{x}$ with $(\alpha, \beta, \mathbf{r})$ hardwired, and the coefficient vectors $\mathbf{L}_j$ are linear in $(\alpha, \beta, \mathbf{r})$. In a PGS, L_j's are regarded as affine functions of $(\mathbf{x}, \alpha, \beta)$ with $\mathbf{r}$ hardwired, and the coefficient vectors $\mathbf{L}_j$ are affine in $\mathbf{r}$. Below, we recall the PGS for ABPs [IW14] in the syntax of an AKGS.

Construction 21 (AKGS for ABPs [IW14])**.** Let $\mathcal{F} = \mathsf{ABP}$ be the class of all ABPs. The AKGS $(\mathsf{Garble}, \mathsf{Eval})$ for $\mathcal{F}$ operates as follows: (We inline comments to and explanation of the scheme in italics.)

- $\mathsf{Garble}(f, \alpha, \beta)$ takes an ABP $f \in \mathcal{F} = \mathsf{ABP}$ and two secrets α, β as input. Suppose f is over $\mathbb{Z}_p^{\mathcal{I}}$ and of size m, and $\alpha, \beta \in \mathbb{Z}_p$. The algorithm computes the matrix $\mathbf{M}$ in Lemma 11 (whose

determinant is the output of the function) and augments it into an $m \times m$ matrix $\mathbf{M}'$:

$$\mathbf{M}' = \left(\begin{array}{ccccc:c} & & \overset{\mathbf{M}}{\downarrow} & & & \\ * & * & \cdots & * & * & \beta \\ -1 & * & \cdots & * & * & 0 \\ & -1 & \cdots & * & * & 0 \\ & & \ddots & \vdots & \vdots & \vdots \\ 0 & & & -1 & * & 0 \\ \hdashline 0 & 0 & \cdots & 0 & -1 & \alpha \end{array}\right) = \begin{pmatrix} \mathbf{M} & \mathbf{m}_1 \\ \mathbf{m}_2^\intercal & \alpha \end{pmatrix}.$$

Here, the augmented matrix $\mathbf{M}'$ is the matrix for an ABP computing $\alpha f(\mathbf{x}) + \beta$. It then samples $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_p^{m-1}$ as its randomness, sets $\mathbf{N} = \begin{pmatrix} \mathbf{I}_{m-1} & \mathbf{r} \\ \mathbf{0} & 1 \end{pmatrix}$, and defines the label functions by

$$\widehat{\mathbf{M}} = \mathbf{M}'\mathbf{N} = \begin{pmatrix} \mathbf{M} & \mathbf{M}\mathbf{r} + \mathbf{m}_1 \\ \mathbf{m}_2^\intercal & \mathbf{m}_2^\intercal \mathbf{r} + \alpha \end{pmatrix} = \left(\begin{array}{ccccc:c} & & & & & L_1(\mathbf{x}) \\ & & \mathbf{M} & & & L_2(\mathbf{x}) \\ & & & & & \vdots \\ & & & & & L_{m-1}(\mathbf{x}) \\ \hdashline 0 & 0 & \cdots & 0 & -1 & L_m(\mathbf{x}) \end{array}\right).$$

The algorithm collects the coefficient vectors $\mathbf{L}_1, \ldots, \mathbf{L}_m$ of $L_1, \ldots, L_m$ and returns them.

Note: *We show* Garble *satisfies the required properties of a linear AKGS (Definitions 13 and 14):*

- *$L_1, \ldots, L_m$ are affine functions of $\mathbf{x}$:* $\mathbf{M}'$ *is affine in* $\mathbf{x}$ *and* $\mathbf{N}$ *is constant with respect to* $\mathbf{x}$*, therefore, the label functions* $L_1, \ldots, L_m$ *(the last column of* $\mathbf{M}'\mathbf{N}$*) are affine in* $\mathbf{x}$*.*
- *Shape determinism holds: The garbling size of* f *is its size as an ABP.*
- Garble *is linear in* $(\alpha, \beta, \mathbf{r})$*, i.e., the coefficients of* $L_1, \ldots, L_m$ *are linear in* $(\alpha, \beta, \mathbf{r})$*:* $\mathbf{M}$ *is constant with respect to* $(\alpha, \beta, \mathbf{r})$*, and* $\mathbf{r}, \mathbf{m}_1, \alpha$ *are linear in* $(\alpha, \beta, \mathbf{r})$*, so* $\mathbf{M}\mathbf{r} + \mathbf{m}_1$ *and* $\mathbf{M}\mathbf{r} + \alpha$ *(hence the coefficients of* $L_1, \ldots, L_{m-1}$ *and* L_m*) are linear in* $(\alpha, \beta, \mathbf{r})$*.*

- $\mathsf{Eval}(f, \mathbf{x}, \ell_1, \ldots, \ell_m)$ takes an ABP $f \in \mathcal{F} = \mathsf{ABP}$ (over $\mathbb{Z}_p^{\mathcal{I}}$ and of size m), an input $\mathbf{x} \in \mathbb{Z}_p^{\mathcal{I}}$ and m labels $\ell_1, \ldots, \ell_m$ as input. It computes $\mathbf{M}$ (with the value of $\mathbf{x}$ substituted) in Lemma 11 and sets

$$\widehat{\mathbf{M}} = \left(\begin{array}{ccccc:c} & & & & & \ell_1 \\ & & \mathbf{M} & & & \ell_2 \\ & & & & & \vdots \\ & & & & & \ell_{m-1} \\ \hdashline 0 & 0 & \cdots & 0 & -1 & \ell_m \end{array}\right).$$

The algorithm returns $\det \widehat{\mathbf{M}}$.

Note: *We show the correctness of the scheme. Since $\ell_j = L_j(\mathbf{x})$ for all $j \in [m]$, the matrix $\widehat{\mathbf{M}}$ in* Eval *is the matrix $\widehat{\mathbf{M}}$ in* Garble *with* $\mathbf{x}$ *plugged into it, and we have*

$$\begin{aligned} \det \widehat{\mathbf{M}} &= \det(\mathbf{M}'\mathbf{N}) = \det \mathbf{M}' \det \mathbf{N} \\ \left(\begin{smallmatrix}\text{Laplace expansion in} \\ \text{the last column of } \mathbf{M}'\end{smallmatrix}\right) \quad &= \det \mathbf{M}' \\ &= \alpha \det \mathbf{M} + \beta \\ (\text{Lemma 11}) \quad &= \alpha f(\mathbf{x}) + \beta. \end{aligned}$$

Eval *is also linear in $\ell_1, \ldots, \ell_m$. This follows by the Laplace expansion of $\det \widehat{\mathbf{M}}$ in the last column.*

Security. Ishai and Wee [IW14] observed that in Garble in Construction 21, all the possible $\mathbf{N}$'s form a matrix subgroup

$$H = \left\{ \begin{pmatrix} \mathbf{I}_{m-1} & \mathbf{r} \\ \mathbf{0} & 1 \end{pmatrix} \middle| \, \mathbf{r} \in \mathbb{Z}_p^{m-1} \right\}.$$

Moreover, the labels are the last column of a uniformly random element from the left coset $\mathbf{M}'H = \{\mathbf{M}'\mathbf{N} \,|\, \mathbf{N} \in H\}$. The (usual) simulation security follows from the observation that each such left coset has a canonical representative $\mathbf{M}''$ determined by $f, \mathbf{x}$ and $\alpha f(\mathbf{x}) + \beta$:

$$\mathbf{M}'' = \begin{pmatrix} & & & & \alpha f(\mathbf{x}) + \beta \\ & \mathbf{M} & & & 0 \\ & & & & \vdots \\ & & & & 0 \\ 0 & 0 & \cdots & 0 & -1 \quad 0 \end{pmatrix} \in \mathbf{M}'H.$$

By the properties of cosets, $\mathbf{M}''H = \mathbf{M}'H$, therefore, given $f, \mathbf{x}, \alpha f(\mathbf{x}) + \beta$, one can sample a uniformly random element from $\mathbf{M}''H = \mathbf{M}'H$ to simulate the labels.

We now proceed to proving the stronger security notions introduced in this work.

Theorem 22. *Construction 21 is special piecewise secure.*

Proof. Adopt the notations in Construction 21. We first show that the coefficient of ℓ_1 in Eval is non-zero. Recall that

$$\widehat{\mathbf{M}} = \begin{pmatrix} & \mathbf{M} & & & & \\ & \downarrow & & & & \\ & * & * & \cdots & * & * \quad \ell_1 \\ & -1 & * & \cdots & * & * \quad \ell_2 \\ & & -1 & \cdots & * & * \quad \ell_3 \\ & & & \ddots & \vdots & \vdots \quad \vdots \\ & 0 & & & -1 & * \quad \ell_{m-1} \\ 0 & 0 & 0 & \cdots & 0 & -1 \quad \ell_m \end{pmatrix}.$$

Consider the Laplace expansion of $\det \widehat{\mathbf{M}}$ in the the last column:

$$\mathsf{Eval}(f, \mathbf{x}, \ell_1, \ldots, \ell_m) = \det \widehat{\mathbf{M}} = A_1\ell_1 + A_2\ell_2 + \cdots + A_m\ell_m,$$

where A_j is the (j,m)-cofactor of $\widehat{\mathbf{M}}$. Observe that by definition, each A_j is solely determined by $(f,\mathbf{x})$, so the above expansion expresses Eval as a linear function of the labels, and in particular, the coefficient of ℓ_1 is $A_1 = (-1)^{1+m}(-1)^{m-1} = 1 \neq 0$.

Now we show that the label functions are in the special form. Recall that the label functions are defined by the last column of

$$\widehat{\mathbf{M}} = \mathbf{M}'\mathbf{N} = \begin{pmatrix} \mathbf{M}[1,1] & \mathbf{M}[1,2] & \cdots & \mathbf{M}[1,m-2] & \mathbf{M}[1,m-1] & \beta \\ -1 & \mathbf{M}[2,2] & \cdots & \mathbf{M}[2,m-2] & \mathbf{M}[2,m-1] & 0 \\ & -1 & \cdots & \mathbf{M}[3,m-2] & \mathbf{M}[3,m-1] & 0 \\ & & \ddots & \vdots & \vdots & \vdots \\ & & & -1 & \mathbf{M}[m-1,m-1] & 0 \\ & & & & -1 & \alpha \end{pmatrix} \begin{pmatrix} 1 & & & & \mathbf{r}[1] \\ & 1 & & & \mathbf{r}[2] \\ & & 1 & & \mathbf{r}[3] \\ & & & \ddots & \vdots \\ & & & & 1\ \mathbf{r}[m-1] \\ & & & & 1 \end{pmatrix}.$$

Computing the label functions $L_2, \ldots, L_m$, we get

$$\text{for } 1 < j < m\text{:} \quad \begin{aligned} L_j(\mathbf{x}) &= \underset{\substack{\uparrow \\ k_j = -1}}{-\mathbf{r}[j-1]} + \sum_{j'=j}^{m-1} \underset{\substack{\uparrow \\ L'_j(\mathbf{x};\alpha,\beta,\mathbf{r}_{\geq j})}}{\mathbf{r}[j']}\mathbf{M}[j,j'], \\ L_m(\mathbf{x}) &= \underset{\substack{\uparrow \\ k_m = -1}}{-\mathbf{r}[m-1]} + \underset{\substack{\uparrow \\ L_m(\mathbf{x};\alpha,\beta)}}{\alpha}. \end{aligned}$$

They are in the required special form (the randomizer of L_j is $\mathbf{r}[j-1]$).

Combining the two, we conclude that Construction 21 is special piecewise secure. □

6.2 1-Key 1-Ciphertext Secure Secret-Key ABE

At the core of our adaptively secure ABE is a construction for the simple case of 1-key 1-ciphertext secure secret-key ABE — we call it 1-ABE. (For technical reasons, it is more convenient to define it as a key encapsulation mechanism.) It captures our key ideas for achieving adaptive security using AKGS and function-hiding IPFE while keeping the ciphertext compact. Below we start with defining the syntax and security of 1-ABE.

Definition 23. *Let $\mathcal{G}$ be a sequence of pairing groups of order $p(\lambda)$. A* 1-ABE scheme based on $\mathcal{G}$ *has the same syntax as an ABE scheme in Definition 4, except that*

- *There is no message space $\mathcal{M}$.*
- Setup *outputs a master secret key* msk*, without a* mpk*.*
- $\mathsf{KeyGen}(1^\lambda, \mathsf{msk}, y, \mu)$ *outputs a secret key* sk *for policy y that encapsulates a pad $\mu \in \mathbb{Z}_{p(\lambda)}$.*
- $\mathsf{Enc}(1^\lambda, \mathsf{msk}, x)$ *uses* msk *and outputs a ciphertext* ct *for attribute x without encrypting a message.*
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$ *outputs $\perp$ or some $[\![\mu']\!]_\mathrm{T}$.*
- *Correctness requires that $\mu = \mu'$ if the decapsulation should be successful, i.e., $P(x,y) = 1$.*

Such a scheme is 1-key 1-ciphertext secure *(or simply* secure*) if $\mathsf{Exp}^0_{\text{1-sk,1-ct}} \approx \mathsf{Exp}^1_{\text{1-sk,1-ct}}$, where $\mathsf{Exp}^b_{\text{1-sk,1-ct}}$ is defined similarly to $\mathsf{Exp}^b_{\mathrm{CPA}}$ in Definition 5:*

- *In* **Setup**, *the adversary* $\mathcal{A}$ *chooses a predicate* P *but does not receive a* mpk.
- *In* **Query I/II**, *when* $\mathcal{A}$ *submits a key query* y, *sample two random pads* $\mu^0, \mu^1 \xleftarrow{\$} \mathbb{Z}_{p(\lambda)}$, *run* $\mathsf{sk} \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, \mathsf{msk}, y, \mu^0)$, *and return* (sk, μ^b) *to* $\mathcal{A}$.
- *In* **Challenge**, *when* $\mathcal{A}$ *submits a challenge attribute* x, *run* $\mathsf{ct} \xleftarrow{\$} \mathsf{Enc}(1^\lambda, \mathsf{msk}, x)$, *and return* ct *to* $\mathcal{A}$.
- *In* **Guess**, $\mathcal{A}$ *outputs a bit* b'. *The outcome of the experiment is* b' *if the adversary makes only a single key query for some* y *and* $P(x, y) = 0$. *Otherwise, the outcome is* 0.

For any function class $\mathcal{F}$ (e.g., arithmetic branching programs), we show how to construct a 1-ABE for the class of zero-test predicates in $\mathcal{F}$ (i.e., predicates of form $f_{\neq 0}, f_{=0}$ that computes whether $f(\mathbf{x})$ evaluates to zero or non-zero), using a piecewise secure AKGS for $\mathcal{F}$ and a function-hiding secret-key IPFE scheme.

Construction 24 (1-ABE)**.** We describe the construction for any fixed value of the security parameter λ and suppress the appearance of λ below for simplicity of notations. Let $(\mathsf{Garble}, \mathsf{Eval})$ be an AKGS for a function class $\mathcal{F}$, $\mathcal{G}$ pairing groups of order p, and $(\mathsf{IPFE.Setup}, \mathsf{IPFE.KeyGen}, \mathsf{IPFE.Enc}, \mathsf{IPFE.Dec})$ a secret-key IPFE based on $\mathcal{G}$. We construct a 1-ABE scheme based on $\mathcal{G}$ for the predicate space $\mathcal{P}$ induced by $\mathcal{F}$:

$$X_n = \mathbb{Z}_p^n, \quad Y_n = \{f_{\neq 0}, f_{=0} \mid f \in \mathcal{F}, f : \mathbb{Z}_p^n \to \mathbb{Z}_p\},$$
$$\mathcal{P} = \{P_n : X_n \times Y_n \to \{0,1\}, (\mathbf{x}, y) \mapsto y(\mathbf{x}) \mid n \in \mathbb{N}\}.$$

The 1-ABE scheme $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ operates as follows:

- $\mathsf{Setup}(1^n)$ takes the attribute length in unary (i.e., P_n is encoded as 1^n) as input. It generates an IPFE master secret key $\mathsf{msk} \xleftarrow{\$} \mathsf{IPFE.Setup}(\mathfrak{s}_{\text{1-ABE}})$ for the index set $\mathfrak{s}_{\text{1-ABE}} = \{\mathsf{const}, \mathsf{coef}_1, \ldots, \mathsf{coef}_n, \mathsf{sim}_1, \mathsf{sim}_\star\}$. The algorithm returns msk as the master secret key.

  Note: *The positions indexed by* $\mathsf{const}, \mathsf{coef}_1, \ldots, \mathsf{coef}_n$ *in the secret key encode the coefficient vectors* $\mathbf{L}_j$ *of the label functions* L_i *produced by garbling* f *with secrets* α, β, *and these positions encode* $(1, \mathbf{x})$ *in the ciphertext. The positions indexed by* $\mathsf{sim}_1, \mathsf{sim}_\star$ *are set to zero by the honest algorithms, and are only used in the security proof.*

- $\mathsf{KeyGen}(\mathsf{msk}, y \in Y_n, \mu \in \mathbb{Z}_p)$ samples $\eta \xleftarrow{\$} \mathbb{Z}_p$ and garbles the function f underlying y as follows:

$$\begin{cases} \alpha \leftarrow \mu, \ \ \beta \leftarrow 0, & \text{if } y = f_{\neq 0}; \\ \alpha \leftarrow \eta, \ \ \beta \leftarrow \mu, & \text{if } y = f_{=0}; \end{cases} \qquad (\mathbf{L}_1, \ldots, \mathbf{L}_m) \xleftarrow{\$} \mathsf{Garble}(f, \alpha, \beta).$$

  It generates an IPFE key $\mathsf{isk}_j \xleftarrow{\$} \mathsf{IPFE.KeyGen}(\mathsf{msk}, [\![\mathbf{v}_j]\!]_2)$ for the following vector $\mathbf{v}_j$ encoding each label function L_j:

vector	const	coef_i	sim_1	$\mathsf{sim}_\star$
$\mathbf{v}_j$	$\mathbf{L}_j[\mathsf{const}]$	$\mathbf{L}_j[\mathsf{coef}_i]$	0	0

  The algorithm returns $\mathsf{sk}_y = (y, \mathsf{isk}_1, \ldots, \mathsf{isk}_m)$ as the secret key.

- $\mathsf{Enc}(\mathsf{msk}, \mathbf{x} \in \mathbb{Z}_p^n)$ generates an IPFE ciphertext $\mathsf{ict} \xleftarrow{\$} \mathsf{IPFE.Enc}(\mathsf{msk}, [\![\mathbf{u}]\!]_1)$ encrypting the vector $\mathbf{u}$ that contains $1, \mathbf{x}$:

vector	const	coef_i	sim_1	$\mathsf{sim}_\star$
$\mathbf{u}$	1	$\mathbf{x}[i]$	0	0

  It returns $\mathsf{ct} = (\mathbf{x}, \mathsf{ict})$ as the ciphertext.

- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$ parses sk as $(y, \mathsf{isk}_1, \ldots, \mathsf{isk}_m)$ and ct as $(\mathbf{x}, \mathsf{ict})$, and returns $\perp$ if $y(\mathbf{x}) = 0$. Otherwise, it does the following:

$$\text{for } j \in [m]: \quad [\![\ell_j]\!]_{\mathrm{T}} \leftarrow \mathsf{IPFE.Dec}(\mathsf{isk}_j, \mathsf{ict}),$$
$$[\![\mu']\!]_{\mathrm{T}} \leftarrow \begin{cases} \frac{1}{f(\mathbf{x})}\mathsf{Eval}(f, \mathbf{x}, [\![\ell_1]\!]_{\mathrm{T}}, \ldots, [\![\ell_m]\!]_{\mathrm{T}}), & \text{if } y = f_{\neq 0}; \\ \mathsf{Eval}(f, \mathbf{x}, [\![\ell_1]\!]_{\mathrm{T}}, \ldots, [\![\ell_m]\!]_{\mathrm{T}}), & \text{if } y = f_{=0}. \end{cases}$$

  The algorithm returns $[\![\mu']\!]_{\mathrm{T}}$ as the decapsulated pad.

  Note: *We show the correctness of the scheme. First, by the correctness of IPFE and the definition of vectors $\mathbf{v}_j, \mathbf{u}$, we have $\ell_j = \langle \mathbf{u}, \mathbf{v}_j \rangle = L_j(\mathbf{x})$ for all $j \in [m]$. Next, by the linearity of* Eval *in $\ell_1, \ldots, \ell_m$, we can evaluate the garbling in the exponent of the target group and obtain $\mathsf{Eval}(f, \mathbf{x}, \ell_1, \ldots, \ell_m) = \alpha f(\mathbf{x}) + \beta$ in the exponent. In the two cases where decapsulation should succeed, we have*

$$\alpha f(\mathbf{x}) + \beta = \begin{cases} \mu f(\mathbf{x}), & \text{if } y = f_{\neq 0} \text{ and } f(\mathbf{x}) \neq 0; \\ \mu, & \text{if } y = f_{=0} \text{ and } f(\mathbf{x}) = 0. \end{cases}$$

  In both cases, the μ' above equals to μ. Therefore, Dec *correctly decapsulates the pad.*

Theorem 25. *Suppose in Construction 24, the AKGS is* piecewise secure *and the IPFE scheme is* function-hiding*, then the constructed 1-ABE scheme is* 1-key 1-ciphertext secure*.*

Proof. Let $\mathcal{A}$ be any efficient adversary. We want to show that the advantage of $\mathcal{A}$ in distinguishing $\mathsf{Exp}^0_{\text{1-sk,1-ct}}$ and $\mathsf{Exp}^1_{\text{1-sk,1-ct}}$ is negligible. Recall that in $\mathsf{Exp}^b_{\text{1-sk,1-ct}}$ for $b \in \{0, 1\}$, the adversary is allowed to obtain a single key sk for a policy y with μ^0 encapsulated, and a single ciphertext ct for an attribute $\mathbf{x}$, where the policy y and the attribute $\mathbf{x}$ are chosen adaptively by $\mathcal{A}$ and satisfy that either $y = f_{=0}$ and $f(\mathbf{x}) \neq 0$, or $y = f_{\neq 0}$ and $f(\mathbf{x}) = 0$. (If these constraints are not satisfied, the outcome of the experiment is set to 0.) In addition, $\mathcal{A}$ receives μ^b with sk and wants to distinguish the two experiments.

Since the views of $\mathcal{A}$ in $\mathsf{Exp}^0_{\text{1-sk,1-ct}}$ and $\mathsf{Exp}^1_{\text{1-sk,1-ct}}$ are identical before $\mathcal{A}$ receives (sk, μ^b), we can analyze the indistinguishability of these two experiments in two cases:

- Case 1: The ciphertext challenge $\mathbf{x}$ appears before the secret key query y. In this case, adaptive security collapses down to selective security. This is because by the time the secret key sk needs to be generated, the challenge attribute $\mathbf{x}$ is already known. The proof then proceeds in two simple steps: *i*) By the function-hiding property of IPFE, we can remove the coefficient vectors $\mathbf{L}_j$ of the label functions L_j encoded in isk_j, and hardwire directly the labels $\ell_j = L_j(\mathbf{x})$ in isk_j. (Such hardwiring is possible thanks to the fact that sk contains many IPFE secret keys, which provide sufficient space for embedding ℓ_j's.) *ii*) After replacing $\mathbf{L}_j$'s with ℓ_j's, by the (usual) simulation security of AKGS, the hardwired labels $\{\ell_j\}_{j \in [m]}$ is independent of μ^0 whenever decapsulation should fail. Since μ^0 and μ^1 are identically distributed, the two hybrids are identical.

- Case 2: The ciphertext challenge $\mathbf{x}$ appears after the secret key query y. In this case, at the time sk needs to be generated, the adversary has not chosen the challenge attribute $\mathbf{x}$. As a result, the above argument breaks down at the step of hardwiring: We can no longer hardwire the labels $\{\ell_j = L_j(\mathbf{x})\}_{j\in[m]}$ in sk, as $\mathbf{x}$ is not yet known. Nor can we hardwire the labels in ct, as ct contains only a single IPFE ciphertext ict and does not have enough space for embedding $\{\ell_j\}_{j\in[m]}$. We resolve this by relying on the piecewise security of AKGS. Roughly speaking, it allows us to gradually switch the coefficient vectors $\mathbf{L}_j$ of the label functions L_j to simulated labels $\ell_j \xleftarrow{\$} \mathbb{Z}_p$ via a series of hybrids. Between neighboring hybrids, only 2 label functions/labels are changed, which can be hardwired in ict.

Below, we first prove security in the simpler Case 1, and then move to the proof of Case 2.

Proof of Case 1. Consider the following hybrids:

- Hybrid H_0^b proceeds identically to $\mathsf{Exp}^b_{\text{1-sk,1-ct}}$, where the vector $\mathbf{u}$ encrypted in the IPFE ciphertext ict (in the ABE ciphertext ct) and the vectors $\mathbf{v}_j$ encoded in the IPFE secret keys isk_j (in the 1-ABE secret key sk) are described in Figure 1. Note that $\mathbf{u}$ appears before $\mathbf{v}_j$ in the experiment.

- Hybrid H_1^b proceeds identically to H_0^b, except that it sets $\mathbf{v}_j[\mathsf{const}]$ to the desired inner product of $\mathbf{v}_j$ with $\mathbf{u}$ — the honest labels $\ell_j = L_j(\mathbf{x})$ — and other positions to 0, as described in Figure 1. Since the inner products $\langle \mathbf{u}, \mathbf{v}_j \rangle$ are the same in H_1^b and H_0^b, it follows directly from the function-hiding property of IPFE that these two hybrids are indistinguishable.

- Hybrid H_2^b proceeds identically to H_1^b, except that the hardwired labels $(\ell_1, \ldots, \ell_m)$ are now simulated using $\mathsf{Sim}(f, \mathbf{x}, \alpha f(\mathbf{x}) + \beta)$. By the usual security of AKGS, H_2^b and H_1^b are identically distributed.

hybrid	vector	const	coef_i	sim_1	$\mathsf{sim}_\star$
$\mathsf{H}_0^b \equiv \mathsf{Exp}^b_{\text{1-sk,1-ct}}$	$\mathbf{u}$	1	$\mathbf{x}[i]$	0	0
	$\mathbf{v}_j$	$\mathbf{L}_j[\mathsf{const}]$	$\mathbf{L}_j[\mathsf{coef}_i]$	0	0
H_1^b	$\mathbf{u}$	1	$\mathbf{x}[i]$	0	0
	$\mathbf{v}_j$	$\ell_j = L_j(\mathbf{x})$	0	0	0
H_2^b	$\mathbf{u}$	1	$\mathbf{x}[i]$	0	0
	$\mathbf{v}_j$	simulated ℓ_j	0	0	0
Simulation in H_2^b: $(\ell_1, \ldots, \ell_m) \xleftarrow{\$} \mathsf{Sim}(f, \mathbf{x}, \alpha f(\mathbf{x}) + \beta)$.					

Figure 1: Hybrids in the security proof of 1-ABE for ABPs for the case where the ciphertext challenge comes before the secret key query.

We further argue that H_2^0 and H_2^1 are identically distributed. In these hybrids, the labels are simulated as $\mathsf{Sim}(f, \mathbf{x}, \alpha f(\mathbf{x}) + \beta)$. In the two cases where decapsulation should fail, we have

$$\alpha f(\mathbf{x}) + \beta = \begin{cases} 0, & \text{if } y = f_{\neq 0} \text{ and } f(\mathbf{x}) = 0; \\ \eta f(\mathbf{x}) + \mu^0 \quad (\eta \xleftarrow{\$} \mathbb{Z}_p), & \text{if } y = f_{=0} \text{ and } f(\mathbf{x}) \neq 0. \end{cases}$$

In both cases, $\alpha f(\mathbf{x}) + \beta$ is independent of μ^0, hence μ^0 is a uniform random value independent of everything else in H_2^0 — so is μ^1 in H_2^1. In addition, everything except μ^0, μ^1 are identically distributed in H_2^0 and H_2^1. Therefore, they are identical.

By a hybrid argument, we conclude that $\mathsf{Exp}^0_{\text{1-sk,1-ct}} \approx \mathsf{Exp}^1_{\text{1-sk,1-ct}}$ conditioned on Case 1.

Proof of Case 2. We prove $\mathsf{Exp}^0_{\text{1-sk,1-ct}} \approx \mathsf{Exp}^1_{\text{1-sk,1-ct}}$ conditioned on Case 2 via a series of hybrids. The hybrids are logically divided into 3 groups: *i*) The first few hybrids remove the first label function, and hardwire the first label that is reversely sampled; *ii*) The middle hybrids are a loop, and in each iteration one label function is replaced by a simulated label; *iii*) In the final hybrid, all the labels are simulated, and the encapsulated pad μ^0 is information-theoretically hidden.

We start with the first few hybrids.

- Hybrid H_0^b proceeds identically to $\mathsf{Exp}^b_{\text{1-sk,1-ct}}$, where the vector $\mathbf{u}$ encrypted in the IPFE ciphertext ict (in the ABE ciphertext ct) and the vectors $\mathbf{v}_j$ encoded in the IPFE secret keys isk_j (in the ABE secret key sk) are described in Figure 2. Different from Case 1, the vector $\mathbf{u}$ appears *after* $\mathbf{v}_j$ in the experiment.

- Hybrid H_1^b proceeds identically to H_0^b, except that it removes the coefficients of the first label function $\mathbf{L}_1$ from $\mathbf{v}_1[\mathsf{const}], \mathbf{v}_1[\mathsf{coef}_i]$, and hardwires the honest first label $\ell_1 = L_1(\mathbf{x})$ in $\mathbf{u}[\mathsf{sim}_1]$ as described in Figure 2. Since the inner product remains the same as in H_0^b, by the function-hiding property of IPFE, we have that H_1^b and H_0^b are indistinguishable.

- Hybrid H_2^b proceeds identically to H_1^b, except that the hardwired first label is now reversely sampled as

$$\ell_1 \xleftarrow{\$} \mathsf{RevSamp}(f, \mathbf{x}, \alpha f(\mathbf{x}) + \beta, \ell_2, \ldots, \ell_m),$$

  where $\ell_j = L_j(\mathbf{x})$ for $j > 1$ are the honest labels. By the reverse sampleability of AKGS (Definition 16), H_2^b and H_1^b are identically distributed.

Hybrid H_4^b is our final hybrid, which proceeds identically to H_2^b, except that for all $j > 1$, the coefficient vector $\mathbf{L}_j$ of the label function L_j is removed from $\mathbf{v}_j$, and a simulated label $\ell_j \xleftarrow{\$} \mathbb{Z}_p$ is hardwired in $\mathbf{v}_j[\mathsf{const}]$, as depicted in Figure 2. Correspondingly, the first label ℓ_1 is reversely sampled using these simulated labels. The hybrid is similar to H_2^b in Case 1, except for the location where ℓ_1 is embedded. However, we cannot move from H_2^b to H_4^b in one shot. Instead, we go through a loop consisting of $\mathsf{H}^b_{3,j,1}, \ldots, \mathsf{H}^b_{3,j,4}$ for $1 < j \leq m$:

- Hybrid $\mathsf{H}^b_{3,j,1}$ proceeds identically to H_2^b, except that for all j' s.t. $1 < j' < j$, the coefficient vector $\mathbf{L}_{j'}$ of the label function $L_{j'}$ is removed from $\mathbf{v}_{j'}$, and a simulated label $\ell_{j'} \xleftarrow{\$} \mathbb{Z}_p$ is hardwired in $\mathbf{v}_{j'}[\mathsf{const}]$, as described in Figure 3. In $\mathsf{H}^b_{3,j,1}$, ℓ_1 is reversely sampled using simulated $\ell_2, \ldots, \ell_{j-1}$ and honest $\ell_j, \ldots, \ell_m$.

- Hybrid $\mathsf{H}^b_{3,j,2}$ proceeds identically to $\mathsf{H}^b_{3,j,1}$, except that it removes the coefficient vector of the j^{th} label function L_j from $\mathbf{v}_j$, and hardwires the honest j^{th} label $\ell_j = L_j(\mathbf{x})$ in $\mathbf{u}[\mathsf{sim}_\star]$. Note that $\mathbf{L}_j$ is gone in $\mathsf{H}^b_{3,j,2}$ and the honest label ℓ_j is left. The modified vectors $\mathbf{v}_j$ and $\mathbf{u}$ are described in Figure 3. Since the inner products remain the same, by the function-hiding property of IPFE, $\mathsf{H}^b_{3,j,2}$ and $\mathsf{H}^b_{3,j,1}$ are indistinguishable.

- Hybrid $\mathsf{H}^b_{3,j,3}$ proceeds identically to $\mathsf{H}^b_{3,j,2}$, except that the j^{th} label is simulated as $\ell_j \xleftarrow{\$} \mathbb{Z}_p$. Note that in hybrids $\mathsf{H}^b_{3,j,2}$ and $\mathsf{H}^b_{3,j,3}$, only the coefficient vectors $\mathbf{L}_{j'}$ of the label functions

hybrid	vector	const	coef_i	sim_1	$\mathsf{sim}_\star$
$\mathsf{H}_0^b \equiv \mathsf{Exp}^b_{\text{1-sk,1-ct}}$	$\mathbf{v}_1$	$\mathbf{L}_1[\mathsf{const}]$	$\mathbf{L}_1[\mathsf{coef}_i]$	0	0
	$j>1$: $\mathbf{v}_j$	$\mathbf{L}_j[\mathsf{const}]$	$\mathbf{L}_j[\mathsf{coef}_i]$	0	0
	$\mathbf{u}$	1	$\mathbf{x}[i]$	0	0
H_1^b	$\mathbf{v}_1$	0	0	1	0
	$j>1$: $\mathbf{v}_j$	$\mathbf{L}_j[\mathsf{const}]$	$\mathbf{L}_j[\mathsf{coef}_i]$	0	0
	$\mathbf{u}$	1	$\mathbf{x}[i]$	$\ell_1 = L_1(\mathbf{x})$	0
$\mathsf{H}_2^b \equiv \mathsf{H}_{3,2,1}^b$	$\mathbf{v}_1$	0	0	1	0
	$j>1$: $\mathbf{v}_j$	$\mathbf{L}_j[\mathsf{const}]$	$\mathbf{L}_j[\mathsf{coef}_i]$	0	0
	$\mathbf{u}$	1	$\mathbf{x}[i]$	$\ell_1 \xleftarrow{\$} \mathsf{RevSamp}(\cdots)$	0
$\mathsf{H}^b_{3,2\sim m,1\sim 4}$			$\cdots\cdots$		
$\mathsf{H}_4^b \equiv \mathsf{H}_{3,m,4}^b$	$\mathbf{v}_1$	0	0	1	0
	$j>1$: $\mathbf{v}_j$	$\ell_j \xleftarrow{\$} \mathbb{Z}_p$	0	0	0
	$\mathbf{u}$	1	$\mathbf{x}[i]$	$\ell_1 \xleftarrow{\$} \mathsf{RevSamp}(\cdots)$	0

Reverse sampling in $\mathsf{H}_2^b, \mathsf{H}_4^b$: $\ell_1 \xleftarrow{\$} \mathsf{RevSamp}(f, \mathbf{x}, \alpha f(\mathbf{x}) + \beta, \ell_2, \ldots, \ell_m)$.
In H_2^b, the labels ℓ_j $(j > 1)$ are computed honestly as $L_j(\mathbf{x})$ using IPFE.
In H_4^b, they are simulated as random and hardwired in sk.

Figure 2: The first few hybrids and the final hybrid in the security proof of 1-ABE for ABPs for the case where the ciphertext challenge comes after the secret key query.

hybrid	vector	const	coef_i	sim_1	$\mathsf{sim}_\star$
$\mathsf{H}^b_{3,j,1}$	$\mathbf{v}_1$	0	0	1	0
	$1 < j' < j$: $\mathbf{v}_{j'}$	$\ell_{j'} \xleftarrow{\$} \mathbb{Z}_p$	0	0	0
	$\mathbf{v}_j$	$\mathbf{L}_j[\mathsf{const}]$	$\mathbf{L}_j[\mathsf{coef}_i]$	0	0
	$j' > j$: $\mathbf{v}_{j'}$	$\mathbf{L}_{j'}[\mathsf{const}]$	$\mathbf{L}_{j'}[\mathsf{coef}_i]$	0	0
	$\mathbf{u}$	1	$\mathbf{x}[i]$	ℓ_1	0
$\mathsf{H}^b_{3,j,2}$	$\mathbf{v}_1$	0	0	1	0
	$1 < j' < j$: $\mathbf{v}_{j'}$	$\ell_{j'} \xleftarrow{\$} \mathbb{Z}_p$	0	0	0
	$\mathbf{v}_j$	0	0	0	1
	$j' > j$: $\mathbf{v}_{j'}$	$\mathbf{L}_{j'}[\mathsf{const}]$	$\mathbf{L}_{j'}[\mathsf{coef}_i]$	0	0
	$\mathbf{u}$	1	$\mathbf{x}[i]$	ℓ_1	$\ell_j = L_j(\mathbf{x})$
$\mathsf{H}^b_{3,j,3}$	$\mathbf{v}_1$	0	0	1	0
	$1 < j' < j$: $\mathbf{v}_{j'}$	$\ell_{j'} \xleftarrow{\$} \mathbb{Z}_p$	0	0	0
	$\mathbf{v}_j$	0	0	0	1
	$j' > j$: $\mathbf{v}_{j'}$	$\mathbf{L}_{j'}[\mathsf{const}]$	$\mathbf{L}_{j'}[\mathsf{coef}_i]$	0	0
	$\mathbf{u}$	1	$\mathbf{x}[i]$	ℓ_1	$\ell_j \xleftarrow{\$} \mathbb{Z}_p$
$\mathsf{H}^b_{3,j,4}$	$\mathbf{v}_1$	0	0	1	0
	$1 < j' < j$: $\mathbf{v}_{j'}$	$\ell_{j'} \xleftarrow{\$} \mathbb{Z}_p$	0	0	0
$\equiv$	$\mathbf{v}_j$	$\ell_j \xleftarrow{\$} \mathbb{Z}_p$	0	0	0
$\mathsf{H}^b_{3,j+1,1}$	$j' > j$: $\mathbf{v}_{j'}$	$\mathbf{L}_{j'}[\mathsf{const}]$	$\mathbf{L}_{j'}[\mathsf{coef}_i]$	0	0
	$\mathbf{u}$	1	$\mathbf{x}[i]$	ℓ_1	0

In this iteration, the labels $\ell_{j'}$ are...

- $j' = 1$: simulated as $\mathsf{RevSamp}(f, \mathbf{x}, \alpha f(\mathbf{x}) + \beta, \ell_2, \ldots, \ell_m)$ and hardwired in ct
- $1 < j' < j$: simulated as random and hardwired in sk
- $j' = j$: computed as $L_j(\mathbf{x})$ using IPFE
  - $\to$ computed as $L_j(\mathbf{x})$ and hardwired in ct
  - $\to$ simulated as random and hardwired in ct
  - $\to$ simulated as random and hardwired in sk
- $j' > j$: computed as $L_{j'}(\mathbf{x})$ using IPFE

Figure 3: The loop hybrids in the security proof of 1-ABE for ABPs for the case where the ciphertext challenge comes after the secret key query.

$L_{j'}$ for $j' > j$ appear. For $1 < j' < j$, a simulated label $\ell_{j'}$ is hardwired in $\mathbf{v}_{j'}[\mathsf{const}]$. In both hybrids, the first label ℓ_1 can be efficiently computed from $\ell_2, \ldots, \ell_{j-1}$ (simulated), ℓ_j (honest or simulated) and $\mathbf{L}_{j+1}, \ldots, \mathbf{L}_m$ (producing honest labels). Therefore, by the marginal randomness property of AKGS (Definition 16), $\mathsf{H}^b_{3,j,3}$ and $\mathsf{H}^b_{3,j,2}$ are identically distributed.

- Hybrid $\mathsf{H}^b_{3,j,4}$ proceeds identically to $\mathsf{H}^b_{3,j,3}$, except that the simulated j^{th} label ℓ_j is moved from $\mathbf{u}[\mathsf{sim}_\star]$ to $\mathbf{v}_j[\mathsf{const}]$ as described in Figure 3 (so that $\mathbf{u}[\mathsf{sim}_\star]$ is free to be used in the next iteration of hybrids $\mathsf{H}^b_{3,j+1,1}$ to $\mathsf{H}^b_{3,j+1,4}$). Since the inner products remain the same, by the function-hiding property of IPFE, $\mathsf{H}^b_{3,j,4}$ and $\mathsf{H}^b_{3,j,3}$ are indistinguishable. In addition, observe that $\mathsf{H}^b_{3,j,4} \equiv \mathsf{H}^b_{3,j+1,1}$.

Moreover, $\mathsf{H}^b_2 \equiv \mathsf{H}^b_{3,2,1}$ and $\mathsf{H}^b_4 \equiv \mathsf{H}^b_{3,m,4}$. Thus, by a hybrid argument, we have $\mathsf{H}^b_2 \approx \mathsf{H}^b_4$. We further argue that H^0_4 and H^1_4 are identically distributed. Observe that in these hybrids, the labels are simulated as

$$\begin{aligned} \ell_{j'} &\xleftarrow{\$} \mathbb{Z}_p \quad \text{for } j' = 2, \ldots, m, \\ \ell_1 &\xleftarrow{\$} \mathsf{RevSamp}(f, \mathbf{x}, \alpha f(\mathbf{x}) + \beta, \ell_2, \ldots, \ell_m). \end{aligned}$$

In the two cases where decapsulation should fail, we have

$$\alpha f(\mathbf{x}) + \beta = \begin{cases} 0, & \text{if } y = f_{\neq 0} \text{ and } f(\mathbf{x}) = 0; \\ \eta f(\mathbf{x}) + \mu^0 \quad (\eta \xleftarrow{\$} \mathbb{Z}_p), & \text{if } y = f_{=0} \text{ and } f(\mathbf{x}) \neq 0. \end{cases}$$

In both cases, $\alpha f(\mathbf{x}) + \beta$ is independent of μ^0, hence μ^0 is a uniform random value independent of everything else in H^0_4 — so is μ^1 in H^1_4. In addition, everything except μ^0, μ^1 are identically distributed in H^0_4 and H^1_4. Therefore, they are identical.

By a hybrid argument, $\mathsf{Exp}^0_{\text{1-sk,1-ct}} \approx \mathsf{Exp}^1_{\text{1-sk,1-ct}}$ conditioned on Case 2. □

6.3 KP-ABE for ABPs

We now lift the 1-ABE scheme to a full-fledged ABE by implementing the ideas discussed in Section 2.2.

Construction 26 (KP-ABE)**.** We describe the construction for any fixed value of the security parameter λ and suppress the appearance of λ below for simplicity of notations. Let $(\mathsf{Garble}, \mathsf{Eval})$ be an AKGS for a function class $\mathcal{F}$, $\mathcal{G}$ pairing groups of order p such that MDDH_k holds in G_2, and $(\mathsf{IPFE.Setup}, \mathsf{IPFE.KeyGen}, \mathsf{IPFE.Enc}, \mathsf{IPFE.Dec})$ a slotted IPFE based on $\mathcal{G}$. We construct an ABE scheme for message space $M = G_{\mathrm{T}}$, the target group of the pairing, and the predicate space $\mathcal{P}$ induced by $\mathcal{F}$:

$$\begin{aligned} X_n &= \mathbb{Z}_p^n, \quad Y_n = \{f_{\neq 0}, f_{=0} \mid f \in \mathcal{F}, f : \mathbb{Z}_p^n \to \mathbb{Z}_p\}, \\ \mathcal{P} &= \{P_n : X_n \times Y_n \to \{0,1\}, (\mathbf{x}, y) \mapsto y(\mathbf{x}) \mid n \in \mathbb{N}\}. \end{aligned}$$

The ABE scheme $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ operates as follows:

- $\mathsf{Setup}(1^n)$ takes the attribute length in unary (i.e., P_n is encoded as 1^n) as input. It generates IPFE master public/secret key pair $(\mathsf{msk}, \mathsf{mpk}) \xleftarrow{\$} \mathsf{IPFE.Setup}(\mathfrak{s}_{\text{pub}}, \mathfrak{s}_{\text{priv}})$ for the following slots:

$$\begin{aligned} \mathfrak{s}_{\text{pub}} &= \{\mathsf{pad}, \mathsf{const}^t, \mathsf{coef}^t_i \mid t \in [k], i \in [n]\}, \\ \mathfrak{s}_{\text{priv}} &= \{\mathsf{const}, \mathsf{coef}_i, \mathsf{sim}_1, \mathsf{sim}_\star \mid i \in [n]\} \ (= \mathfrak{s}_{\text{1-ABE}}). \end{aligned}$$

It returns $(\mathsf{mpk}, \mathsf{msk})$ as the master public/secret key pair.

Note: *We explain the names of the indices. For each $t \in [k]$, the positions indexed by const^t, $\mathsf{coef}_1^t, \ldots, \mathsf{coef}_n^t$ encode the coefficient vectors $\mathbf{L}_j^t$ of the label functions of one garbling in the secret key, and encrypt a random multiple of $(1, \mathbf{x})$ in the ciphertext. The position indexed by pad encodes 1 in the secret key and encrypts a random pad $h \xleftarrow{\$} \mathbb{Z}_p$ in the ciphertext. The private slot is reserved for the security proof and values at those indices are set to 0 by the honest algorithms.*

- $\mathsf{KeyGen}(\mathsf{msk}, y \in Y_n)$ samples $\boldsymbol{\mu} \xleftarrow{\$} \mathbb{Z}_p^k, \boldsymbol{\eta} \xleftarrow{\$} \mathbb{Z}_p^k$ and creates k garblings of the function f underlying y as follows:

$$\begin{aligned} &\begin{cases} \boldsymbol{\alpha} \leftarrow \boldsymbol{\mu}, \quad \boldsymbol{\beta} \leftarrow \mathbf{0}, & \text{if } y = f_{\neq 0}; \\ \boldsymbol{\alpha} \leftarrow \boldsymbol{\eta}, \quad \boldsymbol{\beta} \leftarrow \boldsymbol{\mu}, & \text{if } y = f_{=0}; \end{cases} \\ \text{for } t \in [k]: \quad &(\mathbf{L}_1^t, \ldots, \mathbf{L}_m^t) \leftarrow \mathsf{Garble}(f, \boldsymbol{\alpha}[t], \boldsymbol{\beta}[t]; \mathbf{r}_t). \end{aligned}$$

  The randomness vectors $\mathbf{r}_1, \ldots, \mathbf{r}_k$ for garbling f are independently sampled and written explicitly. It sets the vector $\mathbf{v}_{\mathrm{pad}} \in \mathbb{Z}_p^{\mathfrak{s}}$ to encode the random pads $\boldsymbol{\mu}$, and the vector $\mathbf{v}_j \in \mathbb{Z}_p^{\mathfrak{s}}$ for $j \in [m]$ to encode the j^{th} label function in all the k instances of garbling:

vector	pad	const^t	coef_i^t	in $\mathfrak{s}_{\mathrm{priv}}$
$\mathbf{v}_{\mathrm{pad}}$	1	$\boldsymbol{\mu}[t]$	0	0
$\mathbf{v}_j$	0	$\mathbf{L}_j^t[\mathsf{const}]$	$\mathbf{L}_j^t[\mathsf{coef}_i]$	

  The algorithm then generates IPFE secret keys for them:

$$\begin{aligned} \mathsf{isk}_{\mathrm{pad}} &\xleftarrow{\$} \mathsf{IPFE.KeyGen}(\mathsf{msk}, [\![\mathbf{v}_{\mathrm{pad}}]\!]_2), \\ \text{for } j \in [m]: \quad \mathsf{isk}_j &\xleftarrow{\$} \mathsf{IPFE.KeyGen}(\mathsf{msk}, [\![\mathbf{v}_j]\!]_2). \end{aligned}$$

  It returns $\mathsf{sk} = (\mathsf{isk}_{\mathrm{pad}}, y, \mathsf{isk}_1, \ldots, \mathsf{isk}_m)$.

- $\mathsf{Enc}(\mathsf{mpk}, \mathbf{x} \in X_n, g \in M)$ samples a random pad $h \xleftarrow{\$} \mathbb{Z}_p$ and random multipliers $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_p^k$, and sets the vector $\mathbf{u} \in \mathbb{Z}_p^{\mathfrak{s}_{\mathrm{pub}}}$ to encode them as follows:

vector	pad	const^t	coef_i^t
$\mathbf{u}$	h	$\mathbf{s}[t]$	$\mathbf{s}[t]\mathbf{x}[i]$

  The algorithm generates an IPFE ciphertext $\mathsf{ict} \xleftarrow{\$} \mathsf{IPFE.SlotEnc}(\mathsf{mpk}, [\![\mathbf{u}]\!]_1)$ and returns $\mathsf{ct} = (g + [\![h]\!]_{\mathrm{T}}, \mathbf{x}, \mathsf{ict})$.

- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$ parses sk as $(\mathsf{isk}_{\mathrm{pad}}, y, \mathsf{isk}_1, \ldots, \mathsf{isk}_m)$ and ct as $([\![z]\!]_{\mathrm{T}}, \mathbf{x}, \mathsf{ict})$. It returns $\perp$ if $y(\mathbf{x}) = 0$. Otherwise, it does the following:

$$\begin{aligned} &[\![z']\!]_{\mathrm{T}} \leftarrow \mathsf{IPFE.Dec}(\mathsf{isk}_{\mathrm{pad}}, \mathsf{ict}), \\ \text{for } j \in [m]: \quad &[\![\ell_j]\!]_{\mathrm{T}} \leftarrow \mathsf{IPFE.Dec}(\mathsf{isk}_j, \mathsf{ict}), \\ &[\![\mu']\!]_{\mathrm{T}} \leftarrow \begin{cases} \frac{1}{f(\mathbf{x})}\mathsf{Eval}(f, \mathbf{x}, [\![\ell_1]\!]_{\mathrm{T}}, \ldots, [\![\ell_m]\!]_{\mathrm{T}}), & \text{if } y = f_{\neq 0}; \\ \mathsf{Eval}(f, \mathbf{x}, [\![\ell_1]\!]_{\mathrm{T}}, \ldots, [\![\ell_m]\!]_{\mathrm{T}}), & \text{if } y = f_{=0}. \end{cases} \end{aligned}$$

The algorithm returns $[\![z]\!]_{\mathrm{T}} + [\![\mu']\!]_{\mathrm{T}} - [\![z']\!]_{\mathrm{T}}$ as the decrypted message.

Note: *We show the correctness of the scheme. Consider the random linear combination $\overline{L}_j$ of label functions $L_j^1, \ldots, L_j^k$ with weights $\mathbf{s}$:*

$$\text{for } j \in [m]: \quad \overline{L}_j(\mathbf{x}) = \sum_{t\in[k]} \mathbf{s}[t] L_j^t(\mathbf{x}), \qquad \overline{\mathbf{L}}_j = \sum_{t\in[k]} \mathbf{s}[t] \mathbf{L}_j^t.$$

The coefficient vector of $\overline{L}_j$ is exactly $\overline{\mathbf{L}}_j$ defined above. Furthermore, by the linearity of Garble*, we know $(\overline{\mathbf{L}}_1, \ldots, \overline{\mathbf{L}}_m) = \mathsf{Garble}(f, \overline{\alpha}, \overline{\beta}; \overline{\mathbf{r}})$, where $\overline{\alpha}$, $\overline{\beta}$, and $\overline{\mathbf{r}}$ are the random linear combinations (with the same weights $\mathbf{s}$) of the components of $\boldsymbol{\alpha}$, the components of $\boldsymbol{\beta}$, and the randomness vectors $\{\mathbf{r}_t\}_{t\in[k]}$ in the k instances of garbling, i.e.,*

$$\overline{\alpha} = \langle \mathbf{s}, \boldsymbol{\alpha} \rangle, \quad \overline{\beta} = \langle \mathbf{s}, \boldsymbol{\beta} \rangle, \quad \overline{\mathbf{r}} = \sum_{t\in[k]} \mathbf{s}[t] \mathbf{r}_t.$$

Observe that by the definition of $\mathbf{v}_{\mathrm{pad}}, \mathbf{v}_j, \mathbf{u}$ and the correctness of the IPFE scheme,

$$z' = \langle \mathbf{u}, \mathbf{v}_{\mathrm{pad}} \rangle = h + \langle \mathbf{s}, \boldsymbol{\mu} \rangle = h + \overline{\mu},$$
$$\text{for } j \in [m]: \quad \ell_j = \langle \mathbf{u}, \mathbf{v}_j \rangle = \sum_{t\in[k]} \mathbf{s}[t] L_j^t(\mathbf{x}) = \overline{L}_j(\mathbf{x}),$$

where $\overline{\mu} = \langle \mathbf{s}, \boldsymbol{\mu} \rangle$ is the random linear combination of the components of $\boldsymbol{\mu}$ with weights $\mathbf{s}$. By the linearity of Eval *in $\ell_1, \ldots, \ell_m$, we can evaluate the garbling in the exponent of the target group and obtain $\mathsf{Eval}(f, \mathbf{x}, \ell_1, \ldots, \ell_m) = \overline{\alpha} f(\mathbf{x}) + \overline{\beta}$ in the exponent. In the two cases where decryption should succeed, we have*

$$\overline{\alpha} f(\mathbf{x}) + \overline{\beta} = \begin{cases} \overline{\alpha} f(\mathbf{x}) = \overline{\mu} f(\mathbf{x}), & \text{if } y = f_{\neq 0} \text{ and } f(\mathbf{x}) \neq 0; \\ \overline{\beta} = \overline{\mu}, & \text{if } y = f_{=0} \text{ and } f(\mathbf{x}) = 0. \end{cases}$$

In both cases, we have $\mu' = \overline{\mu}$, hence $[\![z]\!]_{\mathrm{T}} + [\![\mu']\!]_{\mathrm{T}} - [\![z']\!]_{\mathrm{T}} = g + [\![h]\!]_{\mathrm{T}} + [\![\overline{\mu}]\!]_{\mathrm{T}} - [\![h + \overline{\mu}]\!]_{\mathrm{T}} = g$, i.e., Dec *correctly recovers the message.*

Theorem 27. *Suppose in Construction 26, the AKGS is* piecewise secure*, the* MDDH_k *assumption holds in G_2, and the slotted IPFE is* function-hiding*, then the construction is a secure ABE scheme.*

Proof. Let $\mathcal{A}$ be any efficient adversary. We show that the distinguishing advantage of $\mathcal{A}$ against $\mathsf{Exp}^0_{\mathrm{CPA}}$ and $\mathsf{Exp}^1_{\mathrm{CPA}}$ is negligible. Recall that in $\mathsf{Exp}^b_{\mathrm{CPA}}$, the adversary can receive many secret keys sk_q for different policies y_q and a ciphertext ct for attribute $\mathbf{x}$ and message g_b, where $\{y_q\}_{q\in[Q]}$ and $(\mathbf{x}, g_0, g_1)$ are chosen adaptively by $\mathcal{A}$, subject to the constraint that no individual key should decrypt the challenge ciphertext, i.e., $P(\mathbf{x}, y_q) = 0$ for all $q \in [Q]$. (The output of the experiment is set to 0, if the constraint is not satisfied.)

We start with the high-level ideas of the proof. Recall that by construction, ct and sk_q contain respectively a slotted IPFE ciphertext ict and many IPFE secret keys $\mathsf{isk}_{q,\mathrm{pad}}, \{\mathsf{isk}_{q,j}\}_{j\in[m_q]}$ such that decryption computes in the exponent *i*) a garbling $\overline{L}_q(\mathbf{x})$ that reveals a random pad $\overline{\mu}_q$ from the evaluation result $\overline{\alpha}_q f_q(x) + \overline{\beta}_q$ if the policy is satisfied ($P(\mathbf{x}, y_q) = 1$), and *ii*) a sum $h + \overline{\mu}_q$. Revealing any $[\![\overline{\mu}_q]\!]_{\mathrm{T}}$ allows one to recover the message g_b from $g_b + [\![h]\!]_{\mathrm{T}}$ in ct. Therefore, to show $\mathsf{Exp}^0_{\mathrm{CPA}} \approx \mathsf{Exp}^1_{\mathrm{CPA}}$, we want to argue that when no policy is satisfied, all the pads $\overline{\mu}_q$ are pseudorandom and hide h.

To do so, the proof proceeds in three steps via a sequence of hybrids. In the first step — hybrids $\mathsf{H}_0^b \equiv \mathsf{Exp}_{\mathrm{CPA}}^b$ to H_3^b — we embed in the *private* slot of ict the vector $(1, \mathbf{x})$, in the *private* slot of $\mathsf{isk}_{q,\mathrm{pad}}$ *randomly and independently sampled* pads $\widehat{\mu}_q$, and in the *private* slot of $\mathsf{isk}_{q,j}$ *randomly and independently generated* label functions $\widehat{L}_{q,j}$ embedding $\widehat{\mu}_q$. In the second step — hybrids $\mathsf{H}_{4,1}^b \equiv \mathsf{H}_3^b$ to $\mathsf{H}_{4,Q+1}^b \equiv \mathsf{H}_5^b$ — we observe that the *private* slots of ict and $\mathsf{isk}_{q,j}$ match exactly the ciphertext and secret keys of our 1-ABE scheme (i.e., they encode the same vectors and have the function-hiding property). Therefore, using the same argument as the security proof of 1-ABE, we can switch the random pads $\widehat{\mu}_q$ encoded in $\mathsf{isk}_{q,\mathrm{pad}}$ to independent random values $\widehat{\mu}'_q$. Lastly, in the third step — H_5^b to H_6^b — we remove h from ict and $\widehat{\mu}'_q$ from $\mathsf{isk}_{q,\mathrm{pad}}$, and embed $h + \widehat{\mu}'_q$ in $\mathsf{sk}_{q,\mathrm{pad}}$. This allows us to argue that h is hidden by $\widehat{\mu}'_q$, and the message g_b is also hidden.

Next, we proceed to describing the hybrids formally.

- Hybrid H_0^b proceeds identically to $\mathsf{Exp}_{\mathrm{CPA}}^b$, where the vector $\mathbf{u}$ encrypted in the IPFE ciphertext ict (in ct), the vectors $\mathbf{v}_{q,j}$ encoded in the IPFE secret keys $\mathsf{isk}_{q,j}$ (in sk_q), and the vectors $\mathbf{v}_{q,\mathrm{pad}}$ in $\mathsf{isk}_{q,\mathrm{pad}}$ (in sk_q) are described in Figure 4. In particular, note that the values in the private slot of $\mathbf{u}$ are set to 0 and ict is generated using $\mathsf{IPFE.SlotEnc}$ (depicted as $\perp$ in the private slot of $\mathbf{u}$).

- Hybrid H_1^b proceeds identically to H_0^b, except that the IPFE ciphertext ict is generated using $\mathsf{IPFE.Enc}$ with values in the private slot set to 0. In Figure 4, the "$\perp$" values in H_0^b change into 0 in H_1^b. By the *slot-mode* correctness of the slotted IPFE scheme, H_0^b and H_1^b are identically distributed.

- Hybrid H_2^b proceeds identically to H_1^b, except that how the pad $\overline{\mu}_q$ and labels $\overline{L}_{q,j}(\mathbf{x})$ for $q \in [Q], j \in [m_q]$ are computed (as inner products) is changed — instead of combining k independent garblings (in the secret keys) with weights $\mathbf{s}$ (in the ciphertext), we put the k independent garblings as well as their linear combinations in the secret keys and the ciphertext only uses *that* linear combination.

  Recall that in H_1^b (the same as in $\mathsf{Exp}_{\mathrm{CPA}}^b$), $\overline{\mu}_q$ and $\overline{L}_{q,j}(\mathbf{x})$ are linear combinations of $\{\boldsymbol{\mu}_q[t]\}_{t\in[k]}$ and $\{L_{q,j}^t(\mathbf{x})\}_{t\in[k]}$ with weights $\mathbf{s} \in \mathbb{Z}_p^k$, which are computed using IPFE as these inner products:

$$\begin{aligned}\overline{\mu}_q &= \langle \boldsymbol{\mu}_q, \mathbf{s} \rangle,\\ \overline{L}_{q,j}(\mathbf{x}) &= \left\langle \left(\mathbf{L}_{q,j}^1, \ldots, \mathbf{L}_{q,j}^k\right), \left(\mathbf{s}[1](1,\mathbf{x}), \ldots, \mathbf{s}[k](1,\mathbf{x})\right) \right\rangle,\end{aligned}$$

  where the random multiples of the attribute $\left(\mathbf{s}[1](1,\mathbf{x}), \ldots, \mathbf{s}[k](1,\mathbf{x})\right)$ are encoded in the public slot of $\mathbf{u}$, $\boldsymbol{\mu}_q$ in the public slots of $\mathbf{v}_{q,\mathrm{pad}}$, and the coefficients $\left(\mathbf{L}_{q,j}^1, \ldots, \mathbf{L}_{q,j}^k\right)$ of the k label functions in the public slots of $\mathbf{v}_{q,j}$. In H_2^b, $\overline{\mu}_q$ and $\overline{L}_{q,j}(\mathbf{x})$ are instead computed as

$$\begin{aligned}\overline{\mu}_q &= \langle \overline{\mu}_q, 1 \rangle,\\ \overline{L}_{q,j}(\mathbf{x}) &= \langle \overline{\mathbf{L}}_{q,j}, (1,\mathbf{x}) \rangle, \quad \overline{\mathbf{L}}_{q,j} = \sum_{t\in[k]} \mathbf{s}[t]\mathbf{L}_{q,j}^t,\end{aligned}$$

  where $(1, \mathbf{x})$ is encoded in the *private* slot $\mathbf{u}$, $\overline{\mu}_q$ in the *private* slots of $\mathbf{v}_{q,\mathrm{pad}}$, and $\overline{\mathbf{L}}_{q,j}$ in the *private* slots of $\mathbf{v}_{q,j}$. Furthermore, to keep the inner products $\langle \mathbf{u}, \mathbf{v}_{q,\mathrm{pad}} \rangle$ and $\langle \mathbf{u}, \mathbf{v}_{q,j} \rangle$ the same as those in H_1^b, the random multiples of the attribute $\left(\mathbf{s}[1](1,\mathbf{x}), \ldots, \mathbf{s}[k](1,\mathbf{x})\right)$ in the *public* slot of $\mathbf{u}$ are replaced by zero (while the public slots of $\mathbf{v}_{q,\mathrm{pad}}, \mathbf{v}_{q,j}$ remain unchanged). By the function-hiding property of IPFE, H_2^b is indistinguishable from H_1^b.

hybrid	vector	pad	const^t	coef_i^t	const	coef_i	$\mathsf{sim}_1, \mathsf{sim}_\star$
$\mathsf{H}_0^b \equiv \mathsf{Exp}_{\mathrm{CPA}}^b$	sk_q: $\mathbf{v}_{q,\mathrm{pad}}$	1	$\boldsymbol{\mu}_q[t]$	0	0	0	0
	sk_q: $\mathbf{v}_{q,j}$	0	$\mathbf{L}_{q,j}^t[\mathsf{const}]$	$\mathbf{L}_{q,j}^t[\mathsf{coef}_i]$	0	0	0
	$\boxed{\mathsf{ct}}$: $\mathbf{u}$	h	$\mathbf{s}[t]$	$\mathbf{s}[t]\mathbf{x}[i]$	$\perp$	$\perp$	$\perp$
H_1^b	sk_q: $\mathbf{v}_{q,\mathrm{pad}}$	1	$\boldsymbol{\mu}_q[t]$	0	$\boxed{0}$	0	0
	sk_q: $\mathbf{v}_{q,j}$	0	$\mathbf{L}_{q,j}^t[\mathsf{const}]$	$\mathbf{L}_{q,j}^t[\mathsf{coef}_i]$	$\boxed{0}$	$\boxed{0}$	0
	$\boxed{\mathsf{ct}}$: $\mathbf{u}$	h	$\boxed{\mathbf{s}[t]}$	$\boxed{\mathbf{s}[t]\mathbf{x}[i]}$	$\boxed{0}$	$\boxed{0}$	0
H_2^b	sk_q: $\mathbf{v}_{q,\mathrm{pad}}$	1	$\boldsymbol{\mu}_q[t]$	0	$\boxed{\overline{\mu}_q}$	0	0
	sk_q: $\mathbf{v}_{q,j}$	0	$\mathbf{L}_{q,j}^t[\mathsf{const}]$	$\mathbf{L}_{q,j}^t[\mathsf{coef}_i]$	$\boxed{\overline{\mathbf{L}}_{q,j}[\mathsf{const}]}$	$\boxed{\overline{\mathbf{L}}_{q,j}[\mathsf{coef}_i]}$	0
	ct: $\mathbf{u}$	h	$\boxed{0}$	$\boxed{0}$	$\boxed{1}$	$\boxed{\mathbf{x}[i]}$	0
$\mathsf{H}_3^b \equiv \mathsf{H}_{4,1}^b$	sk_q: $\mathbf{v}_{q,\mathrm{pad}}$	1	$\boldsymbol{\mu}_q[t]$	0	$\boxed{\widehat{\mu}_q}$	0	0
	sk_q: $\mathbf{v}_{q,j}$	0	$\mathbf{L}_{q,j}^t[\mathsf{const}]$	$\mathbf{L}_{q,j}^t[\mathsf{coef}_i]$	$\boxed{\widehat{\mathbf{L}}_{q,j}[\mathsf{const}]}$	$\boxed{\widehat{\mathbf{L}}_{q,j}[\mathsf{coef}_i]}$	0
	ct: $\mathbf{u}$	h	0	0	1	$\mathbf{x}[i]$	0
$\mathsf{H}_{4,1\sim Q+1}^b$	.....						
$\mathsf{H}_5^b \equiv \mathsf{H}_{4,Q+1}^b$	sk_q: $\mathbf{v}_{q,\mathrm{pad}}$	1	$\boldsymbol{\mu}_q[t]$	0	$\boxed{\widehat{\mu}'_q \xleftarrow{\$} \mathbb{Z}_p}$	0	0
	sk_q: $\mathbf{v}_{q,j}$	0	$\mathbf{L}_{q,j}^t[\mathsf{const}]$	$\mathbf{L}_{q,j}^t[\mathsf{coef}_i]$	$\widehat{\mathbf{L}}_{q,j}[\mathsf{const}]$	$\widehat{\mathbf{L}}_{q,j}[\mathsf{coef}_i]$	0
	ct: $\mathbf{u}$	$\boxed{h}$	0	0	1	$\mathbf{x}[i]$	0
H_6^b	sk_q: $\mathbf{v}_{q,\mathrm{pad}}$	1	$\boldsymbol{\mu}_q[t]$	0	$\boxed{h + \widehat{\mu}'_q}$	0	0
	sk_q: $\mathbf{v}_{q,j}$	0	$\mathbf{L}_{q,j}^t[\mathsf{const}]$	$\mathbf{L}_{q,j}^t[\mathsf{coef}_i]$	$\widehat{\mathbf{L}}_{q,j}[\mathsf{const}]$	$\widehat{\mathbf{L}}_{q,j}[\mathsf{coef}_i]$	0
	ct: $\mathbf{u}$	$\boxed{0}$	0	0	1	$\mathbf{x}[i]$	0

The hybrids are illustrated with ciphertext challenge coming after the key queries (as in the more difficult case for 1-ABE). In reality, the key queries can be made before and after the ciphertext challenge, and the proof is unaffected.
In H_2^b, the linearly combined $\overline{\mathbf{L}}_{q,j} = \sum_{t\in[k]} \mathbf{s}[t]\mathbf{L}_{q,j}^t$ are valid garblings for the linearly combined pads $\overline{\mu}_q = \langle \mathbf{s}, \boldsymbol{\mu}_q \rangle$.
In H_3^b, the garblings $\widehat{\mathbf{L}}_{q,j}$ are fresh and are for freshly sampled pads $\widehat{\mu}_q$.
In $\mathsf{H}_5^b, \mathsf{H}_6^b$, the pads $\widehat{\mu}'_q$ in $\mathsf{sk}_{q,\mathrm{pad}}$ are independent of the pads $(\widehat{\mu}_q)$ being garbled.

Figure 4: The hybrids (with the loop contracted) in the proof of IND-CPA security of our KP-ABE scheme for ABPs.

- Hybrid H_3^b proceeds identically to H_2^b, except that the random linear combinations $\overline{\mu}_q, \overline{\mathbf{L}}_{q,j}$ in the private slots of the secret keys are replaced by randomly and independently generated pads and garblings $\widehat{\mu}_q, \widehat{\mathbf{L}}_{q,j}$, as shown in Figure 4.

  Recall that in both hybrids, the k instances of garblings in the public slot are generated by

$$\boldsymbol{\mu}_q \xleftarrow{\$} \mathbb{Z}_p^k, \quad \boldsymbol{\eta}_q \xleftarrow{\$} \mathbb{Z}_p^k, \quad \begin{cases} \boldsymbol{\alpha}_q \leftarrow \boldsymbol{\mu}_q, & \boldsymbol{\beta}_q \leftarrow \mathbf{0}, \qquad \text{if } y_q = f_{q,\neq 0}; \\ \boldsymbol{\alpha}_q \leftarrow \boldsymbol{\eta}_q, & \boldsymbol{\beta}_q \leftarrow \boldsymbol{\mu}_q, \qquad \text{if } y_q = f_{q,=0}; \end{cases}$$
$$\text{for } t \in [k]: \quad (\mathbf{L}_{q,1}^t, \ldots, \mathbf{L}_{q,m_q}^t) = \mathsf{Garble}(f_q, \boldsymbol{\alpha}_q[t], \boldsymbol{\beta}_q[t]; \mathbf{r}_{q,t}).$$

  In H_2^b, the private slots of ABE secret keys contain linear combinations of the garblings. By the linearity of AKGS, they can also be generated as follows:

$$\begin{cases} \overline{\alpha}_q = \langle \mathbf{s}, \boldsymbol{\alpha}_q \rangle = \overline{\mu}_q = \langle \mathbf{s}, \boldsymbol{\mu}_q \rangle, & \overline{\beta}_q = \langle \mathbf{s}, \boldsymbol{\beta}_q \rangle = \ 0 = \langle \mathbf{s}, \mathbf{0} \rangle, \quad \text{if } y_q = f_{q,\neq 0}; \\ \overline{\alpha}_q = \langle \mathbf{s}, \boldsymbol{\alpha}_q \rangle = \overline{\eta}_q = \langle \mathbf{s}, \boldsymbol{\eta}_q \rangle, & \overline{\beta}_q = \langle \mathbf{s}, \boldsymbol{\beta}_q \rangle = \overline{\mu}_q = \langle \mathbf{s}, \boldsymbol{\mu}_q \rangle, \quad \text{if } y_q = f_{q,=0}; \end{cases}$$
$$\overline{\mathbf{r}}_q = \sum_{t \in [k]} \mathbf{s}[t] \mathbf{r}_{q,t}, \qquad (\overline{\mathbf{L}}_{q,1}, \ldots, \overline{\mathbf{L}}_{q,m}) = \mathsf{Garble}(f_q, \overline{\alpha}_q, \overline{\beta}_q; \overline{\mathbf{r}}_q).$$

  In H_3^b, $\widehat{\mathbf{L}}_{q,j}$'s are generated for randomly and independently sampled pad $\widehat{\mu}_q$ with secrets $\widehat{\alpha}_q, \widehat{\beta}_q$ set according to $\widehat{\mu}_q$ and independent randomness $\widehat{\mathbf{r}}_q$:

$$\widehat{\mu}_q \xleftarrow{\$} \mathbb{Z}_p, \quad \widehat{\eta}_q \xleftarrow{\$} \mathbb{Z}_p, \quad \begin{cases} \widehat{\alpha}_q \leftarrow \widehat{\mu}_q, & \widehat{\beta}_q \leftarrow 0, \qquad \text{if } y = f_{q,\neq 0}; \\ \widehat{\alpha}_q \leftarrow \widehat{\eta}_q, & \widehat{\beta}_q \leftarrow \widehat{\mu}_q, \qquad \text{if } y = f_{q,=0}; \end{cases}$$
$$(\widehat{\mathbf{L}}_{q,1}, \ldots, \widehat{\mathbf{L}}_{q,m}) = \mathsf{Garble}(f_q, \widehat{\alpha}_q, \widehat{\beta}_q; \widehat{\mathbf{r}}_q).$$

  Note that the only difference between H_2^b and H_3^b is that the former uses linear combinations $\overline{\mu}_q, \overline{\eta}_q, \overline{\mathbf{r}}_q$ to set the the secrets and the randomness for the garbling in the private slot. In contrast, H_3^b uses fresh randomness. Note that in H_2^b, the weights $\mathbf{s}$ and $\boldsymbol{\mu}_q, \boldsymbol{\eta}_q, \mathbf{r}_{q,t}$ are only used in the exponent of the second source group G_2 (since they are only encoded in IPFE secret keys). Therefore, by the MDDH_k assumption, $\overline{\mu}_q, \overline{\eta}_q, \overline{\mathbf{r}}_q$ are pseudorandom in the exponent of G_2, i.e.,

$$\{[\![\underbrace{\boldsymbol{\mu}_q, \boldsymbol{\eta}_q, \{\mathbf{r}_{q,t}\}_{t\in[k]}}_{\mathbf{A}}, \underbrace{\overline{\mu}_q, \overline{\eta}_q, \overline{\mathbf{r}}_q}_{\mathbf{s}^\top \mathbf{A}}]\!]_2\}_{q\in[Q]} \approx \{[\![\boldsymbol{\mu}_q, \boldsymbol{\eta}_q, \{\mathbf{r}_{q,t}\}_{t\in[k]}, \underbrace{\widehat{\mu}_q, \widehat{\eta}_q, \widehat{\mathbf{r}}_q}_{\mathbf{c}^\top}]\!]_2\}_{q\in[Q]}.$$

  Furthermore, by the linearity of Garble, given these randomness encoded in G_2, the hybrids H_2^b and H_3^b can be efficiently generated. Thus, the two hybrids are indistinguishable.

We now have completed the first step of the proof. Observe that in H_3^b, for each $q \in [Q]$, the private slots of $\{\mathsf{isk}_{q,j}\}_{j\in[m_q]}$ encode (the coefficients of) label functions $\widehat{L}_{q,j}$ and the private slot of ict encodes $(1, \mathbf{x})$, identical to that in the secret key and ciphertext of our 1-ABE scheme. The security of 1-ABE (Theorem 25) ensures that the adversary cannot distinguish the pad $\widehat{\mu}_q$ encoded in the secret key from another independent pad $\widehat{\mu}'_q$ (inconsistent with $\widehat{L}_{q,j}$'s). Next, in hybrids $\mathsf{H}_{4,1}^b, \ldots, \mathsf{H}_{4,Q+1}^b$, we use syntactically the same proof to gradually switch each $\widehat{\mu}_q$ to $\widehat{\mu}'_q$.

- Hybrid $\mathsf{H}_{4,q}^b$ proceeds identically to H_3^b, except that in the first $q-1$ secret keys, the random pads $\widehat{\mu}_{q'}$ ($q' < q$) in the private slot of $\mathbf{v}_{q',\mathrm{pad}}$ are replaced by an independent random value $\widehat{\mu}'_{q'}$, as illustrated in Figure 5. The only difference between $\mathsf{H}_{4,q}^b$ and $\mathsf{H}_{4,q+1}^b$ is whether $\mathbf{v}_{q,\mathrm{pad}}$ contains the pad $\widehat{\mu}_q$ consistent with $\widehat{L}_{q,j}$'s or an independent one $\widehat{\mu}'_q$.

hybrid	vector	in $\mathfrak{s}_{\text{pub}}$	const	coef_i	$\mathsf{sim}_1, \mathsf{sim}_\star$
$\mathsf{H}^b_{4,q}$	$q' < q$ $\mathbf{v}_{q',\text{pad}}$	normal	$\widehat{\mu}'_{q'} \xleftarrow{\$} \mathbb{Z}_p$	0	0
	$q' < q$ $\mathbf{v}_{q',j}$		$\widehat{\mathbf{L}}_{q',j}[\mathsf{const}]$	$\widehat{\mathbf{L}}_{q',j}[\mathsf{coef}_i]$	0
	sk_q $\mathbf{v}_{q,\text{pad}}$		$\widehat{\mu}_q$	0	0
	sk_q $\mathbf{v}_{q,j}$		$\widehat{\mathbf{L}}_{q,j}[\mathsf{const}]$	$\widehat{\mathbf{L}}_{q,j}[\mathsf{coef}_i]$	0
	$q' > q$ $\mathbf{v}_{q',\text{pad}}$		$\widehat{\mu}_{q'}$	0	0
	$q' > q$ $\mathbf{v}_{q',j}$		$\widehat{\mathbf{L}}_{q',j}[\mathsf{const}]$	$\widehat{\mathbf{L}}_{q',j}[\mathsf{coef}_i]$	0
	ct: $\mathbf{u}$	$h, \mathbf{0}, \mathbf{0}$	1	$\mathbf{x}[i]$	0
$\mathsf{H}^b_{4,q+1}$	$q' < q$ $\mathbf{v}_{q',\text{pad}}$	normal	$\widehat{\mu}'_{q'} \xleftarrow{\$} \mathbb{Z}_p$	0	0
	$q' < q$ $\mathbf{v}_{q',j}$		$\widehat{\mathbf{L}}_{q',j}[\mathsf{const}]$	$\widehat{\mathbf{L}}_{q',j}[\mathsf{coef}_i]$	0
	sk_q $\mathbf{v}_{q,\text{pad}}$		$\boxed{\widehat{\mu}'_q \xleftarrow{\$} \mathbb{Z}_p}$	0	0
	sk_q $\mathbf{v}_{q,j}$		$\widehat{\mathbf{L}}_{q,j}[\mathsf{const}]$	$\widehat{\mathbf{L}}_{q,j}[\mathsf{coef}_i]$	0
	$q' > q$ $\mathbf{v}_{q',\text{pad}}$		$\widehat{\mu}_{q'}$	0	0
	$q' > q$ $\mathbf{v}_{q',j}$		$\widehat{\mathbf{L}}_{q',j}[\mathsf{const}]$	$\widehat{\mathbf{L}}_{q',j}[\mathsf{coef}_i]$	0
	ct: $\mathbf{u}$	$h, \mathbf{0}, \mathbf{0}$	1	$\mathbf{x}[i]$	0

The proof is unaffected by whether each sk appears before/after ct.
The "normal" in $\mathbf{v}|_{\mathfrak{s}_{\text{pub}}}$ represents the values specified by the honest KeyGen (the same as in Figure 4).
The garblings $\widehat{\mathbf{L}}_{q',j}$ are fresh and are for freshly sampled pads $\widehat{\mu}_{q'}$, and $\widehat{\mu}'_{q'}$ are independent of them.
In this iteration, sk_q is made *apparently* useless for decrypting ct.

Figure 5: The loop hybrids in the proof of IND-CPA security of our KP-ABE scheme for ABPs.

The indistinguishability of $\mathsf{H}^b_{4,q}$ and $\mathsf{H}^b_{4,q+1}$ follows by syntactically the same proof of 1-ABE security (Theorem 25). Recall that the proof of 1-ABE goes through a sequence of hybrids where the coefficients of the label functions $\widehat{L}_{q,j}$'s are gradually removed from $\mathbf{v}_{q,j}|_{\mathfrak{s}_{\text{1-ABE}}}$ and replaced by a simulated label hardcoded in either $\mathbf{v}_{q,j}[\mathsf{const}]$ (using $\mathbf{u}[\mathsf{sim}_\star]$ as a relay) or $\mathbf{u}[\mathsf{sim}_1]$. Simulated labels are sampled randomly $\ell_{q,j} \xleftarrow{\$} \mathbb{Z}_p$ for every $j > 1$, except that the first one is reversely sampled

$$\ell_{q,1} \xleftarrow{\$} \mathsf{RevSamp}(f_q, \mathbf{x}, \widehat{\alpha}_q f_q(\mathbf{x}) + \widehat{\beta}_q, \ell_{q,2}, \ldots, \ell_{q,m_q}).$$

If decryption should fail, $\widehat{\alpha}_q f_q(\mathbf{x}) + \widehat{\beta}_q$ perfectly hides $\widehat{\mu}_q$, and $\widehat{\mu}_q$ can be switched to $\widehat{\mu}'_q$. The indistinguishability of neighboring hybrids follows either from the AKGS piecewise security or the function-hiding property of IPFE.

The only differences between 1-ABE security proof and $\mathsf{H}^b_{4,q}$ / $\mathsf{H}^b_{4,q+1}$ are that in the latter the IPFE secret keys and ciphertext encode vectors with additional public slots, and that there are many (namely, Q) ABE secret keys. These differences do not affect the application of AKGS piecewise security nor the function-hiding property of IPFE. First, the piecewise security of AKGS is information-theoretic — given that $\widehat{L}_{q,j}$ is randomly generated and independent of the other variables in $\mathsf{H}^b_{4,q}$ and $\mathsf{H}^b_{4,q+1}$, it still applies. Second, the hybrids in 1-ABE security proof only modify the values in $\mathbf{v}_{q,j}|_{\mathfrak{s}_{\text{1-ABE}}}$ and $\mathbf{u}[\mathsf{sim}_1], \mathbf{u}[\mathsf{sim}_\star]$, all of which are in the private slot. Moreover, the vectors $\mathbf{v}_{q',j}, \mathbf{v}_{q',\text{pad}}$ in the additional secret keys (for $q' \neq q$) have values at $\mathsf{sim}_1, \mathsf{sim}_\star$ set to zero, and cannot "detect" the changes in $\mathbf{u}[\mathsf{sim}_1], \mathbf{u}[\mathsf{sim}_\star]$. Hence, all the inner products are kept the same and the function-hiding property of IPFE still applies. Therefore, $\mathsf{H}^b_{4,q}$ and $\mathsf{H}^b_{4,q+1}$ are indistinguishable.

We now have completed the second step of the proof. Observe that H^b_3 and $\mathsf{H}^b_{4,1}$ are identical. We need another invocation of the function-hiding property of IPFE to remove h from $\mathbf{u}$ (in ict):

- Hybrid H^b_5 proceeds identically to H^b_3, except that all the pads $\widehat{\mu}_q$ in $\mathbf{v}_{q,\text{pad}}$'s are replaced by independent random values $\widehat{\mu}'_q$, as illustrated in Figure 4. Observe that $\mathsf{H}^b_{4,Q+1} \equiv \mathsf{H}^b_5$, i.e., the loop hybrids connect H^b_3 and H^b_5.

  Note that in this hybrid, the inner products of $\mathbf{u}$ and $\mathbf{v}_{q,\text{pad}}$'s are $h + \widehat{\mu}'_q$, which hide h. However, h still appears in $\mathbf{u}$. In the next hybrid, we remove h from $\mathbf{u}$ and directly hardwire the inner products $h + \widehat{\mu}'_q$ in $\mathbf{v}_{q,\text{pad}}$'s so that h becomes information-theoretically hidden.

- Hybrid H^b_6 proceeds identically to H^b_5, except that the pads h and $\widehat{\mu}'_q$ are removed from $\mathbf{u}[\mathsf{pad}]$ and $\mathbf{v}_{q,\text{pad}}[\mathsf{const}]$ respectively, and that their sums $h + \widehat{\mu}'_q$ are put in $\mathbf{v}_{q,\text{pad}}[\mathsf{const}]$. The change is illustrated in Figure 4. Since the inner products remain the same as in H^b_5, by the function-hiding property of IPFE, H^b_5 and H^b_6 are indistinguishable.

Finally, we argue that $\mathsf{H}^0_6 \equiv \mathsf{H}^1_6$: The secret keys are independent of h since each $\widehat{\mu}_q$ perfectly hides h. The only other place where h appears is for hiding the message: $g_b + [\![h]\!]_{\mathrm{T}}$ in ct. This means g_b is information-theoretically hidden by h.

By a hybrid argument, we have $\mathsf{Exp}^0_{\text{CPA}} \approx \mathsf{Exp}^1_{\text{CPA}}$, i.e., Construction 26 is IND-CPA secure. □

Combining Theorems 22 and 27, we obtain compact and adaptively secure ABE for ABP:

Corollary 28. *Assuming* MDDH_k *in pairing groups, there is a* compact *and* adaptively secure *KP-ABE for ABP.*

7 ABE for Uniform Logspace Turing Machines

In this section, we construct a KP-ABE scheme for L, uniform deterministic log-space computation. Here, each secret key is associated with a Turing machine M (with an arbitrary number of states Q), and each ciphertext is associated with an input $\mathbf{x}$ (of arbitrary length N) and time/space complexity bounds $T, S \geq 1$. Decryption succeeds if and only if M accepts $\mathbf{x}$ within time T and space S. More formally, the scheme handles a single predicate P:

$$X = \left\{(\mathbf{x}, 1^T, 1^{2^S}) \,\middle|\, \mathbf{x} \in \{0,1\}^N \text{ for some } N \geq 1,\ T, S \geq 1\right\}, \quad Y = \mathsf{TM},$$
$$P : X \times Y \to \{0,1\},\ \left((\mathbf{x}, 1^T, 1^{2^S}), M\right) \mapsto M|_{N,T,S}(\mathbf{x}) \text{ for } \mathbf{x} \in \{0,1\}^N,$$

where $M|_{N,T,S}$ (Definition 12) indicates whether M accepts $\mathbf{x}$ within time T and space S. In our scheme, the size of a secret key is linear in Q and the size of a ciphertext is linear in $TNS2^S$.

Given our construction of ABE for L, we can derive ABE for DFA as a special case, since DFA can be viewed as a Turing machine with time complexity $T = N$ and space complexity $S = 1$, which always moves the input tape pointer to the right and never uses the work tape. Moreover, our ABE scheme for L extends immediately to *non-deterministic, unambiguous* logspace Turing machines (the complexity class UL). Such machines have at most one accepting path for any input. In fact, our construction handles all *non-deterministic* logspace Turing machines (the class NL), except when the number of accepting paths is exactly *a multiple of* p, the order of the pairing groups.

Below, we start with our AKGS for Turing machines, then build a 1-ABE scheme based on the AKGS and a function-hiding secret-key IPFE, and lastly lift it to a full-fledged KP-ABE scheme using slotted IPFE. For simplicity of exposition, we describe our construction based on the SXDH assumption in the pairing groups. It readily extends to be based on the MDDH_k assumption for any $k \geq 1$.

7.1 AKGS for Turing Machines with Time/Space Bounds

To obtain our AKGS for Turing machines with time/space bounds, we first represent the computation of Turing machines as a sequence of matrix multiplications, and then design an AKGS for matrix multiplication. See Section 2.3 for an overview of our AKGS.

Transition Matrix. Given a Turing machine $M = (Q, \mathbf{y}_{\mathrm{acc}}, \delta)$, upper bounds of time and space complexity $T, S \geq 1$, and an input $\mathbf{x} \in \{0,1\}^N$ for some $N \geq 1$, we consider the length-T *computation path* of M with input $\mathbf{x}$ and space bound S. Recall that the set of internal configurations is

$$\mathcal{C}_{M,N,S} = [N] \times [S] \times \{0,1\}^S \times [Q].$$

An internal configuration $z = (i, j, \mathbf{W}, q) \in \mathcal{C}_{M,N,S}$ specifies that the input and work tape pointers are at positions i and j respectively, the work tape has content $\mathbf{W}$, and the current state is q. In particular, the initial configuration is $(1, 1, \mathbf{0}_S, 1)$: the input/work tape pointers point to the first cell, the work tape is all-0, and the state is the initial state 1. An accepting configuration satisfies that $\mathbf{y}_{\mathrm{acc}}[q] = 1$.

We construct a transition matrix $\mathbf{M}_{N,S}(\mathbf{x}) \in \{0,1\}^{\mathcal{C}_{M,N,S} \times \mathcal{C}_{M,N,S}}$ such that $\mathbf{M}_{N,S}(\mathbf{x})[z, z']$ is 1 if and only if the internal configuration of M is z' after 1 step of computation starting from internal configuration z. According to how the Turing machine operates in each step depending on the

transition function δ, the entries of $\mathbf{M}_{N,S}$ are defined as follows:

$$\mathbf{M}_{N,S}(\mathbf{x})[(i,j,\mathbf{W},q),(i',j',\mathbf{W}',q')] = \begin{cases} 1, & \text{if } \delta(q,\mathbf{x}[i],\mathbf{W}[j]) = (q',\mathbf{W}'[j],i'-i,j'-j) \\ & \quad \text{and } \mathbf{W}'[j''] = \mathbf{W}[j''] \text{ for all } j'' \neq j; \\ 0, & \text{otherwise;} \end{cases}$$

$$= \quad \mathbf{x}[i] \times \begin{cases} 1, & \text{if } \delta(q,1,\mathbf{W}[j]) = (q',\mathbf{W}'[j],i'-i,j'-j) \\ & \quad \text{and } \mathbf{W}'[j''] = \mathbf{W}[j''] \text{ for all } j'' \neq j; \\ 0, & \text{otherwise;} \end{cases}$$

$$(1-\mathbf{x}[i]) \times \begin{cases} 1, & \text{if } \delta(q,0,\mathbf{W}[j]) = (q',\mathbf{W}'[j],i'-i,j'-j) \\ & \quad \text{and } \mathbf{W}'[j''] = \mathbf{W}[j''] \text{ for all } j'' \neq j; \\ 0, & \text{otherwise.} \end{cases}$$

With the transition matrix, we can now write the computation of Turing machine as a sequence of matrix multiplication. We represent internal configurations using one-hot encoding — the internal configuration z is represented by the basis vector $\mathbf{e}_z \in \{0,1\}^{\mathcal{C}_{M,N,S}}$ whose z-entry is 1 and other entries are 0. Observe that multiplying $\mathbf{e}_z^\intercal$ on the right by the transition matrix $\mathbf{M}_{N,S}(\mathbf{x})$ produces exactly the next internal configuration: If there is no valid internal configuration of M after 1 step of computation starting from z, we have $\mathbf{e}_z^\intercal \mathbf{M}_{N,S}(\mathbf{x}) = \mathbf{0}$; otherwise, the next internal configuration z' is unique and $\mathbf{e}_z^\intercal \mathbf{M}_{N,S}(\mathbf{x}) = \mathbf{e}_{z'}^\intercal$. The function $M|_{N,T,S}(\mathbf{x})$ can be written as

$$M|_{N,T,S}(\mathbf{x}) = \mathbf{e}_{(1,1,\mathbf{0}_S,1)}^\intercal \big(\mathbf{M}_{N,S}(\mathbf{x})\big)^T \big(\mathbf{1}_{[N]\times[S]\times\{0,1\}^S} \otimes \mathbf{y}_{\text{acc}}\big),$$

where $\mathbf{e}_{(1,1,\mathbf{0}_S,1)}$ represents the initial internal configuration. The sequence of multiplication advances the computation by T steps and tests whether the final internal configuration is in an accepting state. We elaborate on the last step: The tensor product $\mathbf{1}_{[N]\times[S]\times\{0,1\}^S} \otimes \mathbf{y}_{\text{acc}}$ is a vector in $\{0,1\}^{\mathcal{C}_{M,N,S}}$ such that its $(i,j,\mathbf{W},q)$-entry is 1 if and only if $\mathbf{y}_{\text{acc}}[q] = 1$, i.e., q is an accepting state. Therefore, taking the inner product of $\mathbf{e}_{(1,1,\mathbf{0}_S,1)}^\intercal \big(\mathbf{M}_{N,S}(\mathbf{x})\big)^T = \mathbf{e}_{z'}^\intercal$ (z' is the final internal configuration) or $\mathbf{0}$ with the tensor product indicates whether M accepts $\mathbf{x}$ within time T and space S.

Transition Blocks. We observe that the transition matrix has the following two useful properties:

- $\mathbf{M}_{N,S}(\mathbf{x})$ is affine in $\mathbf{x}$ when regarded as an integer matrix.
- $\mathbf{M}_{N,S}(\mathbf{x})$ has the following block structure: There is a finite set $\{\mathbf{M}_\tau\}_\tau$ of $Q \times Q$ matrices defined by the transition function δ, called the *transition blocks*, such that for every $(i,j,\mathbf{W})$ and $(i',j',\mathbf{W}')$ in $[N] \times [S] \times \{0,1\}^S$, the submatrix $\mathbf{M}_{N,S}(\mathbf{x})[(i,j,\mathbf{W},\text{␣}),(i',j',\mathbf{W}',\text{␣})]$ is either some $\mathbf{M}_\tau$ or $\mathbf{0}$.

We define the transition blocks below.

Definition 29. *Let $M = (Q, \mathbf{y}_{\text{acc}}, \delta)$ be a Turing machine and $\mathcal{T} = \{0,1\}^3 \times \{0,\pm 1\}^2$ the set of* transition types*. The* transition blocks *of M consist of 72 matrices $\mathbf{M}_\tau \in \{0,1\}^{Q\times Q}$ for $\tau = (x,w,w',\Delta i,\Delta j) \in \mathcal{T}$, each encoding the possible transitions among the states given the following information: the input tape symbol x under scan, the work tape symbol w under scan, the symbol w' overwriting w, the direction Δi to which the input tape pointer moves, and the direction Δj to which the work tape pointer moves. Formally,*

$$\mathbf{M}_{x,w,w',\Delta i,\Delta j}[q,q'] = \begin{cases} 1, & \text{if } \delta(q,x,w) = (q',w',\Delta i,\Delta j); \\ 0, & \text{otherwise.} \end{cases}$$

In $\mathbf{M}_{N,S}(\mathbf{x})$, each $Q \times Q$ block is either one of the transition blocks or $\mathbf{0}$:

$$\mathbf{M}_{N,S}(\mathbf{x})[(i,j,\mathbf{W},\sqcup),(i',j',\mathbf{W}',\sqcup)] = \begin{cases} \mathbf{M}_{\mathbf{x}[i],\mathbf{W}[j],\mathbf{W}'[j],i'-i,j'-j}, & \text{if } i'-i, j'-j \in \{0,\pm 1\} \text{ and} \\ & \mathbf{W}[j''] = \mathbf{W}'[j''] \text{ for all } j'' \neq j; \\ \mathbf{0}, & \text{otherwise.} \end{cases}$$

Observe further that in $\mathbf{M}_{N,S}(\mathbf{x})[(i,j,\mathbf{W},\sqcup),(\sqcup,\sqcup,\sqcup,\sqcup)]$, each transition block appears at most once.

AKGS for Turing machines. Above, we have represented the Turing machine computation as a sequence of matrix multiplication *over the integers*:

$$M|_{N,T,S}(\mathbf{x}) = \mathbf{e}_{(1,1,\mathbf{0}_S,1)}^{\mathsf{T}} \big(\mathbf{M}_{N,S}(\mathbf{x})\big)^T \big(\mathbf{1}_{[N]\times[S]\times\{0,1\}^S} \otimes \mathbf{y}_{\mathrm{acc}}\big) \quad \text{for } \mathbf{x} \in \{0,1\}^N. \tag{7}$$

We can formally extend[13] $M|_{N,T,S} : \{0,1\}^N \to \{0,1\}$ to a $\mathbb{Z}_p^N \to \mathbb{Z}_p$ function using the same matrix multiplication formula, preserving its behavior when the input comes from $\{0,1\}^N$. When p is clear from the context, we use $M|_{N,T,S}$ to represent its extension over $\mathbb{Z}_p$.

We now construct an AKGS for Turing machine computations and prove its special piecewise security. It is constructed via a recursive mechanism for garbling matrix multiplications, which is inspired[14] by the garbling scheme for arithmetic NC^1 circuits in [AIK11] and the garbling mechanism for multiplication gates in [BGG+14].

Construction 30 (AKGS for $M|_{N,T,S}$)**.** Let

$$\mathcal{F} = \big\{ M|_{N,T,S} : \mathbb{Z}_p^N \to \mathbb{Z}_p \,\big|\, M \in \mathsf{TM},\ N,T,S \geq 1,\ p \text{ prime} \big\}$$

be the set of time/space bounded Turing machine computations. The AKGS $(\mathsf{Garble}, \mathsf{Eval})$ for the function class $\mathcal{F}$ operates as follows:

- $\mathsf{Garble}\big((M, 1^N, 1^T, 1^{2^S}, p), \alpha, \beta\big)$ takes a function $M|_{N,T,S}$ over $\mathbb{Z}_p$ from $\mathcal{F}$ and two secrets $\alpha, \beta \in \mathbb{Z}_p$ as input. Suppose $M = (Q, \mathbf{y}_{\mathrm{acc}}, \delta)$, the algorithm samples $\mathbf{r}$ as the randomness by

$$\text{for } t \in [0..T]\text{:} \quad \mathbf{r}_t \xleftarrow{\$} \mathbb{Z}_p^{\mathcal{C}_{M,N,S}} \quad (\mathcal{C}_{M,N,S} = [N]\times[S]\times\{0,1\}^S\times[Q]),$$
$$\mathbf{r} \in \mathbb{Z}_p^{[0..T]\times\mathcal{C}_{M,N,S}}, \quad \mathbf{r}[(t,i,j,\mathbf{W},q)] = \mathbf{r}_t[(i,j,\mathbf{W},q)].$$

  It computes the transition matrix $\mathbf{M}_{N,S}(\mathbf{x})$ as a function of $\mathbf{x}$ and defines the label functions by

$$L_{\mathrm{init}}(\mathbf{x}) = \beta + \mathbf{e}_{(1,1,\mathbf{0}_S,1)}^{\mathsf{T}} \mathbf{r}_0,$$
$$\text{for } t \in [T]\text{:} \quad \big(L_{t,z}(\mathbf{x})\big)_{z\in\mathcal{C}_{M,N,S}} = -\mathbf{r}_{t-1} + \mathbf{M}_{N,S}(\mathbf{x})\mathbf{r}_t,$$
$$\big(L_{T+1,z}(\mathbf{x})\big)_{z\in\mathcal{C}_{M,N,S}} = -\mathbf{r}_T + \alpha \mathbf{1}_{[N]\times[S]\times\{0,1\}^S} \otimes \mathbf{y}_{\mathrm{acc}}.$$

  It collects the coefficients of these label functions and returns them.

  Note: *We show* Garble *satisfies the required properties of a linear AKGS (Definitions 13 and 14):*

[13] This is done to arithmetize the computation, adhering to the formalism of AKGS.

[14] As a historical fact, the authors first came up with the AKGS for arithmetic formulae, abstracted out piecewise security, and only verified that the PGS for ABPs [IW14] is also a piecewise secure AKGS.

– *The label functions are affine in* $\mathbf{x}$*:* L_{init} *and* $L_{T+1,z}$ *for all* $z \in \mathcal{C}_{M,N,S}$ *are constant with respect to* $\mathbf{x}$*. The rest are* $L_{t,z}(x) = \big(-\mathbf{r}_{t-1} + \mathbf{M}_{N,S}(\mathbf{x})\mathbf{r}_t\big)[z]$*. Since* $\mathbf{M}_{N,S}(\mathbf{x})$ *is affine in* $\mathbf{x}$ *and* $\mathbf{r}_{t-1}, \mathbf{r}_t$ *are constant with respect to* $\mathbf{x}$*, these label functions are also affine in* $\mathbf{x}$*.*

– *Shape determinism holds: The garbling size of* $M|_{N,T,S}$ *is* $1 + (T+1)NS2^S Q$*.*

– Garble *is linear in* $(\alpha, \beta, \mathbf{r})$*, i.e., the coefficients of the label functions are linear in* $(\alpha, \beta, \mathbf{r})$*. Observe that* $\mathbf{M}_{N,S}(\mathbf{x}), \mathbf{e}_{(1,1,\mathbf{0}_S,1)}$ *and* $\mathbf{y}_{\text{acc}}$ *are constant with respect to* $(\alpha, \beta, \mathbf{r})$*, and* α, β *and* $\mathbf{r}_t$ *for all* $t \in [0..T]$ *are linear in* $(\alpha, \beta, \mathbf{r})$*. By the definition of the label functions, their coefficients are linear in* $(\alpha, \beta, \mathbf{r})$*.*

- Eval$\big((M, 1^N, 1^T, 1^{2^S}, p), \mathbf{x}, \ell_{\text{init}}, (\ell_{t,z})_{t\in[T+1], z\in\mathcal{C}_{M,N,S}}\big)$ takes a function $M|_{N,T,S}$ over $\mathbb{Z}_p$ from $\mathcal{F}$, an input string $\mathbf{x} \in \mathbb{Z}_p^N$ and the labels as input. It first computes the transition matrix $\mathbf{M}_{N,S}(\mathbf{x})$ with $\mathbf{x}$ substituted into it and sets $\boldsymbol{\ell}_t = (\ell_{t,z})_{z\in\mathcal{C}_{M,N,S}}$ for $t \in [T+1]$. The algorithm computes and returns

$$\ell_{\text{init}} + \mathbf{e}^{\mathsf{T}}_{(1,1,\mathbf{0}_S,1)} \sum_{t=1}^{T+1} \big(\mathbf{M}_{N,S}(\mathbf{x})\big)^{t-1} \boldsymbol{\ell}_t.$$

Note: *We show the correctness of the scheme. Plugging* $\ell_{t,z} = L_{t,z}(\mathbf{x})$ *and the formula for* $M|_{N,T,S}$ *into the summation, we find that it is a telescoping sum:*

$$\begin{aligned}
\mathbf{e}^{\mathsf{T}}_{(1,1,\mathbf{0}_S,1)} \sum_{t=1}^{T+1} \big(\mathbf{M}_{N,S}(\mathbf{x})\big)^{t-1} \boldsymbol{\ell}_t &= \mathbf{e}^{\mathsf{T}}_{(1,1,\mathbf{0}_S,1)} \sum_{t=1}^{T} \big(\mathbf{M}_{N,S}(\mathbf{x})\big)^{t-1} \big(-\mathbf{r}_{t-1} + \mathbf{M}_{N,S}(\mathbf{x})\mathbf{r}_t\big) \\
&\quad + \mathbf{e}^{\mathsf{T}}_{(1,1,\mathbf{0}_S,1)} \big(\mathbf{M}_{N,S}(\mathbf{x})\big)^{T} \big(-\mathbf{r}_T + \alpha \mathbf{1}_{[N]\times[S]\times\{0,1\}^S} \otimes \mathbf{y}_{\text{acc}}\big) \\
&= \mathbf{e}^{\mathsf{T}}_{(1,1,\mathbf{0}_S,1)} \sum_{t=1}^{T} \Big(-\big(\mathbf{M}_{N,S}(\mathbf{x})\big)^{t-1} \mathbf{r}_{t-1} + \big(\mathbf{M}_{N,S}(\mathbf{x})\big)^{t} \mathbf{r}_t\Big) \\
&\quad - \mathbf{e}^{\mathsf{T}}_{(1,1,\mathbf{0}_S,1)} \big(\mathbf{M}_{N,S}(\mathbf{x})\big)^{T} \mathbf{r}_T + \alpha M|_{N,T,S}(\mathbf{x}) \\
&= -\mathbf{e}^{\mathsf{T}}_{(1,1,\mathbf{0}_S,1)} \mathbf{r}_0 + \alpha M|_{N,T,S}(\mathbf{x}).
\end{aligned}$$

The value returned by Eval *is*

$$\begin{aligned}
\ell_{\text{init}} + \mathbf{e}^{\mathsf{T}}_{(1,1,\mathbf{0}_S,1)} \sum_{t=1}^{T+1} \big(\mathbf{M}_{N,S}(\mathbf{x})\big)^{t-1} \boldsymbol{\ell}_t &= \big(\beta + \mathbf{e}^{\mathsf{T}}_{(1,1,\mathbf{0}_S,1)} \mathbf{r}_0\big) + \big(-\mathbf{e}^{\mathsf{T}}_{(1,1,\mathbf{0}_S,1)} \mathbf{r}_0 + \alpha M|_{N,T,S}(\mathbf{x})\big) \\
&= \beta + \alpha M|_{N,T,S}(\mathbf{x}).
\end{aligned}$$

Therefore, the scheme is correct. Moreover, Eval *is linear in the labels, as seen from the formula of* Eval*.*

Theorem 31. *Construction 30 is special piecewise secure. More specifically, the label functions are ordered as* $L_{\text{init}}, (L_{1,z})_{z\in\mathcal{C}_{M,N,S}}, (L_{2,z})_{z\in\mathcal{C}_{M,N,S}}, \ldots, (L_{T+1,z})_{z\in\mathcal{C}_{M,N,S}}$*, the randomness is ordered as* $\mathbf{r}_0, \mathbf{r}_1, \ldots, \mathbf{r}_T$*, and the randomizer of* $L_{t,z}$ *is* $\mathbf{r}_{t-1}[z]$*. For each* $t \in [T+1]$*, the ordering of the components in* $(L_{t,z})_{z\in\mathcal{C}_{M,N,S}}$ *and* $\mathbf{r}_{t-1}$ *can be arbitrary, as long as the two are consistent.*

Proof. By definition, the coefficient of ℓ_{init} in Eval is $1 \neq 0$. Now we prove the marginal randomness property. Recall that all the other label functions are grouped by time step $t \in [T+1]$:

$$\begin{aligned}
\text{for } t \in [T]: \quad \big(L_{t,z}(\mathbf{x})\big)_{z\in\mathcal{C}_{M,N,S}} &= -\mathbf{r}_{t-1} + \mathbf{M}_{N,S}(\mathbf{x})\mathbf{r}_t, \\
\big(L_{T+1,z}(\mathbf{x})\big)_{z\in\mathcal{C}_{M,N,S}} &= -\mathbf{r}_T + \alpha \mathbf{1}_{[N]\times[S]\times\{0,1\}^S} \otimes \mathbf{y}_{\text{acc}}.
\end{aligned}$$

Fix a group $t \in [T+1]$ of the label functions. Each $L_{t,z}$ (for $z \in \mathcal{C}_{M,N,S}$), aside from $\mathbf{r}_{t-1}[z]$ (which is in the constant), only uses components in $\mathbf{r}_t$ (if $t \le T$) as randomness. Moreover, all the label functions in groups $t+1, t+2, \ldots, T+1$ only use components in $\mathbf{r}_t, \mathbf{r}_{t+1}, \ldots, \mathbf{r}_T$ as randomness. Therefore, all those label functions are in the special form, the randomizer of $L_{t,z}$ is $\mathbf{r}_{t-1}[z]$, and the components in each group of $L_{t,z}$'s and $\mathbf{r}_{t-1}$ can be reordered as long as they are consistent. □

7.2 1-ABE for L

With the AKGS for Turing machines, we are ready to construct our 1-ABE for L and prove its security. To focus on the core idea, we present our construction assuming SXDH in the pairing groups and briefly discuss how to generalize the construction to be based on MDDH_k for any $k \ge 1$.

New Challenge. As discussed in the technical overview (see Section 2.3), building ABE for L from AKGS faces new challenges comparing with ABE for ABPs. In particular, the size of the AKGS garbling is roughly $TNS2^SQ$. While the key generation algorithm knows Q and the encryption algorithms knows N, T, S, neither of them alone has full information of the size of the AKGS garbling. Furthermore, the total size of a pair of secret key and ciphertext is way smaller than the garbling size. Therefore, it is impossible to encode the label functions of AKGS in either the secret key or the ciphertext (in contrast, in ABE for ABPs, the label functions are completely encoded in the secret key). To overcome this challenge, we will *i*) let the secret key and ciphertext jointly generate the label functions, and *ii*) use pseudorandomness in the exponent.

An Exercise of Algebra. Recall that the AKGS for L samples randomness $\mathbf{r} \in \mathbb{Z}_p^{[0..T]\times\mathcal{C}_{M,N,S}}$. We will use "structured" elements $\mathbf{r} = \mathbf{r}_{\mathrm{x}} \otimes \mathbf{r}_{\mathrm{f}}$ for $\mathbf{r}_{\mathrm{x}} \xleftarrow{\$} \mathbb{Z}_p^{[0..T]\times[N]\times[S]\times\{0,1\}^S}$ and $\mathbf{r}_{\mathrm{f}} \xleftarrow{\$} \mathbb{Z}_p^Q$ as *the* randomness for the AKGS garbling. Before laying out the construction, we first show that $\mathbf{M}_{N,S}(\mathbf{x})\mathbf{r}_t$ (a central part of the label functions) can be expressed as a *bilinear* function of $\mathbf{x}, \mathbf{r}_{\mathrm{x}}, \mathbf{x}\otimes\mathbf{r}_{\mathrm{x}}$ (known at encryption time) and $\mathbf{M}_\tau\mathbf{r}_{\mathrm{f}}, \mathbf{r}_{\mathrm{f}}$'s (known at key generation time), and hence can be computed as the inner products of vectors depending on these two groups of variables separately.

By our choice of randomness, $\mathbf{r}_t = \mathbf{r}[t, \sqcup, \sqcup, \sqcup, \sqcup]$ is a block vector with each block being a multiple of $\mathbf{r}_{\mathrm{f}}$. More precisely, $\mathbf{r}_t[i, j, \mathbf{W}, \sqcup] = \mathbf{r}_{\mathrm{x}}[(t, i, j, \mathbf{W})]\mathbf{r}_{\mathrm{f}}$. We compute each block of the product $\mathbf{M}_{N,S}(\mathbf{x})\mathbf{r}_t$:

$$
\begin{aligned}
& \big(\mathbf{M}_{N,S}(\mathbf{x})\mathbf{r}_t\big)[(i, j, \mathbf{W}, \sqcup)] \\
\left(\begin{smallmatrix}\text{row } r \text{ of } AB \text{ is}\\ \text{row } r \text{ of } A \text{ times } B\end{smallmatrix}\right) \quad & = \mathbf{M}_{N,S}(\mathbf{x})[(i, j, \mathbf{W}, \sqcup), (\sqcup, \sqcup, \sqcup, \sqcup)]\mathbf{r}_t \\
\left(\begin{smallmatrix}\text{block matrix}\\ \text{multiplication}\end{smallmatrix}\right) \quad & = \sum_{\substack{i'\in[N], j'\in[S]\\ \mathbf{W}'\in\{0,1\}^S}} \mathbf{M}_{N,S}(\mathbf{x})[(i, j, \mathbf{W}, \sqcup), (i', j', \mathbf{W}', \sqcup)]\mathbf{r}_t[(i', j', \mathbf{W}', \sqcup)] \\
& = \sum_{\substack{i'\in[N], j'\in[S]\\ \mathbf{W}'\in\{0,1\}^S}} \mathbf{M}_{N,S}(\mathbf{x})[(i, j, \mathbf{W}, \sqcup), (i', j', \mathbf{W}', \sqcup)]\mathbf{r}_{\mathrm{x}}[(t, i', j', \mathbf{W}')]\mathbf{r}_{\mathrm{f}}.
\end{aligned}
$$

Recall that in $\mathbf{M}_{N,S}(\mathbf{x})[(i, j, \mathbf{W}, \sqcup), (\sqcup, \sqcup, \sqcup, \sqcup)]$, each transition block appears at most once, and the other $Q\times Q$ blocks are $\mathbf{0}$. More specifically, $\mathbf{M}_{x,w,w',\Delta i,\Delta j}$ appears at $\mathbf{M}_{N,S}(\mathbf{x})[(i, j, \mathbf{W}, \sqcup), (i', j', \mathbf{W}', \sqcup)]$ if $x = \mathbf{x}[i]$, $w = \mathbf{W}[j]$, $\Delta i = i' - i$, $\Delta j = j' - j$, and $\mathbf{W}'$ is $\mathbf{W}$ with its j^{th} entry changed to w'.

Therefore, we have

$$\big(\mathbf{M}_{N,S}(\mathbf{x})\mathbf{r}_t\big)[(i,j,\mathbf{W},\sqcup)] = \sum_{\substack{w'\in\{0,1\}\\ \Delta i,\Delta j\in\{0,\pm1\}\\ i+\Delta j\in[N],j+\Delta j\in[S]}} \mathbf{M}_{\mathbf{x}[i],\mathbf{W}[j],w',\Delta i,\Delta j}\mathbf{r}_{\mathrm{x}}[(t,i+\Delta i,j+\Delta j,\mathbf{W}')]\mathbf{r}_{\mathrm{f}}$$

$$= \sum_{\substack{x,w,w'\in\{0,1\}\\ \Delta i,\Delta j\in\{0,\pm1\}}} \mathbf{M}_{x,w,w',\Delta i,\Delta j}\mathbf{r}_{\mathrm{f}} \times \begin{cases} \mathbf{r}_{\mathrm{x}}[(t,i+\Delta i,j+\Delta j,\mathbf{W}')], & \text{if } x=\mathbf{x}[i],\ i+\Delta i\in[N],\\ & \qquad w=\mathbf{W}[j],\ j+\Delta j\in[S];\\ 0, & \text{otherwise.}\end{cases} \tag{8}$$

Here, $\mathbf{W}'[j]=w'$ and $\mathbf{W}'[j'']=\mathbf{W}[j'']$ for all $j''\neq j$. Note that in the last summation formula, there are exactly 72 summands. Moreover, each summand is $\mathbf{M}_{x,w,w',\Delta i,\Delta j}\mathbf{r}_{\mathrm{f}}$ (depending only on $\mathbf{r}_{\mathrm{f}}$ and the transition blocks) multiplied by an entry in $\mathbf{r}_{\mathrm{x}}$ or 0 (depending only on $\mathbf{x},\mathbf{r}_{\mathrm{x}}$). To simplify notations, we define *transition coefficients*:

Definition 32. *Let* $\mathcal{T}=\{0,1\}^3\times\{0,\pm1\}^2$ *be the set of* transition types. *For all* $\tau=(x,w,w',\Delta i,\Delta j)\in\mathcal{T}$, $N,T,S\geq 1$, *and* $\mathbf{x}\in\{0,1\}^N, t\in[T], i\in[N], j\in[S], \mathbf{W}\in\{0,1\}^S, \mathbf{r}_{\mathrm{x}}\in\mathbb{Z}_p^{[0..T]\times[N]\times[S]\times\{0,1\}^S}$, *define the* transition coefficient *as*

$$\mathfrak{c}_{x,w,w',\Delta i,\Delta j}(\mathbf{x};t,i,j,\mathbf{W};\mathbf{r}_{\mathrm{x}}) = \begin{cases} \mathbf{r}_{\mathrm{x}}[(t,i+\Delta i,j+\Delta j,\mathbf{W}')], & \text{if } x=\mathbf{x}[i],\ i+\Delta i\in[N],\\ & \qquad w=\mathbf{W}[j],\ j+\Delta j\in[S];\\ 0, & \text{otherwise;}\end{cases}$$

where $\mathbf{W}'\in\{0,1\}^S$, $\mathbf{W}'[j]=w'$, *and* $\mathbf{W}'[j'']=\mathbf{W}[j'']$ *for all* $j''\neq j$.

With the above definition, Equation (8) can be restated as

$$\big(\mathbf{M}_{N,S}(\mathbf{x})\mathbf{r}_t\big)[(i,j,\mathbf{W},\sqcup)] = \sum_{\tau\in\mathcal{T}} \mathfrak{c}_\tau(\mathbf{x};t,i,j,\mathbf{W};\mathbf{r}_{\mathrm{x}})\mathbf{M}_\tau\mathbf{r}_{\mathrm{f}}. \tag{9}$$

We are now ready to construct 1-ABE for L.

Construction 33 (1-ABE for L)**.** We describe the construction for any fixed value of the security parameter λ and suppress the appearance of λ below for simplicity of notations. Let the function class $\mathcal{F}$ be

$$\mathcal{F}=\big\{M|_{N,T,S}:\mathbb{Z}_p^N\to\mathbb{Z}_p \,\big|\, M\in\mathsf{TM}, N,T,S\geq 1, p \text{ prime}\big\}.$$

Let $(\mathsf{Garble},\mathsf{Eval})$ be the AKGS for $\mathcal{F}$ in Construction 30, $\mathcal{G}$ pairing groups of order p, and $(\mathsf{IPFE.Setup},\mathsf{IPFE.KeyGen},\mathsf{IPFE.Enc},\mathsf{IPFE.Dec})$ a secret-key IPFE based on $\mathcal{G}$. We construct a 1-ABE scheme based on $\mathcal{G}$ for the following singleton predicate space $\mathcal{P}$:

$$X=\big\{(\mathbf{x},1^T,1^{2^S}) \,\big|\, \mathbf{x}\in\{0,1\}^N \text{ for some } N\geq 1,\ T,S\geq 1\big\},\quad Y=\mathsf{TM},$$
$$P:X\times Y\to\{0,1\},\ \big((\mathbf{x},1^T,1^{2^S}),M\big)\mapsto M|_{N,T,S}(\mathbf{x}) \text{ for } \mathbf{x}\in\{0,1\}^N,$$
$$\mathcal{P}=\{P\}.$$

The 1-ABE scheme $(\mathsf{Setup},\mathsf{KeyGen},\mathsf{Enc},\mathsf{Dec})$ operates as follows:

- $\mathsf{Setup}(P)$ takes the only predicate as input. It generates an IPFE master secret key $\mathsf{msk}\xleftarrow{\$}\mathsf{IPFE.Setup}(\mathfrak{s}_{\text{1-ABE}})$ for the index set

$$\mathfrak{s}_{\text{1-ABE}}=\big\{\mathsf{init},\ \mathsf{rand},\mathsf{rand}^{\mathsf{comp}},\mathsf{rand}^{\mathsf{temp}},\mathsf{rand}^{\mathsf{temp,comp}},\ \mathsf{acc},\mathsf{acc}^{\mathsf{temp}},\ \mathsf{sim},\mathsf{sim}^{\mathsf{temp}},\mathsf{sim}^{\mathsf{comp}}\big\}$$
$$\cup\big\{\mathsf{tb}_\tau,\mathsf{tb}_\tau^{\mathsf{comp}},\mathsf{tb}_\tau^{\mathsf{temp}},\mathsf{tb}_\tau^{\mathsf{temp,comp}} \,\big|\, \tau\in\mathcal{T}\big\}.$$

The algorithm returns msk as the master secret key.

Note: *Let us explain the names of the indices: The index* init *is used for computing* ℓ_{init}*. It is also used in the security proof. The indices* tb_τ *(short for "transition blocks") are used for computing* $\mathbf{M}_{N,S}(\mathbf{x})\mathbf{r}_t$ *in* $\ell_{t,z}$*'s,* rand *(short for "randomizer") for* $-\mathbf{r}_{t-1}$*, and* acc *(for "acceptance vector") for* $\alpha\mathbf{1}_{[N]\times[S]\times\{0,1\}^S} \otimes \mathbf{y}_{\mathrm{acc}}$ *in* $\ell_{T+1,z}$*'s. Values at all the other indices are set to zero by the honest algorithms, and those indices are only used in the security proof. Jumping ahead,* sim *is used for simulating the labels, and the rest of the indices are used for holding temporary values in the hybrids;* temp *(resp.* comp*) is short for "temporary" (resp. "compensation"). More details in the proof.*

- $\mathsf{KeyGen}(\mathsf{msk}, M, \mu)$ takes the master secret key, a Turing machine $M = (Q, \mathbf{y}_{\mathrm{acc}}, \delta) \in \mathsf{TM} = Y$, and a pad $\mu \in \mathbb{Z}_p$ as input. It computes the transition blocks $\mathbf{M}_\tau$ for $\tau \in \mathcal{T}$ from δ, samples $\mathbf{r}_{\mathrm{f}} \xleftarrow{\$} \mathbb{Z}_p^Q$, and sets the vectors $\mathbf{v}_{\mathrm{init}}$ and $\mathbf{v}_q$ for $q \in [Q]$ as follows:

vector	init	rand	acc	tb_τ	the other indices
$\mathbf{v}_{\mathrm{init}}$	$\mathbf{r}_{\mathrm{f}}[1]$	0	0	0	0
$\mathbf{v}_q$	0	$-\mathbf{r}_{\mathrm{f}}[q]$	$\mu\mathbf{y}_{\mathrm{acc}}[q]$	$(\mathbf{M}_\tau\mathbf{r}_{\mathrm{f}})[q]$	

The algorithm generates an IPFE secret key for each vector defined above:

$$\mathsf{isk}_{\mathrm{init}} \xleftarrow{\$} \mathsf{IPFE.KeyGen}(\mathsf{msk}, [\![\mathbf{v}_{\mathrm{init}}]\!]_2),$$
$$\text{for } q \in [Q]\text{:}\quad \mathsf{isk}_q \xleftarrow{\$} \mathsf{IPFE.KeyGen}(\mathsf{msk}, [\![\mathbf{v}_q]\!]_2).$$

It returns $\mathsf{sk}_M = (M, \mathsf{isk}_{\mathrm{init}}, \mathsf{isk}_1, \ldots, \mathsf{isk}_Q)$ as the secret key for M encapsulating μ.

Note: KeyGen *creates the garbling (or rather, half of it) with secrets* $\alpha = \mu, \beta = 0$ *so that* μ *can be recovered using* Eval *if and only if* $M|_{N,T,S}(\mathbf{x}) \neq 0$*.*

- $\mathsf{Enc}\big(\mathsf{msk}, (\mathbf{x}, 1^T, 1^{2^S})\big)$ takes the master secret key, an input string $\mathbf{x} \in \{0,1\}^N$ for some $N \geq 1$, and time/space complexity bounds $T, S \geq 1$ (encoded as 1^T and 1^{2^S}) as input. It samples $\mathbf{r}_{\mathrm{x}} \xleftarrow{\$} \mathbb{Z}_p^{[0..T]\times[N]\times[S]\times\{0,1\}^S}$ and sets the vectors $\mathbf{u}_{\mathrm{init}}$ and $\mathbf{u}_{t,i,j,\mathbf{W}}$ for $t \in [T+1], i \in [N], j \in [S]$, $\mathbf{W} \in \{0,1\}^S$ as follows:

vector	init	rand	acc	tb_τ	the other indices
$\mathbf{u}_{\mathrm{init}}$	$\mathbf{r}_{\mathrm{x}}[(0,1,1,\mathbf{0}_S)]$	0	0	0	
$t \leq T$: $\mathbf{u}_{t,i,j,\mathbf{W}}$	0	$\mathbf{r}_{\mathrm{x}}[(t-1,i,j,\mathbf{W})]$	0	$\mathfrak{c}_\tau(\mathbf{x};t,i,j,\mathbf{W};\mathbf{r}_{\mathrm{x}})$	0
$\mathbf{u}_{T+1,i,j,\mathbf{W}}$	0	$\mathbf{r}_{\mathrm{x}}[(T,i,j,\mathbf{W})]$	1	0	

The algorithm then generates IPFE ciphertexts for the vectors defined above:

$$\mathsf{ict}_{\mathrm{init}} \xleftarrow{\$} \mathsf{IPFE.Enc}(\mathsf{msk}, [\![\mathbf{u}_{\mathrm{init}}]\!]_1),$$
$$\text{for } t \in [T+1], i \in [N], j \in [S], \mathbf{W} \in \{0,1\}^S\text{:}\quad \mathsf{ict}_{t,i,j,\mathbf{W}} \xleftarrow{\$} \mathsf{IPFE.Enc}(\mathsf{msk}, [\![\mathbf{u}_{t,i,j,\mathbf{W}}]\!]_1).$$

It returns $\mathsf{ct}_{\mathbf{x},T,S} = \big((\mathbf{x}, T, S), \mathsf{ict}_{\mathrm{init}}, (\mathsf{ict}_{t,i,j,\mathbf{W}})_{t\in[T+1],i\in[N],j\in[S],\mathbf{W}\in\{0,1\}^S}\big)$ as the ciphertext for input $\mathbf{x}$ with time/space bounds T, S.

- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$ takes a secret key sk and a ciphertext ct as input. It parses

$$\mathsf{sk} \text{ as } \big(M, \mathsf{isk}_{\mathrm{init}}, \mathsf{isk}_1, \dots, \mathsf{isk}_Q\big) \text{ with } M = (Q, \mathbf{y}_{\mathrm{acc}}, \delta) \in \mathsf{TM} \qquad \text{and}$$
$$\mathsf{ct} \text{ as } \big((\mathbf{x}, T, S), \mathsf{ict}_{\mathrm{init}}, (\mathsf{ict}_{t,i,j,\mathbf{W}})_{t\in[T+1],i\in[N],j\in[S],\mathbf{W}\in\{0,1\}^S}\big) \text{ with } \mathbf{x} \in \{0,1\}^N.$$

The algorithm returns $\perp$ if $M|_{N,T,S}(\mathbf{x}) = 0$. Otherwise, it computes the labels (recall that $\mathcal{C}_{M,N,S} = [N] \times [S] \times \{0,1\}^S \times [Q]$)

$$[\![\ell_{\mathrm{init}}]\!]_{\mathrm{T}} \leftarrow \mathsf{IPFE.Dec}(\mathsf{isk}_{\mathrm{init}}, \mathsf{ict}_{\mathrm{init}}),$$
$$\text{for } t \in [T+1], (i, j, \mathbf{W}, q) \in \mathcal{C}_{M,N,S}\colon \quad [\![\ell_{t,i,j,\mathbf{W},q}]\!]_{\mathrm{T}} \leftarrow \mathsf{IPFE.Dec}(\mathsf{isk}_q, \mathsf{ict}_{t,i,j,\mathbf{W}}),$$

and recovers the encapsulated pad by

$$[\![\mu']\!]_{\mathrm{T}} \leftarrow \mathsf{Eval}\big((M, 1^N, 1^T, 1^{2^S}, p), \mathbf{x}, [\![\ell_{\mathrm{init}}]\!]_{\mathrm{T}}, ([\![\ell_{t,z}]\!]_{\mathrm{T}})_{t\in[T+1],z\in\mathcal{C}_{M,N,S}}\big).$$

The algorithm returns $[\![\mu']\!]_{\mathrm{T}}$ as the decapsulated pad.

Note: *We show the correctness of the scheme, which follows by an observation that* ℓ_{init}, $(\ell_{t,z})_{t\in[T+1],z\in\mathcal{C}_{M,N,S}}$ *is a valid garbling of* $M|_{N,T,S}$ *with secrets* $\alpha = \mu$ *and* $\beta = 0$. *Consider*

$$\alpha = \mu, \quad \beta = 0, \quad \mathbf{r} = \mathbf{r}_{\mathrm{x}} \otimes \mathbf{r}_{\mathrm{f}}, \quad \mathbf{r}_t = \mathbf{r}[(t, \sqcup, \sqcup, \sqcup, \sqcup)] \ (t \in [0..T]),$$
$$\big(\mathbf{L}_{\mathrm{init}}, (\mathbf{L}_{t,z})_{t\in[T+1],z\in\mathcal{C}_{M,N,S}}\big) = \mathsf{Garble}\big((M, 1^N, 1^T, 1^{2^S}, p), \alpha, \beta; \mathbf{r}\big),$$

where the randomness used by Garble *is explicitly written. Let* L_{init} *and* $L_{t,z}$*'s be the label functions (affine in* $\mathbf{x}$ *with coefficient vectors* $\mathbf{L}_{\mathrm{init}}$ *and* $\mathbf{L}_{t,z}$*'s, respectively). First, by the correctness of IPFE and the definition of vectors* $\mathbf{v}_{\mathrm{init}}$ *and* $\mathbf{u}_{\mathrm{init}}$*, we have*

$$\ell_{\mathrm{init}} = \mathbf{r}_{\mathrm{x}}[(0,1,1,\mathbf{0}_S)]\mathbf{r}_{\mathrm{f}}[1] = \mathbf{r}_0[(1,1,\mathbf{0}_S,1)] = \beta + \mathbf{e}^{\mathsf{T}}_{(1,1,\mathbf{0}_S,1)}\mathbf{r}_0 = L_{\mathrm{init}}(\mathbf{x}).$$

Next, by the correctness of IPFE and the definition of vectors $\mathbf{v}_q$ *and* $\mathbf{u}_{t,i,j,\mathbf{W}}$*, we have*

$$\begin{aligned}
\ell_{t,i,j,\mathbf{W},q} &= -\mathbf{r}_{\mathrm{x}}[(t-1,i,j,\mathbf{W})]\mathbf{r}_{\mathrm{f}}[q] + \sum_{\tau\in\mathcal{T}} \mathfrak{c}_\tau(\mathbf{x}; t, i, j, \mathbf{W}; \mathbf{r}_{\mathrm{x}})(\mathbf{M}_\tau\mathbf{r}_{\mathrm{f}})[q] \\
&= -\mathbf{r}_{t-1}[(i,j,\mathbf{W},q)] + \left(\sum_{\tau\in\mathcal{T}} \mathfrak{c}_\tau(\mathbf{x}; t, i, j, \mathbf{W}; \mathbf{r}_{\mathrm{x}})\mathbf{M}_\tau\mathbf{r}_{\mathrm{f}}\right)[q] \\
\text{(Equation (9))} \quad &= -\mathbf{r}_{t-1}[(i,j,\mathbf{W},q)] + \big(\mathbf{M}_{N,S}(\mathbf{x})\mathbf{r}_t\big)[(i,j,\mathbf{W},\sqcup)][q] \\
&= -\mathbf{r}_{t-1}[(i,j,\mathbf{W},q)] + \big(\mathbf{M}_{N,S}(\mathbf{x})\mathbf{r}_t\big)[(i,j,\mathbf{W},q)] \\
&= L_{t,i,j,\mathbf{W},q}(\mathbf{x}) \quad \text{for all } t \in [T], (i,j,\mathbf{W},q) \in \mathcal{C}_{M,N,S},
\end{aligned}$$

$$\begin{aligned}
\ell_{T+1,i,j,\mathbf{W},q} &= -\mathbf{r}_{\mathrm{x}}[(T,i,j,\mathbf{W})]\mathbf{r}_{\mathrm{f}}[q] + \alpha\mathbf{y}_{\mathrm{acc}}[q] \\
&= -\mathbf{r}_T[(i,j,\mathbf{W},q)] + \alpha\big(\mathbf{1}_{[N]\times[S]\times\{0,1\}^S} \otimes \mathbf{y}_{\mathrm{acc}}\big)[(i,j,\mathbf{W},q)] \\
&= L_{T+1,i,j,\mathbf{W},q}(\mathbf{x}) \quad \text{for all } (i,j,\mathbf{W},q) \in \mathcal{C}_{M,N,S}.
\end{aligned}$$

Lastly, by the linearity of Eval *in the labels, we can evaluate the garbling in the exponent of the target group and obtain* $\mu' = \alpha M|_{N,T,S}(\mathbf{x}) + \beta = \mu$ *(note that* $M|_{N,T,S}(\mathbf{x})$ *is either* 0 *or* 1 *for* $\mathbf{x} \in \{0,1\}^N$*) in the exponent. Therefore,* Dec *correctly decapsulates the pad.*

Remarks. The 1-ABE scheme for L based on MDDH_k can be obtained by modifying the above scheme to use pseudorandomness $\mathbf{r} = \mathbf{R}_{\mathrm{x}}^{\intercal}\mathbf{R}_{\mathrm{f}}$ for $\mathbf{R}_{\mathrm{x}} \xleftarrow{\$} \mathbb{Z}_p^{[k]\times([0..T]\times[N]\times[S]\times\{0,1\}^S)}$ and $\mathbf{R}_{\mathrm{f}} \xleftarrow{\$} \mathbb{Z}_p^{k\times Q}$.

Theorem 34. *Suppose in Construction 33, the* SXDH *assumption holds in $\mathcal{G}$ and the IPFE scheme is* function-hiding*, then the constructed 1-ABE scheme is* 1-key 1-ciphertext secure*.*

Proof Idea. We discuss the high-level ideas of the proof.

Recall that in the proof of adaptive security of 1-ABE for ABPs (Theorem 25), for the case where the secret key query appears before the ciphertext challenge, we gradually replace the label functions in the secret key by labels simulated as random values, except that the first label is reversely sampled.

That approach, however, fails for Construction 33 for two reasons: *i*) the garbling uses pseudorandomness instead of true randomness; *ii*) there is not enough space in the ciphertext, the secret key, or both to embed $\Omega(TNS2^SQ)$ labels simulated as random values. We make two tweaks to the proof of Theorem 25 to solve the problems: *i*) we rely on the SXDH assumption to gradually switch pseudorandomness to true randomness so that by special piecewise security, we can simulate the labels as random; *ii*) in the final hybrid, the labels are simulated as pseudorandom values, which can be embedded in the ciphertext and the secret key.

At a high level, the proof involves 3 groups of hybrids manipulating the vectors encrypted by IPFE:

1. The first few hybrids replace the first label function L_{init} jointly encoded in $\mathbf{u}_{\mathrm{init}}, \mathbf{v}_{\mathrm{init}}$ by the reversely sampled first label ℓ_{init} hardwired in either $\mathbf{u}_{\mathrm{init}}$ or $\mathbf{v}_{\mathrm{init}}$, whichever is generated later.

2. The middle hybrids are a loop. Along the hybrids, we gradually replace the label functions $L_{t,i,j,\mathbf{W},q}$ (jointly encoded in $\mathbf{u}_{t,i,j,\mathbf{W}}$'s and $\mathbf{v}_q$'s) generated using pseudorandomness by simulated labels $\ell_{t,i,j,\mathbf{W},q} = \mathbf{s}_{\mathrm{x}}[(t,i,j,\mathbf{W})]\mathbf{s}_{\mathrm{f}}[q]$, where $\mathbf{s}_{\mathrm{x}} \xleftarrow{\$} \mathbb{Z}_p^{[T+1]\times[N]\times[S]\times\{0,1\}^S}$ is embedded in $\mathbf{u}_{t,i,j,\mathbf{W}}$'s and $\mathbf{s}_{\mathrm{f}} \xleftarrow{\$} \mathbb{Z}_p^Q$ is embedded in $\mathbf{v}_q$'s. How the loop works depends on whether the ciphertext challenge comes first or second.

3. The last few hybrids do some clean-up work to completely remove μ^0 from the vectors so that it is information-theoretically hidden.

We elaborate on the loop hybrids. The reason why the strategy depends on whether the ciphertext challenge comes first or second is that we need to invoke the DDH assumption to temporarily make some label, say $\ell_{t,i,j,\mathbf{W},q}$, truly random. The reduction algorithm (reducing neighboring indistinguishability to DDH) needs to reversely sample ℓ_{init} using $\mathsf{RevSamp}$, which takes as input $M, \mathbf{x}, T, S$ and all the other labels $\ell_{t',i',j',\mathbf{W}',q'}$, including $\ell_{t,i,j,\mathbf{W},q}$. On one hand, since $M, \mathbf{x}, T, S$ are only fully specified after both the key and ciphertext queries are made, ℓ_{init} can only be embedded in the key if it comes after the ciphertext, and in the ciphertext otherwise. On the other hand, since $\ell_{t,i,j,\mathbf{W},q}$ is being switched from pseudorandom to random by DDH, the reduction only has access to $\ell_{t,i,j,\mathbf{W},q}$ in the exponent of the group G_b where DDH is invoked. Thus, ℓ_{init} must be reversely sampled in the exponent of G_b. Adding these requirements together necessitates that DDH must be invoked in the same group where ℓ_{init} is hardcoded:

- Case 1: The ciphertext challenge appears before the secret key query. In this case, $\ell_{\mathrm{init}} \xleftarrow{\$} \mathsf{RevSamp}(\cdots)$ is hardwired in $\mathbf{v}_{\mathrm{init}}$ (in $\mathsf{isk}_{\mathrm{init}}$, encoded in G_2). The middle hybrids loop over $(t,i,j,\mathbf{W})$ in lexicographical order. In each iteration, we first move everything from $\mathbf{u}_{t,i,j,\mathbf{W}}$ to all the

$\mathbf{v}_q$'s in one shot, hardwiring the honest labels $\ell_{t,i,j,\mathbf{W},q}$ into $\mathbf{v}_q$ for all q. Next, we invoke DDH in G_2 to switch $\mathbf{r}_t[(i,j,\mathbf{W},q)]$ from $\mathbf{r}_{\mathrm{x}}[(t,i,j,\mathbf{W})]\mathbf{r}_{\mathrm{f}}[q]$ to truly random for all q, which makes the labels $\ell_{t,i,j,\mathbf{W},q}$ truly random for all q. Then, we invoke DDH in G_2 again to make $\ell_{t,i,j,\mathbf{W},q} = \mathbf{s}_{\mathrm{x}}[(t,i,j,\mathbf{W})]\mathbf{s}_{\mathrm{f}}[q]$ pseudorandom for all q. Lastly, we move $\mathbf{s}_{\mathrm{x}}[(t,i,j,\mathbf{W})]$ from all $\mathbf{v}_q$'s back to $\mathbf{u}_{t,i,j,\mathbf{W}}$.

- Case 2: The secret key query appears before the ciphertext challenge. In this case, $\ell_{\mathrm{init}} \xleftarrow{\$} \mathsf{RevSamp}(\cdots)$ is hardwired in $\mathbf{u}_{\mathrm{init}}$ (in $\mathsf{ict}_{\mathrm{init}}$, encoded in G_1). The middle hybrids form a two-level loop with outer loop over t in increasing order and inner loop over q in increasing order. In each iteration, we first move all occurrences of $\mathbf{r}_{\mathrm{f}}[q]$ and $\mathbf{s}_{\mathrm{f}}[q]$ into all the $\mathbf{u}_{t',i',j',\mathbf{W}'}$'s in one shot (using the extra indices) and hardwire the honest labels $\ell_{t,i,j,\mathbf{W},q}$ into $\mathbf{u}_{t,i,j,\mathbf{W}}$'s for all $i,j,\mathbf{W}$. Next, we invoke DDH in G_1 to switch $\mathbf{r}_t[(i,j,\mathbf{W},q)]$ from $\mathbf{r}_{\mathrm{x}}[(t,i,j,\mathbf{W})]\mathbf{r}_{\mathrm{f}}[q]$ to truly random for all $i,j,\mathbf{W}$, making the labels $\ell_{t,i,j,\mathbf{W},q}$ truly random for all $i,j,\mathbf{W}$. Then, we invoke DDH in G_1 again to make $\ell_{t,i,j,\mathbf{W},q} = \mathbf{s}_{\mathrm{x}}[(t,i,j,\mathbf{W})]\mathbf{s}_{\mathrm{f}}[q]$ pseudorandom for all $i,j,\mathbf{W}$. Lastly, we move $\mathbf{r}_{\mathrm{f}}[q]$ and $\mathbf{s}_{\mathrm{f}}[q]$ back from all the $\mathbf{u}_{t',i',j',\mathbf{W}'}$'s.

The values must be carefully embedded in appropriate places, subject to the compactness requirement and the constraint that labels are still correctly computed, so that all the invocations of DDH are indeed valid.

We start with the first/last few hybrids, for which the two cases can be handled together, and then complete the proof by connecting the hybrids in two separate claims for the two cases.

Proof (Theorem 34). Let $\mathcal{A}$ be any efficient adversary. We want to show that the distinguishing advantage of $\mathcal{A}$ against $\mathsf{Exp}^0_{\text{1-sk,1-ct}}$ and $\mathsf{Exp}^1_{\text{1-sk,1-ct}}$ is negligible. In these two experiments, the adversary receives a single secret key sk encoding $M = (Q, \mathbf{y}_{\mathrm{acc}}, \delta)$ and secret μ^0, and a ciphertext ct encoding $(\mathbf{x}, 1^T, 1^{2^S})$. The only difference between these two experiments is that the adversary additionally receives either μ^0 or μ^1 and wants to distinguish them. Recall that by construction of our 1-ABE, the ciphertext ct contains a set of IPFE ciphertexts $\mathsf{ict}_{\mathrm{init}}$ and $\mathsf{ict}_{t,i,j,\mathbf{W}}$'s encrypting $\mathbf{u}_{\mathrm{init}}$ and $\mathbf{u}_{t,i,j,\mathbf{W}}$'s, and sk contains IPFE secret keys $\mathsf{isk}_{\mathrm{init}}$ and isk_q's encrypting $\mathbf{v}_{\mathrm{init}}$ and $\mathbf{v}_q$'s, respectively.

The first/last few hybrids are illustrated in Figure 6. The indices with superscripts are suppressed as they are only used in the loop hybrids. Note that though $\mathbf{v}$'s (in isk's) are listed before $\mathbf{u}$'s (in ict's), the argument about these hybrids is unaffected by whether the ciphertext challenge comes first or second.

- Hybrid H^b_0 proceeds identically to $\mathsf{Exp}^b_{\text{1-sk,1-ct}}$, where $\mathbf{v}$'s contain $\mu^0, \mathbf{r}_{\mathrm{f}}$ and the transition blocks, and $\mathbf{u}$'s contain $\mathbf{x}$ and $\mathbf{r}_{\mathrm{x}}$.
- Hybrid H^b_1 proceeds identically to H^b_0, except that ℓ_{init} is hardwired in $\mathbf{u}_{\mathrm{init}}$ or $\mathbf{v}_{\mathrm{init}}$, whichever is generated later, and that $\mathbf{s}_{\mathrm{f}} \xleftarrow{\$} \mathbb{Z}_p^Q$ is embedded in $\mathbf{v}_q[\mathsf{sim}]$'s. More specifically, the first change is implemented as follows:
  - If the *ciphertext* challenge comes first, $\mathbf{u}_{\mathrm{init}}[\mathsf{init}]$ is set to 1 (at encryption time) and $\mathbf{v}_{\mathrm{init}}[\mathsf{init}]$ is set to $\mathbf{r}_{\mathrm{x}}[(0,1,1,\mathbf{0}_S)]\mathbf{r}_{\mathrm{f}}[1]$ (at key generation time).
  - If the *secret key* query comes first, $\mathbf{v}_{\mathrm{init}}[\mathsf{init}]$ is set to 1 (at key generation time) and $\mathbf{u}_{\mathrm{init}}[\mathsf{init}]$ is set to $\mathbf{r}_{\mathrm{x}}[(0,1,1,\mathbf{0}_S)]\mathbf{r}_{\mathrm{f}}[1]$ (at encryption time).

  The first change prepares ℓ_{init} to be reversely sampled starting in the next hybrid by hardwiring it. The second change prepares the $\ell_{t,i,j,\mathbf{W},q}$'s to be simulated as pseudorandom values

hybrid	vector	init	$\mathsf{rand}, \mathsf{acc}, \mathsf{tb}_\tau$	sim
$\mathsf{H}_0^b \equiv \mathsf{Exp}^b_{\text{1-sk,1-ct}}$	$\mathbf{v}_{\text{init}}$	$\mathbf{r}_{\text{f}}[1]$		
	$\mathbf{v}_q$		normal	0
	$\mathbf{u}_{\text{init}}$	$\mathbf{r}_{\text{x}}[(0,1,1,0^S)]$		
	$\mathbf{u}_{t,i,j,\mathbf{W}}$		normal	0
H_1^b	$\mathbf{v}_{\text{init}}$	1 or $\mathbf{r}_{\text{x}}[(0,1,1,0^S)]\mathbf{r}_{\text{f}}[1]$		
	$\mathbf{v}_q$		normal	$\mathbf{s}_{\text{f}}[q]$
	$\mathbf{u}_{\text{init}}$	$\mathbf{r}_{\text{x}}[(0,1,1,0^S)]\mathbf{r}_{\text{f}}[1]$ or 1		
	$\mathbf{u}_{t,i,j,\mathbf{W}}$		normal	0
H_2^b	$\mathbf{v}_{\text{init}}$	1 or $\ell_{\text{init}} \leftarrow \mathsf{RevSamp}(\cdots)$		
	$\mathbf{v}_q$		normal	$\mathbf{s}_{\text{f}}[q]$
	$\mathbf{u}_{\text{init}}$	$\ell_{\text{init}} \leftarrow \mathsf{RevSamp}(\cdots)$ or 1		
	$\mathbf{u}_{t,i,j,\mathbf{W}}$		normal	0
loop		.....		
H_4^b	$\mathbf{v}_{\text{init}}$	1 or $\ell_{\text{init}} \leftarrow \mathsf{RevSamp}(\cdots)$		
	$\mathbf{v}_q$		normal	$\mathbf{s}_{\text{f}}[q]$
	$\mathbf{u}_{\text{init}}$	$\ell_{\text{init}} \leftarrow \mathsf{RevSamp}(\cdots)$ or 1		
	$\mathbf{u}_{t,i,j,\mathbf{W}}$		$0,0,0$	$\mathbf{s}_{\text{x}}[(t,i,j,\mathbf{W})]$
H_5^b	$\mathbf{v}_{\text{init}}$	1 or $\ell_{\text{init}} \leftarrow \mathsf{RevSamp}(\cdots)$		
	$\mathbf{v}_q$		$0,0,0$	$\mathbf{s}_{\text{f}}[q]$
	$\mathbf{u}_{\text{init}}$	$\ell_{\text{init}} \leftarrow \mathsf{RevSamp}(\cdots)$ or 1		
	$\mathbf{u}_{t,i,j,\mathbf{W}}$		$0,0,0$	$\mathbf{s}_{\text{x}}[(t,i,j,\mathbf{W})]$

The "normal" in...	rand	acc	tb_τ
$t \le T, \mathbf{u}_{t,i,j,\mathbf{W}}$:	$\mathbf{r}_{\text{x}}[t-1,i,j,\mathbf{W}]$	0	$\mathfrak{c}(\mathbf{x};t,i,j,\mathbf{W})$
$\mathbf{u}_{T+1,i,j,\mathbf{W}}$:	$\mathbf{r}_{\text{x}}[T,i,j,\mathbf{W}]$	1	0
$\mathbf{v}_q$:	$-\mathbf{r}_{\text{f}}[q]$	$\mu^0\mathbf{y}_{\text{acc}}[q]$	$(\mathbf{M}_\tau\mathbf{r}_{\text{f}})[q]$

In H_1^b,		sk before ct	ct before sk
	$\mathbf{v}_{\text{init}}[\mathsf{init}] =$	1	$\mathbf{r}_{\text{x}}[(0,1,1,\mathbf{0}_S)]\mathbf{r}_{\text{f}}[1]$
	$\mathbf{u}_{\text{init}}[\mathsf{init}] =$	$\mathbf{r}_{\text{x}}[(0,1,1,\mathbf{0}_S)]\mathbf{r}_{\text{f}}[1]$	1

In $\mathsf{H}_2^b, \mathsf{H}_4^b, \mathsf{H}_5^b$,		sk before ct	ct before sk
	$\mathbf{v}_{\text{init}}[\mathsf{init}] =$	1	$\mathsf{RevSamp}(\cdots)$
	$\mathbf{u}_{\text{init}}[\mathsf{init}] =$	$\mathsf{RevSamp}(\cdots)$	1

The reverse sampling works by (by the constraint, $\mu^0 M|_{N,T,S}(\mathbf{x}) = 0$)
$\ell_{\text{init}} \leftarrow \mathsf{RevSamp}\big((M, 1^N, 1^T, 1^{2^S}), \mathbf{x}, 0, (\ell_{t,z})_{t\in[T+1], z\in\mathcal{C}_{M,N,S}}\big)$.
In H_2^b, $\ell_{t,i,j,\mathbf{W},q} = L_{t,i,j,\mathbf{W},q}(\mathbf{x})$ are computed honestly using IPFE.
The garbling randomness is $\mathbf{r} = \mathbf{r}_{\text{x}} \otimes \mathbf{r}_{\text{f}}$ with $\mathbf{r}_{\text{x}}$ in ct and $\mathbf{r}_{\text{f}}$ in sk.
In H_4^b, $\ell_{t,i,j,\mathbf{W},q} = \mathbf{s}_{\text{x}}[(t,i,j,\mathbf{W})]\mathbf{s}_{\text{f}}[q]$ are simulated and computed using IPFE with $\mathbf{s}_{\text{x}}$ in ct and $\mathbf{s}_{\text{f}}$ in sk.
In H_5^b, the labels are simulated and μ^0 does not appear.

Figure 6: The first/last few hybrids in the security proof of 1-ABE for L.

in the loop hybrids — it has no effect at this moment since the newly embedded values in $\mathbf{v}$'s multiply with 0 in $\mathbf{u}$'s. The inner products in H_0^b and H_1^b remain unchanged, therefore, by the function-hiding property of IPFE, H_0^b and H_1^b are indistinguishable.

- Hybrid H_2^b proceeds identically to H_1^b, except that ℓ_{init} (was $\mathbf{r}_{\mathrm{x}}[(0,1,1,\mathbf{0}_S)]\mathbf{r}_{\mathrm{f}}[1]$) in $\mathbf{u}_{\text{init}}$ or $\mathbf{v}_{\text{init}}$ is reversely sampled from the other labels. By the special piecewise security of AKGS, H_1^b and H_2^b are identical.

- Hybrid H_4^b proceeds identically to H_2^b, except that the inner products between $\mathbf{u}_{t,i,j,\mathbf{W}}$'s and $\mathbf{v}_q$'s change from the honest labels to simulated labels $\mathbf{s}_{\mathrm{x}}[(t,i,j,\mathbf{W})]\mathbf{s}_{\mathrm{f}}[q]$. In Figure 6, this is reflected by the $\mathbf{u}_{t,i,j,\mathbf{W}}$'s having their values at $\mathsf{rand}, \mathsf{acc}$ and tb_τ's cleared and embedding $\mathbf{s}_{\mathrm{x}} \xleftarrow{\$} \mathbb{Z}_p^{[T+1]\times[N]\times[S]\times\{0,1\}^S}$ at sim. We will show $\mathsf{H}_2^b \approx \mathsf{H}_4^b$ as two separate claims.

  Claim 35. $\mathsf{H}_2^b \approx \mathsf{H}_4^b$ *conditioned on Case 1.*

  Claim 36. $\mathsf{H}_2^b \approx \mathsf{H}_4^b$ *conditioned on Case 2.*

- Hybrid H_5^b proceeds identically to H_4^b, except that all the $\mathbf{v}_q$'s have their (leftover) values at $\mathsf{rand}, \mathsf{acc}$ and tb_τ's cleared. These values multiply with 0 in $\mathbf{u}$'s in H_4^b, so the inner products remain unchanged, and $\mathsf{H}_5^b \approx \mathsf{H}_4^b$ by the function-hiding property of IPFE.

Observe that $\mathsf{H}_5^0 \equiv \mathsf{H}_5^1$ — since μ^0 never appears in the vectors, both μ^0 and μ^1 are just uniformly random values independent of everything else in the hybrids. By a hybrid argument, we conclude $\mathsf{Exp}^0_{\text{1-sk,1-ct}} \approx \mathsf{Exp}^1_{\text{1-sk,1-ct}}$. □

Proof (Claim 35). For Case 1, we show $\mathsf{H}_2^b \approx \mathsf{H}_4^b$ via a series of hybrids $\mathsf{H}_{3,t,i,j,\mathbf{W},1}, \ldots, \mathsf{H}^b_{3,t,i,j,\mathbf{W},5}$ for $(t,i,j,\mathbf{W}) \in [T+1]\times[N]\times[S]\times\{0,1\}^S$ in lexicographical order. Denote by $(t,i,j,\mathbf{W})+1$ the next tuple of indices in increasing order.

The hybrids are illustrated in Figure 7. Note that the $\mathbf{u}$'s are listed before the $\mathbf{v}$'s, as in Case 1, the ciphertext is generated before the secret key. In the figure, all the indices with superscripts except $\mathsf{sim}^{\mathsf{temp}}$ (temp means "temporary") are suppressed as they are not used.[15]

- Hybrid $\mathsf{H}^b_{3,t,i,j,\mathbf{W},1}$ proceeds identically to H_2^b, except that for all $(t',i',j',\mathbf{W}') < (t,i,j,\mathbf{W})$, $\mathbf{u}_{t',i',j',\mathbf{W}'}$ has its values in $\mathsf{rand}, \mathsf{acc}$ and tb_τ's cleared, and that a random value $\mathbf{s}_{\mathrm{x}}[t',i',j',\mathbf{W}']$ is embedded in $\mathbf{u}_{t',i',j',\mathbf{W}'}[\mathsf{sim}]$. This means all the labels with $(t',i',j',\mathbf{W}') < (t,i,j,\mathbf{W})$ are simulated, the first label ℓ_{init} is reversely sampled, and the rest are honestly computed.

- Hybrid $\mathsf{H}^b_{3,t,i,j,\mathbf{W},2}$ proceeds identically to $\mathsf{H}^b_{3,t,i,j,\mathbf{W},1}$, except that the values in $\mathbf{u}_{t,i,j,\mathbf{W}}$ are zeroed out, and its inner products with all $\mathbf{v}_q$'s — the labels $\ell_{t,i,j,\mathbf{W},q}$ for all q's — are hardcoded into $\mathbf{v}_q$'s:
  - The values of $\mathbf{u}_{t,i,j,\mathbf{W}}$ at $\mathsf{rand}, \mathsf{acc}$ and tb_τ's are set to 0.
  - $\mathbf{u}_{t,i,j,\mathbf{W}}[\mathsf{sim}^{\mathsf{temp}}]$ is set to 1 to pick up the values $\mathbf{v}_q[\mathsf{sim}^{\mathsf{temp}}]$ for all q.
  - The honest labels $\ell_{t,i,j,\mathbf{W},q} = -\mathbf{r}_{\mathrm{x}}[(t-1,i,j,\mathbf{W})]\mathbf{r}_{\mathrm{f}}[q] + \cdots$ are embedded in $\mathbf{v}_q[\mathsf{sim}^{\mathsf{temp}}]$ for each $q \in [Q]$, where "$\cdots$" is either $\mu^0\mathbf{y}_{\mathrm{acc}}[q]$ (if $t = T+1$) or $\sum_{\tau\in\mathcal{T}} \mathfrak{c}_\tau(\mathbf{x};t,i,j,\mathbf{W};\mathbf{r}_{\mathrm{x}})(\mathbf{M}_\tau\mathbf{r}_{\mathrm{f}})[q]$ (if $t \le T$; it is $(\mathbf{M}_{N,S}\mathbf{r}_t)[(t,i,j,\mathbf{W},q)]$ for pseudorandom $\mathbf{r}_t = \mathbf{r}_{\mathrm{x}}[t,\sqcup,\sqcup,\sqcup] \otimes \mathbf{r}_{\mathrm{f}}$).

[15] Case 1 is the simpler case, though it is already not simple. Note that this coincides with the selective (or semi-adaptive) case, but the simplicity is more of a consequence of the AKGS structure than that of selectivity.

hybrid	vector	rand	acc	tb_τ	sim	$\mathsf{sim}^{\mathsf{temp}}$
$\mathsf{H}^b_{3,t,i,j,\mathbf{W},1}$	$\mathbf{u}_{t',i',j',\mathbf{W}'}$ $_{<(t,i,j,\mathbf{W})}$	0	0	0	$\mathbf{s}_{\mathrm{x}}[(t',i',j',\mathbf{W}')]$	0
	$\mathbf{u}_{t,i,j,\mathbf{W}}$	$\mathbf{r}_{\mathrm{x}}[(t-1,i,j,\mathbf{W})]$	0 or 1	$\mathfrak{c}_\tau(\mathbf{x};t,i,j,\mathbf{W};\mathbf{r}_{\mathrm{x}})$ or 0	0	0
	$\mathbf{u}_{t',i',j',\mathbf{W}'}$ $_{>(t,i,j,\mathbf{W})}$	$\mathbf{r}_{\mathrm{x}}[(t'-1,i',j',\mathbf{W}')]$	0 or 1	$\mathfrak{c}_\tau(\mathbf{x};t',i',j',\mathbf{W}';\mathbf{r}_{\mathrm{x}})$ or 0	0	0
	$\mathbf{v}_q$	$-\mathbf{r}_{\mathrm{f}}[q]$	$\mu^0\mathbf{y}_{\mathrm{acc}}[q]$	$(\mathbf{M}_\tau\mathbf{r}_{\mathrm{f}})[q]$	$\mathbf{s}_{\mathrm{f}}[q]$	0
$\mathsf{H}^b_{3,t,i,j,\mathbf{W},2}$	$\mathbf{u}_{t,i,j,\mathbf{W}}$	0	0	0	0	1
	$\mathbf{v}_q$	$-\mathbf{r}_{\mathrm{f}}[q]$	$\mu^0\mathbf{y}_{\mathrm{acc}}[q]$	$(\mathbf{M}_\tau\mathbf{r}_{\mathrm{f}})[q]$	$\mathbf{s}_{\mathrm{f}}[q]$	honest $\ell_{t,i,j,\mathbf{W},q}=$ $-\mathbf{r}_{\mathrm{x}}[(t-1,i,j,\mathbf{W})]\mathbf{r}_{\mathrm{f}}[q]+\cdots$
$\mathsf{H}^b_{3,t,i,j,\mathbf{W},3}$	$\mathbf{u}_{t,i,j,\mathbf{W}}$	0	0	0	0	1
	$\mathbf{v}_q$	$-\mathbf{r}_{\mathrm{f}}[q]$	$\mu^0\mathbf{y}_{\mathrm{acc}}[q]$	$(\mathbf{M}_\tau\mathbf{r}_{\mathrm{f}})[q]$	$\mathbf{s}_{\mathrm{f}}[q]$	$\ell_{t,i,j,\mathbf{W},q} \xleftarrow{\$} \mathbb{Z}_p$
$\mathsf{H}^b_{3,t,i,j,\mathbf{W},4}$	$\mathbf{u}_{t,i,j,\mathbf{W}}$	0	0	0	0	1
	$\mathbf{v}_q$	$-\mathbf{r}_{\mathrm{f}}[q]$	$\mu^0\mathbf{y}_{\mathrm{acc}}[q]$	$(\mathbf{M}_\tau\mathbf{r}_{\mathrm{f}})[q]$	$\mathbf{s}_{\mathrm{f}}[q]$	simulated $\ell_{t,i,j,\mathbf{W},q}=$ $\mathbf{s}_{\mathrm{x}}[(t,i,j,\mathbf{W})]\mathbf{s}_{\mathrm{f}}[q]$
$\mathsf{H}^b_{3,t,i,j,\mathbf{W},5}$ $\equiv$ $\mathsf{H}^b_{3,t',i',j',\mathbf{W}',1}$ for $(t',i',j',\mathbf{W}')$ $=$ $(t,i,j,\mathbf{W})+1$	$\mathbf{u}_{t',i',j',\mathbf{W}'}$ $_{<(t,i,j,\mathbf{W})}$	0	0	0	$\mathbf{s}_{\mathrm{x}}[(t',i',j',\mathbf{W}')]$	0
	$\mathbf{u}_{t,i,j,\mathbf{W}}$	0	0	0	$\mathbf{s}_{\mathrm{x}}[(t,i,j,\mathbf{W})]$	0
	$\mathbf{u}_{t',i',j',\mathbf{W}'}$ $_{>(t,i,j,\mathbf{W})}$	$\mathbf{r}_{\mathrm{x}}[(t'-1,i',j',\mathbf{W}')]$	0 or 1	$\mathfrak{c}_\tau(\mathbf{x};t',i',j',\mathbf{W}';\mathbf{r}_{\mathrm{x}})$ or 0	0	0
	$\mathbf{v}_q$	$-\mathbf{r}_{\mathrm{f}}[q]$	$\mu^0\mathbf{y}_{\mathrm{acc}}[q]$	$(\mathbf{M}_\tau\mathbf{r}_{\mathrm{f}})[q]$	$\mathbf{s}_{\mathrm{f}}[q]$	0

For brevity, $\mathbf{u}_{\mathrm{init}}$ and $\mathbf{v}_{\mathrm{init}}$ are suppressed. The reversely sampled ℓ_{init} is hardwired in $\mathbf{v}_{\mathrm{init}}$, and is only needed (and can only be computed so by the reduction) in the exponent of G_2:

$$[\![\ell_{\mathrm{init}}]\!]_2 \leftarrow \mathsf{RevSamp}\big((M,1^N,1^T,1^{2^S}),\mathbf{x},[\![0]\!]_2,([\![\ell_{t,z}]\!]_2)_{t\in[T+1],z\in\mathcal{C}_{M,N,S}}\big).$$

In the intermediate hybrids, $\mathbf{u}_{t',i',j',\mathbf{W}'}$'s are suppressed. They remain unchanged in this iteration.
The choices between "0 or 1" and "$\mathfrak{c}_\tau(\cdots)$ or 0" are the former if $t' \le T$, and the latter if $t' = T+1$.

In this interation, the labels $\ell_{t',i',j',\mathbf{W}',q}$ with $(t',i',j',\mathbf{W}')$... are...

- $< (t,i,j,\mathbf{W})$: simulated as $\mathbf{s}_{\mathrm{x}}[(t',i',j',\mathbf{W}')]\mathbf{s}_{\mathrm{f}}[q]$ and computed using IPFE
- $= (t,i,j,\mathbf{W})$: computed honestly using IPFE
  - $\to$ computed honestly and hardwired in sk
  - $\to$ simulated as random and hardwired in sk
  - $\to$ simulated as $\mathbf{s}_{\mathrm{x}}[(t,i,j,\mathbf{W})]\mathbf{s}_{\mathrm{f}}[q]$ and hardwired in sk
  - $\to$ simulated as $\mathbf{s}_{\mathrm{x}}[(t,i,j,\mathbf{W})]\mathbf{s}_{\mathrm{f}}[q]$ and computed using IPFE
- $> (t,i,j,\mathbf{W})$: computed honestly using IPFE

When a label is computed honestly, the garbling randomness is $\mathbf{r} = \mathbf{r}_{\mathrm{x}} \otimes \mathbf{r}_{\mathrm{f}}$.
When a label is computed using IPFE, $\mathbf{r}_{\mathrm{x}}, \mathbf{s}_{\mathrm{x}}$ components are in ct and $\mathbf{r}_{\mathrm{f}}, \mathbf{s}_{\mathrm{f}}$ components are in sk.

Figure 7: The loop hybrids in the security proof of 1-ABE for L for the case where the ciphertext challenge comes before the secret key query.

As depicted in Figure 7, the inner products in $\mathsf{H}^b_{3,t,i,j,\mathbf{W},1}$ and $\mathsf{H}^b_{3,t,i,j,\mathbf{W},2}$ remain unchanged, so they are indistinguishable by the function-hiding property of IPFE.

- Hybrid $\mathsf{H}^b_{3,t,i,j,\mathbf{W},3}$ proceeds identically to $\mathsf{H}^b_{3,t,i,j,\mathbf{W},2}$, except that the labels $\ell_{t,i,j,\mathbf{W},q}$ are replaced by truly random values. Observe that in both $\mathsf{H}^b_{3,t,i,j,\mathbf{W},2}$ and $\mathsf{H}^b_{3,t,i,j,\mathbf{W},3}$, the random values $\mathbf{r}_{\mathrm{x}}[(t-1,i,j,\mathbf{W})]$ and $\mathbf{r}_{\mathrm{f}}[q]$'s only appear in G_2 (the group encoding secret key vectors). The indistinguishability between them reduces to the DDH assumption in G_2: Given an $\mathrm{MDDH}_{1,q}$ challenge

$$[\![\mathbf{r}_{\mathrm{f}}[1],\ldots,\mathbf{r}_{\mathrm{f}}[Q];\, z_1,\ldots,z_Q]\!]_2$$
$$\text{with } \begin{cases} z_q = \mathbf{r}_{\mathrm{x}}[(t-1,i,j,\mathbf{W})]\mathbf{r}_{\mathrm{f}}[q], & \text{if a DDH tuple is given;} \\ z_q \xleftarrow{\$} \mathbb{Z}_p, & \text{if a truly random tuple is given;} \end{cases}$$

  we compute the labels $\ell_{t,i,j,\mathbf{W},q}$ as $-z_q + \cdots$. If a DDH tuple is given, these labels use pseudorandom *randomizers* $\mathbf{r}_{t-1}[(i,j,\mathbf{W},\sqcup)] = \mathbf{r}_{\mathrm{x}}[(t-1,i,j,\mathbf{W})]\mathbf{r}_{\mathrm{f}}$ as in $\mathsf{H}^b_{3,t,i,j,\mathbf{W},2}$. If a truly random tuple is given, these labels use truly random *randomizers* $\mathbf{r}_{t-1}[(i,j,\mathbf{W},\sqcup)] \xleftarrow{\$} \mathbb{Z}_p^Q$, thus are themselves truly random (as in $\mathsf{H}^b_{3,t,i,j,\mathbf{W},3}$) due to the special piecewise security of AKGS. Note that everything else in the two hybrids can be efficiently computed — in particular, the value $[\![\ell_{\mathrm{init}}]\!]_2 \leftarrow \mathsf{RevSamp}(\cdots)$ can be computed efficiently *in the exponent*.

- Hybrid $\mathsf{H}^b_{3,t,i,j,\mathbf{W},4}$ proceeds identically to $\mathsf{H}^b_{3,t,i,j,\mathbf{W},3}$, except that the truly random labels $\ell_{t,i,j,\mathbf{W},q}$ for all $q \in [Q]$ are replaced by pseudorandom values $\mathbf{s}_{\mathrm{x}}[(t,i,j,\mathbf{W})]\mathbf{s}_{\mathrm{f}}[q]$ with[16] $\mathbf{s}_{\mathrm{x}}[(t,i,j,\mathbf{W})] \xleftarrow{\$} \mathbb{Z}_p$. The indistinguishability between the two hybrids reduces to the DDH assumption in G_2.

- Hybrid $\mathsf{H}^b_{3,t,i,j,\mathbf{W},5}$ proceeds identically to $\mathsf{H}^b_{3,t,i,j,\mathbf{W},4}$, except that the pseudorandom labels $\ell_{t,i,j,\mathbf{W},q} = \mathbf{s}_{\mathrm{x}}[(t,i,j,\mathbf{W})]\mathbf{s}_{\mathrm{f}}[q]$ hardwired in $\mathbf{v}_q[\mathsf{sim}^{\mathsf{temp}}]$'s are *split* into $\mathbf{u}_{t,i,j,\mathbf{W}}[\mathsf{sim}]$ (embedding the factor $\mathbf{s}_{\mathrm{x}}[(t,i,j,\mathbf{W})]$) and $\mathbf{v}_q[\mathsf{sim}]$'s (embedding the factor $\mathbf{s}_{\mathrm{f}}[q]$), as shown in Figure 7. The inner products in $\mathsf{H}^b_{3,t,i,j,\mathbf{W},4}$ and $\mathsf{H}^b_{3,t,i,j,\mathbf{W},5}$ remain unchanged, so the two hybrids are indistinguishable by the function-hiding property of IPFE. Moreover, $\mathsf{H}^b_{3,t,i,j,\mathbf{W},5} \equiv \mathsf{H}^b_{3,t',i',j',\mathbf{W}',1}$ for $(t',i',j',\mathbf{W}') = (t,i,j,\mathbf{W}) + 1$.

By a hybrid argument, we have $\mathsf{H}^b_{3,1,1,1,\mathbf{0}_S,1} \approx \mathsf{H}^b_{3,T+1,N,S,\mathbf{1}_S,5}$. Lastly, observe that $\mathsf{H}^b_{3,1,1,1,\mathbf{0}_S,1} \equiv \mathsf{H}^b_2$ and $\mathsf{H}^b_{3,T+1,N,S,\mathbf{1}_S,5} \equiv \mathsf{H}^b_4$. Therefore, $\mathsf{H}^b_2 \approx \mathsf{H}^b_4$ conditioned on Case 1. □

Proof (Claim 36). The proof of $\mathsf{H}^b_2 \approx \mathsf{H}^b_4$ is more involved in Case 2. In this case, the ciphertext challenge appears after the secret key query, and we must hardwire the reversely sampled ℓ_{init} in the ciphertext.

Suppose at some point, we want to make $\ell_{t,i,j,\mathbf{W},q}$ truly random by invoking DDH. In Case 1, we move $\mathbf{r}_{\mathrm{x}}[(t-1,i,j,\mathbf{W})]$ into the secret key vectors, which is easy because by the time we need to invoke DDH on $\mathbf{r}_{\mathrm{x}}[(t-1,i,j,\mathbf{W})]$, it only appears in one place (namely, in $\mathbf{u}_{t,i,j,\mathbf{W}}[\mathsf{rand}]$) and goes away after the invocation of DDH. In Case 2, since $\ell_{\mathrm{init}} \leftarrow \mathsf{RevSamp}(\cdots)$ must be computed in the exponent in G_1, the only option is to move $\mathbf{r}_{\mathrm{f}}[q]$ into the ciphertext vectors. However, depending on the transition blocks, $\mathbf{r}_{\mathrm{f}}[q]$ might appear in $(\mathbf{M}_\tau \mathbf{r}_{\mathrm{f}})[q']$ of any $\mathbf{v}_{q'}$. There are many limitations to take into consideration when sorting out the proof:

[16] Note that this is *the first hybrid* in which $\mathbf{s}_{\mathrm{x}}[(t,i,j,\mathbf{W})]$ appears — each such value is introduced only when needed.

- The special piecewise security only allows us to simulate $\ell_{t,i,j,\mathbf{W},q}$'s in increasing order of t.
- To simulate $\ell_{t,i,j,\mathbf{W},q}$, all occurrences of $\mathbf{r}_{\mathrm{f}}[q]$ must be in the ciphertext.
- There is not enough space in the ciphertext to embed all the $\mathbf{r}_{\mathrm{f}}[q]$'s at the same time.
- The value $\mathbf{r}_{\mathrm{f}}[q]$ must *not* go away until *all* $\ell_{t,i,j,\mathbf{W},q}$'s are simulated. Indeed, $\mathbf{r}_{\mathrm{f}}[q]$ still resides in $\mathbf{v}_{q'}$'s in H_4^b, the end hybrid (for this claim).

Combining these factors, a feasible strategy is to switch (in increasing order of t, q) batches of $NS2^S$ label functions (i.e., for fixed t, q and all $i, j, \mathbf{W}$) into simulated labels by moving $\mathbf{r}_{\mathrm{f}}[q]$'s back and forth in each iteration (to keep the labels correctly computed) until all the labels are simulated. The above also applies to $\mathbf{s}_{\mathrm{f}}[q]$ when we consider invoking DDH again to make truly random labels only pseudorandom.

More specifically, in iteration t, q, when moving $\mathbf{r}_{\mathrm{f}}[q]$ into the ciphertext vectors, we erase all its occurrences in the secret key vectors and must *compensate* some $\ell_{t',i,j,\mathbf{W},q'}$'s for their loss of $\mathbf{r}_{\mathrm{f}}[q]$ using the indices with superscript comp. This is done by separating out the terms with $\mathbf{r}_{\mathrm{f}}[q]$. See the compensation identity in the notes of Figure 9 for details.

To better understand the procedure, we define the *modes* of a label $\ell_{t',i,j,\mathbf{W},q'}$. There are three orthogonal groups of modes. Each label has one mode in each group. We will refer to these modes in the description of hybrids below, their meaning will also become clearer in the context of hybrids.

- The first group is about the value of the label. A label is *honest* if its value is $L_{t',i,j,\mathbf{W},q'}(\mathbf{x})$ with garbling randomness set to $\mathbf{r} = \mathbf{r}_{\mathrm{x}} \otimes \mathbf{r}_{\mathrm{f}}$. It is *random* if its value is sampled uniformly at random. It is *simulated* if its value is $\mathbf{s}_{\mathrm{x}}[(t', i, j, \mathbf{W})]\mathbf{s}_{\mathrm{f}}[q']$.
- The second group is about where the terms $\mathbf{r}_{\mathrm{f}}, \mathbf{s}_{\mathrm{f}}$ of a label resides. A label is normal (this is the default) if $\mathbf{r}_{\mathrm{f}}, \mathbf{s}_{\mathrm{f}}$ resides in the secret key. It is *compensated* if $\mathbf{r}_{\mathrm{f}}[q], \mathbf{s}_{\mathrm{f}}[q]$ are in the ciphertext with the other components of $\mathbf{r}_{\mathrm{f}}, \mathbf{s}_{\mathrm{f}}$ in the secret key. It is *hardwired* if the value (in its entirety) is hardwired in the ciphertext (for conceptual simplicity, this mode only applies to the labels with $t' = t, q' = q$). A non-normal label will use indices with superscript comp.
- The last group is a highly technical one. A label is normal (default) if it is computed without indices with superscript temp. It is *temporary* if it is computed with indices with superscript temp (a label can be in this mode only when $t' = t$).

The loop hybrids are a *two-level loop* with outer loop over $t = 1, \ldots, T+1$ (Figure 8) and inner loop over $q = 1, \ldots, Q$ (Figure 9). In these hybrids, $\mathbf{u}$'s (in ict's) appear after $\mathbf{v}$'s (in isk's). The outer loop consists of the following hybrids:

- Hybrid $\mathsf{H}_{3,t,1}^b$ proceeds identically to H_2^b, except that for all $t' < t$ and all $i, j, \mathbf{W}$, the vectors $\mathbf{u}_{t',i,j,\mathbf{W}}$ have their values at $\mathsf{rand}, \mathsf{acc}$ and tb_τ's cleared and embed $\mathbf{s}_{\mathrm{x}}[(t', i, j, \mathbf{W})]$ at sim. In this hybrid, labels with $t = t'$ are in the *honest* mode.
- Hybrid $\mathsf{H}_{3,t,2}^b$ proceeds identically to $\mathsf{H}_{3,t,1}^b$, except that the mode of $\ell_{t,i,j,\mathbf{W},q}$'s (for all $i, j, \mathbf{W}, q$) are changed to *honest and temporary*, and that a random value $\mathbf{s}_{\mathrm{x}}[(t, i, j, \mathbf{W})]$ is embedded in $\mathbf{u}_{t,i,j,\mathbf{W}}[\mathsf{sim}^{\mathsf{temp}}]$ for all $i, j, \mathbf{W}$. The first change is implemented as follows:
  - For all q, the values of $\mathbf{v}_q$'s at $\mathsf{rand}, \mathsf{acc}$ and tb_τ are *copied* to their counterparts with superscript temp.

hybrid	vector	$\mathsf{rand}, \mathsf{acc}, \mathsf{tb}_\tau$	$\mathsf{rand}^{\mathsf{temp}}, \mathsf{acc}^{\mathsf{temp}}, \mathsf{tb}_\tau^{\mathsf{temp}}$	sim	$\mathsf{sim}^{\mathsf{temp}}$
$\mathsf{H}^b_{3,t,1}$	$\mathbf{v}_q$	normal	$0,0,0$	$\mathbf{s}_{\mathrm{f}}[q]$	0
	$\mathbf{u}_{t'<t,i,j,\mathbf{W}}$	$0,0,0$	$0,0,0$	$\mathbf{s}_{\mathrm{x}}[(t',i,j,\mathbf{W})]$	0
	$\mathbf{u}_{t,i,j,\mathbf{W}}$	normal	$0,0,0$	0	0
	$\mathbf{u}_{t'>t,i,j,\mathbf{W}}$	normal	$0,0,0$	0	0
$\mathsf{H}^b_{3,t,2} \equiv \mathsf{H}^b_{3,t,3,1,1}$	$\mathbf{v}_q$	normal	normal	$\mathbf{s}_{\mathrm{f}}[q]$	0
	$\mathbf{u}_{t'<t,i,j,\mathbf{W}}$	$0,0,0$	$0,0,0$	$\mathbf{s}_{\mathrm{x}}[(t',i,j,\mathbf{W})]$	0
	$\mathbf{u}_{t,i,j,\mathbf{W}}$	$0,0,0$	normal	0	$\mathbf{s}_{\mathrm{x}}[(t,i,j,\mathbf{W})]$
	$\mathbf{u}_{t'>t,i,j,\mathbf{W}}$	normal	$0,0,0$	0	0
$\mathsf{H}^b_{3,t,3,1\sim Q,1\sim 5}$			.....		
$\mathsf{H}^b_{3,t,4} \equiv \mathsf{H}^b_{3,t,3,Q,5}$	$\mathbf{v}_q$	normal	$0,0,0$	$\mathbf{s}_{\mathrm{f}}[q]$	$\mathbf{s}_{\mathrm{f}}[q]$
	$\mathbf{u}_{t'<t,i,j,\mathbf{W}}$	$0,0,0$	$0,0,0$	$\mathbf{s}_{\mathrm{x}}[(t',i,j,\mathbf{W})]$	0
	$\mathbf{u}_{t,i,j,\mathbf{W}}$	$0,0,0$	normal	0	$\mathbf{s}_{\mathrm{x}}[(t,i,j,\mathbf{W})]$
	$\mathbf{u}_{t'>t,i,j,\mathbf{W}}$	normal	$0,0,0$	0	0
$\mathsf{H}^b_{3,t,5} \equiv \mathsf{H}^b_{3,t+1,1}$	$\mathbf{v}_q$	normal	$0,0,0$	$\mathbf{s}_{\mathrm{f}}[q]$	0
	$\mathbf{u}_{t'<t,i,j,\mathbf{W}}$	$0,0,0$	$0,0,0$	$\mathbf{s}_{\mathrm{x}}[(t',i,j,\mathbf{W})]$	0
	$\mathbf{u}_{t,i,j,\mathbf{W}}$	$0,0,0$	$0,0,0$	$\mathbf{s}_{\mathrm{x}}[(t,i,j,\mathbf{W})]$	0
	$\mathbf{u}_{t'>t,i,j,\mathbf{W}}$	normal	$0,0,0$	0	0

For brevity, $\mathbf{u}_{\mathrm{init}}$ and $\mathbf{v}_{\mathrm{init}}$ are suppressed.
The reversely sampled ℓ_{init} is hardwired in $\mathbf{u}_{\mathrm{init}}$:

$$\ell_{\mathrm{init}} \leftarrow \mathsf{RevSamp}\big((M, 1^N, 1^T, 1^{2^S}), \mathbf{x}, 0, (\ell_{t,z})_{t\in[T+1],z\in\mathcal{C}_{M,N,S}}\big).$$

The "normal" in...	$\mathsf{rand}, \mathsf{rand}^{\mathsf{temp}}$	$\mathsf{acc}, \mathsf{acc}^{\mathsf{temp}}$	$\mathsf{tb}_\tau, \mathsf{tb}_\tau^{\mathsf{temp}}$
$\mathbf{v}_q$:	$-\mathbf{r}_{\mathrm{f}}[q]$	$\mu^0\mathbf{y}_{\mathrm{acc}}[q]$	$(\mathbf{M}_\tau\mathbf{r}_{\mathrm{f}})[q]$
$t' \le T, \mathbf{u}_{t',i,j,\mathbf{W}}$:	$\mathbf{r}_{\mathrm{x}}[t'-1,i,j,\mathbf{W}]$	0	$\mathfrak{c}(\mathbf{x};t',i,j,\mathbf{W})$
$\mathbf{u}_{T+1,i,j,\mathbf{W}}$:	$\mathbf{r}_{\mathrm{x}}[T,i,j,\mathbf{W}]$	1	0

In this iteration, the *modes* of $\ell_{t',i,j,\mathbf{W},q}$ are...
- $t' < t$: simulated
- $t' = t$: honest $\to$ honest and temporary $\to$ simulated and temporary $\to$ simulated
- $t' > t$: honest

Figure 8: The outer loop hybrids in the security proof of 1-ABE for L for the case where the ciphertext challenge comes after the secret key query.

– For all $i, j, \mathbf{W}$, the values of $\mathbf{u}_{t,i,j,\mathbf{W}}$'s at $\mathsf{rand}, \mathsf{acc}$ and tb_τ are *moved* to their counterparts with superscript temp.

This way, the labels are still correctly computed, i.e., the inner products remain unchanged. The second change prepares $\ell_{t,i,j,\mathbf{W},q}$'s to be simulated and currently has no effect — the newly embedded values multiply with 0 in the $\mathbf{v}_q$'s. Since the inner products remain unchanged, $\mathsf{H}^b_{3,t,1} \approx \mathsf{H}^b_{3,t,2}$ by the function-hiding property of IPFE.

- Hybrid $\mathsf{H}^b_{3,t,4}$ proceeds identically to $\mathsf{H}^b_{3,t,2}$, except that the mode of $\ell_{t,i,j,\mathbf{W},q}$'s (for all $i, j, \mathbf{W}, q$) is switched from *honest and temporary* to *simulated and temporary*. In Figure 8, this is reflected by $\mathbf{v}_q$'s (for all q) having their values at $\mathsf{rand}^{\mathsf{temp}}, \mathsf{acc}^{\mathsf{temp}}$ and $\mathsf{tb}^{\mathsf{temp}}_\tau$'s cleared and embedding $\mathbf{s}_{\mathrm{f}}[q]$ at $\mathsf{sim}^{\mathsf{temp}}$. We will show $\mathsf{H}^b_{3,t,2} \approx \mathsf{H}^b_{3,t,4}$ via a subseries of hybrids (the inner loop hybrids).

- Hybrid $\mathsf{H}^b_{3,t,5}$ proceeds identically to $\mathsf{H}^b_{3,t,4}$, except that the mode of $\ell_{t,i,j,\mathbf{W},q}$'s is switched from *simulated and temporary* to *simulated* and some clean-up work is done in preparation for the next iteration:

  – $\mathbf{v}_q$'s (for all q) have their values at $\mathsf{sim}^{\mathsf{temp}}$ cleared.
  – $\mathbf{u}_{t,i,j,\mathbf{W}}$'s (for all $i, j, \mathbf{W}$) have $\mathbf{s}_{\mathrm{f}}[(t, i, j, \mathbf{W})]$'s moved from $\mathsf{sim}^{\mathsf{temp}}$ into sim.
  – $\mathbf{u}_{t,i,j,\mathbf{W}}$'s (for all $i, j, \mathbf{W}$) have their values at $\mathsf{rand}^{\mathsf{temp}}, \mathsf{acc}^{\mathsf{temp}}$ and $\mathsf{tb}^{\mathsf{temp}}_\tau$'s cleared.

  It is easy to check that the inner products remain unchanged, so $\mathsf{H}^b_{3,t,4} \approx \mathsf{H}^b_{3,t,5}$ by the function-hiding property of IPFE. Furthermore, we have $\mathsf{H}^b_{3,t,5} \equiv \mathsf{H}^b_{3,t+1,1}$.

Observe that $\mathsf{H}^b_{3,1,1} \equiv \mathsf{H}^b_2$ and $\mathsf{H}^b_{3,T+2,5} \equiv \mathsf{H}^b_4$. Once we prove $\mathsf{H}^b_{3,t,2} \approx \mathsf{H}^b_{3,t,4}$, we can conclude $\mathsf{H}^b_2 \approx \mathsf{H}^b_4$ by a hybrid argument.

The inner loop hybrids connect $\mathsf{H}^b_{3,t,2}$ and $\mathsf{H}^b_{3,t,4}$ by switching $\ell_{t,i,j,\mathbf{W},q}$ from *honest and temporary* to *simulated and temporary* one by one for $q = 1, \ldots, Q$:

- Hybrid $\mathsf{H}^b_{3,t,3,q,1}$ proceeds identically to $\mathsf{H}^b_{3,t,2}$, except that for $q' < q$, all the $\mathbf{v}_{q'}$ have their values at $\mathsf{rand}^{\mathsf{temp}}, \mathsf{acc}^{\mathsf{temp}}$ and $\mathsf{tb}^{\mathsf{temp}}_\tau$'s cleared, and the value $\mathbf{s}_{\mathrm{f}}[q']$ is embedded at $\mathsf{sim}^{\mathsf{temp}}$, i.e., the labels $\ell_{t,i,j,\mathbf{W},q'}$ for all $i, j, \mathbf{W}$ and all $q' < q$ have been switched from *honest and temporary* to *simulated and temporary*.

- Hybrid $\mathsf{H}^b_{3,t,3,q,2}$ proceeds identically to $\mathsf{H}^b_{3,t,3,q,1}$, except that all occurrences of $\mathbf{r}_{\mathrm{f}}[q]$ and $\mathbf{s}_{\mathrm{f}}[q]$ are moved from $\mathbf{v}_{q'}$'s into $\mathbf{v}_{t',i,j,\mathbf{W}}$'s according to the compensation identity. The implementation is illustrated in Figure 9. The labels with $q' = q$ or $t' > t$ or $t' = t, q' > q$ gain *compensated* mode on top of their existing modes, and the labels $\ell_{t,i,j,\mathbf{W},q}$ for all $i, j, \mathbf{W}$ become *honest and hardwired* (hardwired in $\mathbf{u}_{t,i,j,\mathbf{W}}[\mathsf{sim}^{\mathsf{comp}}]$). The inner products (the labels) remain unchanged, so $\mathsf{H}^b_{3,t,3,q,1} \approx \mathsf{H}^b_{3,t,3,q,2}$ by the function-hiding property of IPFE.

- Hybrid $\mathsf{H}^b_{3,t,3,q,3}$ proceeds identically to $\mathsf{H}^b_{3,t,3,q,2}$, except the labels $\ell_{t,i,j,\mathbf{W},q}$ (for all $i, j, \mathbf{W}$) hardwired in $\mathbf{u}_{t,i,j,\mathbf{W}}[\mathsf{sim}^{\mathsf{comp}}]$ become *random and hardwired*. The indistinguishability between $\mathsf{H}^b_{3,t,3,q,2}$ and $\mathsf{H}^b_{3,t,3,q,3}$ reduces to the DDH assumption in G_1.

- Hybrid $\mathsf{H}^b_{3,t,3,q,4}$ proceeds identically to $\mathsf{H}^b_{3,t,3,q,2}$, except the labels $\ell_{t,i,j,\mathbf{W},q}$ (for all $i, j, \mathbf{W}$) hardwired in $\mathbf{u}_{t,i,j,\mathbf{W}}[\mathsf{sim}^{\mathsf{comp}}]$ are changed to be pseudorandom $\mathbf{s}_{\mathrm{x}}[(t, i, j, \mathbf{W})]\mathbf{s}_{\mathrm{f}}[q]$, which makes them *simulated and hardwired*. The indistinguishability between $\mathsf{H}^b_{3,t,3,q,2}$ and $\mathsf{H}^b_{3,t,3,q,3}$ again reduces to the DDH assumption in G_1.

hybrid	vector	$\mathsf{rand}, \mathsf{rand}^{\mathsf{comp}}, \mathsf{acc}, \mathsf{tb}_\tau, \mathsf{tb}_\tau^{\mathsf{comp}}$	$\mathsf{rand}^{\mathsf{temp}}, \mathsf{rand}^{\mathsf{temp,comp}}, \mathsf{acc}^{\mathsf{temp}}, \mathsf{tb}_\tau^{\mathsf{temp}}, \mathsf{tb}_\tau^{\mathsf{temp,comp}}$	sim	$\mathsf{sim}^{\mathsf{temp}}$	$\mathsf{sim}^{\mathsf{comp}}$
$\mathsf{H}^b_{3,t,3,q,1}$	$\mathbf{v}_{q'<q}$	normal	$0,0,0,0,0$	$\mathbf{s}_{\mathrm{f}}[q']$	$\mathbf{s}_{\mathrm{f}}[q']$	0
	$\mathbf{v}_{q}$	normal	normal	$\mathbf{s}_{\mathrm{f}}[q]$	0	0
	$\mathbf{v}_{q'>q}$	normal	normal	$\mathbf{s}_{\mathrm{f}}[q']$	0	0
	$\mathbf{u}_{t'<t,i,j,\mathbf{W}}$	$0,0,0,0,0$	$0,0,0,0,0$	$\mathbf{s}_{\mathrm{x}}[(t',i',j',\mathbf{W}')]$	0	0
	$\mathbf{u}_{t,i,j,\mathbf{W}}$	$0,0,0,0,0$	normal	0	$\mathbf{s}_{\mathrm{x}}[(t,i,j,\mathbf{W})]$	0
	$\mathbf{u}_{t'>t,i,j,\mathbf{W}}$	normal	$0,0,0,0,0$	0	0	0
$\mathsf{H}^b_{3,t,3,q,2}$	$\mathbf{v}_{q'<q}$	✗ $\mathbf{r}_{\mathrm{f}}[q]$	$0,0,0,0,0$	$\mathbf{s}_{\mathrm{f}}[q']$	$\mathbf{s}_{\mathrm{f}}[q']$	0
	$\mathbf{v}_{q}$	✗ $\mathbf{r}_{\mathrm{f}}[q]$	$0,0,0,0,0$	0	0	1
	$\mathbf{v}_{q'>q}$	✗ $\mathbf{r}_{\mathrm{f}}[q]$	✗ $\mathbf{r}_{\mathrm{f}}[q]$	$\mathbf{s}_{\mathrm{f}}[q']$	0	0
	$\mathbf{u}_{t'<t,i,j,\mathbf{W}}$	$0,0,0,0,0$	$0,0,0,0,0$	$\mathbf{s}_{\mathrm{x}}[(t',i',j',\mathbf{W}')]$	0	$\mathbf{s}_{\mathrm{x}}[(t',i',j',\mathbf{W}')]\mathbf{s}_{\mathrm{f}}[q]$
	$\mathbf{u}_{t,i,j,\mathbf{W}}$	$0,0,0,0,0$	✓ $\mathbf{r}_{\mathrm{f}}[q]$	0	$\mathbf{s}_{\mathrm{x}}[(t,i,j,\mathbf{W})]$	honest $\ell_{t,i,j,\mathbf{W},q}=$ $-\mathbf{r}_{\mathrm{x}}[(t-1,i,j,\mathbf{W})]\mathbf{r}_{\mathrm{f}}[q]+\cdots$
	$\mathbf{u}_{t'>t,i,j,\mathbf{W}}$	✓ $\mathbf{r}_{\mathrm{f}}[q]$	$0,0,0,0,0$	0	0	0
$\mathsf{H}^b_{3,t,3,q,3}$	$\mathbf{v}_{q'<q}$	✗ $\mathbf{r}_{\mathrm{f}}[q]$	$0,0,0,0,0$	$\mathbf{s}_{\mathrm{f}}[q']$	$\mathbf{s}_{\mathrm{f}}[q']$	0
	$\mathbf{v}_{q}$	✗ $\mathbf{r}_{\mathrm{f}}[q]$	$0,0,0,0,0$	0	0	1
	$\mathbf{v}_{q'>q}$	✗ $\mathbf{r}_{\mathrm{f}}[q]$	✗ $\mathbf{r}_{\mathrm{f}}[q]$	$\mathbf{s}_{\mathrm{f}}[q']$	0	0
	$\mathbf{u}_{t'<t,i,j,\mathbf{W}}$	$0,0,0,0,0$	$0,0,0,0,0$	$\mathbf{s}_{\mathrm{x}}[(t',i',j',\mathbf{W}')]$	0	$\mathbf{s}_{\mathrm{x}}[(t',i',j',\mathbf{W}')]\mathbf{s}_{\mathrm{f}}[q]$
	$\mathbf{u}_{t,i,j,\mathbf{W}}$	$0,0,0,0,0$	✓ $\mathbf{r}_{\mathrm{f}}[q]$	0	$\mathbf{s}_{\mathrm{x}}[(t,i,j,\mathbf{W})]$	$\ell_{t,i,j,\mathbf{W},q} \xleftarrow{\$} \mathbb{Z}_p$
	$\mathbf{u}_{t'>t,i,j,\mathbf{W}}$	✓ $\mathbf{r}_{\mathrm{f}}[q]$	$0,0,0,0,0$	0	0	0
$\mathsf{H}^b_{3,t,3,q,4}$	$\mathbf{v}_{q'<q}$	✗ $\mathbf{r}_{\mathrm{f}}[q]$	$0,0,0,0,0$	$\mathbf{s}_{\mathrm{f}}[q']$	$\mathbf{s}_{\mathrm{f}}[q']$	0
	$\mathbf{v}_{q}$	✗ $\mathbf{r}_{\mathrm{f}}[q]$	$0,0,0,0,0$	0	0	1
	$\mathbf{v}_{q'>q}$	✗ $\mathbf{r}_{\mathrm{f}}[q]$	✗ $\mathbf{r}_{\mathrm{f}}[q]$	$\mathbf{s}_{\mathrm{f}}[q']$	0	0
	$\mathbf{u}_{t'<t,i,j,\mathbf{W}}$	$0,0,0,0,0$	$0,0,0,0,0$	$\mathbf{s}_{\mathrm{x}}[(t',i',j',\mathbf{W}')]$	0	$\mathbf{s}_{\mathrm{x}}[(t',i',j',\mathbf{W}')]\mathbf{s}_{\mathrm{f}}[q]$
	$\mathbf{u}_{t,i,j,\mathbf{W}}$	$0,0,0,0,0$	✓ $\mathbf{r}_{\mathrm{f}}[q]$	0	$\mathbf{s}_{\mathrm{x}}[(t,i,j,\mathbf{W})]$	simulated $\ell_{t,i,j,\mathbf{W},q}=$ $\mathbf{s}_{\mathrm{x}}[(t,i,j,\mathbf{W})]\mathbf{s}_{\mathrm{f}}[q]$
	$\mathbf{u}_{t'>t,i,j,\mathbf{W}}$	✓ $\mathbf{r}_{\mathrm{f}}[q]$	$0,0,0,0,0$	0	0	0
$\mathsf{H}^b_{3,t,3,q,5} \equiv \mathsf{H}^b_{3,t,3,q+1,1}$	$\mathbf{v}_{q'<q}$	normal	$0,0,0,0,0$	$\mathbf{s}_{\mathrm{f}}[q']$	$\mathbf{s}_{\mathrm{f}}[q']$	0
	$\mathbf{v}_{q}$	normal	$0,0,0,0,0$	$\mathbf{s}_{\mathrm{f}}[q]$	$\mathbf{s}_{\mathrm{f}}[q]$	0
	$\mathbf{v}_{q'>q}$	normal	normal	$\mathbf{s}_{\mathrm{f}}[q']$	0	0
	$\mathbf{u}_{t'<t,i,j,\mathbf{W}}$	$0,0,0,0,0$	$0,0,0,0,0$	$\mathbf{s}_{\mathrm{x}}[(t',i',j',\mathbf{W}')]$	0	0
	$\mathbf{u}_{t,i,j,\mathbf{W}}$	$0,0,0,0,0$	normal	0	$\mathbf{s}_{\mathrm{x}}[(t,i,j,\mathbf{W})]$	0
	$\mathbf{u}_{t'>t,i,j,\mathbf{W}}$	normal	$0,0,0,0,0$	0	0	0

Figure 9: The inner loop hybrids in the security proof of 1-ABE for L for the case where the ciphertext challenge comes after the secret key query (notes).

For brevity, $\mathbf{u}_{\text{init}}$ and $\mathbf{v}_{\text{init}}$ are suppressed. The reversely sampled ℓ_{init} is hardwired in $\mathbf{u}_{\text{init}}$, and is only needed (and can only be computed so by the reduction) in the exponent of G_1:

$$[\![\ell_{\text{init}}]\!]_1 \leftarrow \mathsf{RevSamp}\big((M, 1^N, 1^T, 1^{2^S}), \mathbf{x}, [\![0]\!]_1, ([\![\ell_{t,z}]\!]_1)_{t\in[T+1],z\in\mathcal{C}_{M,N,S}}\big).$$

The "normal" in...	$\mathsf{rand}, \mathsf{rand}^{\mathsf{temp}}$	$\mathsf{acc}, \mathsf{acc}^{\mathsf{temp}}$	$\mathsf{tb}_\tau, \mathsf{tb}_\tau^{\mathsf{temp}}$	$\mathsf{rand}^{\mathsf{comp}}, \mathsf{rand}^{\mathsf{temp,comp}}$, $\mathsf{tb}_\tau^{\mathsf{comp}}, \mathsf{tb}_\tau^{\mathsf{temp,comp}}$
$\mathbf{v}_q$:	$-\mathbf{r}_{\mathrm{f}}[q]$	$\mu^0\mathbf{y}_{\mathrm{acc}}[q]$	$(\mathbf{M}_\tau\mathbf{r}_{\mathrm{f}})[q]$	0
$t' \le T, \mathbf{u}_{t',i,j,\mathbf{W}}$:	$\mathbf{r}_{\mathrm{x}}[t'-1,i,j,\mathbf{W}]$	0	$\mathfrak{c}(\mathbf{x};t',i,j,\mathbf{W})$	0
$\mathbf{u}_{T+1,i,j,\mathbf{W}}$:	$\mathbf{r}_{\mathrm{x}}[T,i,j,\mathbf{W}]$	1	0	0

The compensation ($\boldsymbol{\times}$ $\mathbf{r}_{\mathrm{f}}[q]$, $\boldsymbol{\checkmark}$ $\mathbf{r}_{\mathrm{f}}[q]$) components in...

	$\mathsf{rand}^?$	$\mathsf{rand}^{?,\mathsf{comp}}$	$\mathsf{acc}^?$	$\mathsf{tb}_\tau^?$	$\mathsf{tb}_\tau^{?,\mathsf{comp}}$
$q' \neq q, \mathbf{v}_{q'}$:	$-\mathbf{r}_{\mathrm{f}}[q']$	0	$\mu^0\mathbf{y}_{\mathrm{acc}}[q']$	$(\mathbf{M}_\tau(\mathbf{r}_{\mathrm{f}}-\mathbf{r}_{\mathrm{f}}[q]\mathbf{e}_q))[q']$	$(\mathbf{M}_\tau\mathbf{e}_q)[q']$
$\mathbf{v}_q$:	0	-1	$\mu^0\mathbf{y}_{\mathrm{acc}}[q\,]$	$(\mathbf{M}_\tau(\mathbf{r}_{\mathrm{f}}-\mathbf{r}_{\mathrm{f}}[q]\mathbf{e}_q))[q\,]$	$(\mathbf{M}_\tau\mathbf{e}_q)[q\,]$
when $t \le T$					
$\mathbf{u}_{t\ ,i,j,\mathbf{W}}$:	$\mathbf{r}_{\mathrm{x}}[(t\ -1,i,j,\mathbf{W})]$	0	0	$\mathfrak{c}_\tau(\mathbf{x};t\ ,i,j,\mathbf{W};\mathbf{r}_{\mathrm{x}})$	$\mathfrak{c}_\tau(\mathbf{x};t\ ,i,j,\mathbf{W};\mathbf{r}_{\mathrm{x}})\mathbf{r}_{\mathrm{f}}[q]$
$t' < t \le T, \mathbf{u}_{t',i,j,\mathbf{W}}$:	$\mathbf{r}_{\mathrm{x}}[(t'-1,i,j,\mathbf{W})]$	$\mathbf{r}_{\mathrm{x}}[(t'-1,i,j,\mathbf{W})]\mathbf{r}_{\mathrm{f}}[q]$	0	$\mathfrak{c}_\tau(\mathbf{x};t',i,j,\mathbf{W};\mathbf{r}_{\mathrm{x}})$	$\mathfrak{c}_\tau(\mathbf{x};t',i,j,\mathbf{W};\mathbf{r}_{\mathrm{x}})\mathbf{r}_{\mathrm{f}}[q]$
$\mathbf{u}_{T+1,i,j,\mathbf{W}}$:	$\mathbf{r}_{\mathrm{x}}[(T,i,j,\mathbf{W})]$	$\mathbf{r}_{\mathrm{x}}[(T,i,j,\mathbf{W})]\mathbf{r}_{\mathrm{f}}[q]$	1	0	0
when $t = T+1$					
$\mathbf{u}_{T+1,i,j,\mathbf{W}}$:	$\mathbf{r}_{\mathrm{x}}[(T,i,j,\mathbf{W})]$	0	1	0	0

In the table above, "?" is either nothing or "temp", i.e., if the values are set in both non-temporary slots and temporary slots, they are the same. Note that $\mathbf{r}_{\mathrm{f}} - \mathbf{r}_{\mathrm{f}}[q]\mathbf{e}_q$ is simply $\mathbf{r}_{\mathrm{f}}$ with its q^{th} entry changed to 0, whence $\mathbf{r}_{\mathrm{f}}[q]$ does not appear. The compensation is governed by the following identity:

$$\begin{aligned}
\ell_{t',i,j,\mathbf{W},q'} &= -\mathbf{r}_{\mathrm{x}}[(t'-1,i,j,\mathbf{W})]\mathbf{r}_{\mathrm{f}}[q'] + \sum_{\tau\in\mathcal{T}} \mathfrak{c}_\tau(\mathbf{x};t',i,j,\mathbf{W};\mathbf{r}_{\mathrm{x}})(\mathbf{M}_\tau(\mathbf{r}_{\mathrm{f}} - \mathbf{r}_{\mathrm{f}}[q]\mathbf{e}_q + \mathbf{r}_{\mathrm{f}}[q]\mathbf{e}_q))[q'] \\
&= -\mathbf{r}_{\mathrm{x}}[(t'-1,i,j,\mathbf{W})]\mathbf{r}_{\mathrm{f}}[q'] + \sum_{\tau\in\mathcal{T}} \mathfrak{c}_\tau\big(\mathbf{x};t',i,j,\mathbf{W};\mathbf{r}_{\mathrm{x}}\big)\cdot\big(\mathbf{M}_\tau(\mathbf{r}_{\mathrm{f}} - \mathbf{r}_{\mathrm{f}}[q]\mathbf{e}_q)\big)[q'] \\
&\qquad + \sum_{\tau\in\mathcal{T}} \mathfrak{c}_\tau(\mathbf{x};t',i,j,\mathbf{W};\mathbf{r}_{\mathrm{x}})\mathbf{r}_{\mathrm{f}}[q]\cdot(\mathbf{M}_\tau\mathbf{e}_q)[q'].
\end{aligned}$$

In this iteration, the *modes* of $\ell_{t',i,j,\mathbf{W},q'}$ are...

	$q' < q$	$q' = q$	$q' > q$
$t' < t$	S	S $\to$ SC $\to$ S	S
$t' = t$	ST	HT $\to$ HW $\to$ RW $\to$ SW $\to$ ST	HT $\to$ HCT $\to$ HT
$t' > t$	H $\to$ HC $\to$ H	H $\to$ HC $\to$ H	H $\to$ HC $\to$ H

The shorthands are **H**onest, **R**andom, **S**imulated, **C**ompensated, hard**W**ired, **T**emporary.
For three-mode chains, the middle one spans $\mathsf{H}^b_{3,t,3,q,2\sim4}$.
The net effect is that $\ell_{t,i,j,\mathbf{W},q}$'s change from *honest and temporary* to *simulated and temporary*.

Figure 9 (continued) — Notes.

- Hybrid $\mathsf{H}^b_{3,t,3,q,5}$ proceeds identically to $\mathsf{H}^b_{3,t,3,q,4}$, except that all occurrences of $\mathbf{r}_\mathrm{f}[q]$ and $\mathbf{s}_\mathrm{f}[q]$ are moved back to $\mathbf{v}_{q'}$'s (including putting $\mathbf{s}_\mathrm{f}[q]$ back to $\mathbf{v}_q[\mathsf{sim}]$) — compensation is undone — and that some clean-up work is done to prepare for the next iteration. More specifically, the clean-up work involves clearing the values of $\mathbf{v}_q, \mathbf{u}_{t,i,j,\mathbf{W}}$ at $\mathsf{sim}^{\mathsf{comp}}$ for all $i, j, \mathbf{W}$ and putting $\mathbf{s}_\mathrm{f}[q]$ into $\mathbf{v}_q[\mathsf{sim}^{\mathsf{temp}}]$. As a result, all the labels lose their *compensated* mode, and the labels $\ell_{t,i,j,\mathbf{W},q}$ for all $i, j, \mathbf{W}$ become *simulated and temporary*. Note that $\mathsf{H}^b_{3,t,3,q,5} \equiv \mathsf{H}^b_{3,t,3,q+1,1}$.

Lastly, observe that $\mathsf{H}^b_{3,t,3,1,1} \equiv \mathsf{H}^b_{3,t,2}$ and $\mathsf{H}^b_{3,t,3,Q,5} \equiv \mathsf{H}^b_{3,t,4}$. By a hybrid argument over the inner loop hybrids, we have $\mathsf{H}^b_{3,t,2} \approx \mathsf{H}^b_{3,t,4}$.

Hybriding over the outer loop hybrids yields $\mathsf{H}^b_2 \approx \mathsf{H}^b_4$ in Case 2. □

Remarks. For the version based on MDDH_k, the labels are simulated as

$$\ell_{t,i,j,\mathbf{W},q} = (\mathbf{S}_\mathrm{x}^\intercal \mathbf{S}_\mathrm{f})[(t,i,j,\mathbf{W}),q] \quad \text{for} \quad \mathbf{S}_\mathrm{x} \xleftarrow{\$} \mathbb{Z}_p^{[k]\times([T+1]\times[N]\times[S]\times\{0,1\}^S)},\ \mathbf{S}_\mathrm{f} \xleftarrow{\$} \mathbb{Z}_p^{k\times Q}.$$

The proof for Case 1 goes by invoking MDDH_k instead of DDH. We discuss the change in the proof for Case 2. At some point, we need to switch the randomizers of the labels with some specific t, q to truly random. Invoking MDDH_k apparently requires completely removing the q^{th} column of $\mathbf{R}_\mathrm{f}$, yet this column is used to compute the randomizers for the labels with $q' = q$ that are still honest. We got away with this when assuming DDH because both distributions give us access to $\mathbf{r}_\mathrm{f}[q]$ (a convenient shortcut), i.e.,

$$\begin{aligned} & \left[\!\left[\mathbf{r}_\mathrm{x}[(t-1,\sqcup,\sqcup,\sqcup)], \mathbf{r}_\mathrm{f}[q], \mathbf{r}_\mathrm{x}[(t-1,\sqcup,\sqcup,\sqcup)]\mathbf{r}_\mathrm{f}[q] \right]\!\right]_1 \\ (\text{DDH}) \quad \approx & \left[\!\left[\mathbf{r}_\mathrm{x}[(t-1,\sqcup,\sqcup,\sqcup)], \mathbf{r}_\mathrm{f}[q], \quad \mathbf{r}[(t-1,\sqcup,\sqcup,\sqcup,q)] \right]\!\right]_1. \end{aligned}$$

A naïve attempt to generalize the proof to work with MDDH_k is to say

$$\begin{aligned} & \left[\!\left[\mathbf{R}_\mathrm{x}[\sqcup,(t-1,\sqcup,\sqcup,\sqcup)], \mathbf{R}_\mathrm{f}[\sqcup,q], (\mathbf{R}_\mathrm{x}[\sqcup,(t-1,\sqcup,\sqcup,\sqcup)])^\intercal \mathbf{R}_\mathrm{f}[\sqcup,q] \right]\!\right]_1 \\ \overset{??}{\approx} & \left[\!\left[\mathbf{R}_\mathrm{x}[\sqcup,(t-1,\sqcup,\sqcup,\sqcup)], \mathbf{R}_\mathrm{f}[\sqcup,q], \qquad \mathbf{r}[(t-1,\sqcup,\sqcup,\sqcup),q] \right]\!\right]_1, \end{aligned}$$

which does *not* follow by MDDH_k (and is false for symmetric pairing groups). The correct proof uses

$$\begin{aligned} & \left[\!\left[\mathbf{R}_\mathrm{x}[\sqcup,(\sqcup,\sqcup,\sqcup,\sqcup)], \overbrace{\{(\mathbf{R}_\mathrm{x}[\sqcup,(t'-1,\sqcup,\sqcup,\sqcup)])^\intercal \mathbf{R}_\mathrm{f}[\sqcup,q]\}_{t'>t}}^{\substack{\text{in } \mathbf{u}_{t'',\sqcup,\sqcup,\sqcup} \text{ at } \mathsf{rand}^{\mathsf{comp}},\ \mathsf{tb}_\tau^{\mathsf{comp}}\text{'s} \\ \text{and in } \mathbf{u}_{t,\sqcup,\sqcup,\sqcup} \text{ at } \mathsf{tb}_\tau^{\mathsf{temp,comp}}\text{'s}}}, \overbrace{(\mathbf{R}_\mathrm{x}[\sqcup,(t-1,\sqcup,\sqcup,\sqcup)])^\intercal \mathbf{R}_\mathrm{f}[\sqcup,q]}^{\text{in } \mathbf{u}_{t,\sqcup,\sqcup,\sqcup} \text{ at } \mathsf{sim}^{\mathsf{comp}}} \right]\!\right]_1 \\ (\mathrm{MDDH}_k) \approx & \left[\!\left[\mathbf{R}_\mathrm{x}[\sqcup,(\sqcup,\sqcup,\sqcup,\sqcup)], \{ \quad \mathbf{r}[(t'-1,\sqcup,\sqcup,\sqcup),q] \quad \}_{t'>t}, \quad \mathbf{r}[(t-1,\sqcup,\sqcup,\sqcup),q] \right]\!\right]_1 \\ (\mathrm{MDDH}_k) \approx & \left[\!\left[\mathbf{R}_\mathrm{x}[\sqcup,(\sqcup,\sqcup,\sqcup,\sqcup)], \{(\mathbf{R}_\mathrm{x}[\sqcup,(t'-1,\sqcup,\sqcup,\sqcup)])^\intercal \mathbf{R}_\mathrm{f}[\sqcup,q]\}_{t'>t}, \quad \mathbf{r}[(t-1,\sqcup,\sqcup,\sqcup),q] \right]\!\right]_1. \end{aligned}$$

In other words, we first switch all the *randomizers* of the labels with $q' = q$ to truly random, then switch back the *randomizers* of the labels with $t' > t, q' = q$ to pseudorandom (the randomizers for the labels with $t' < t$ are already gone in that iteration, because those labels are already simulated). A similar strategy is used for $\mathbf{S}_\mathrm{x}, \mathbf{S}_\mathrm{f}$ when we switch the truly random *labels* to pseudorandom simulated ones.

7.3 KP-ABE for L

We use the same technique in Section 6.3 to convert 1-ABE for L into a full-fledged ABE for L. The only difference is that we will need *two* copies of the indices in the private slot.

In the security proof of KP-ABE for ABPs, the private slot can be thought as a compartment isolating the interaction between all the secret keys and *the challenge ciphertext* from the other ciphertexts (which can be generated by the adversary using the master public key in the public slot). All the secret keys can share the same private slot because the security of 1-ABE for ABPs is unaffected by the extra keys. However, the security of 1-ABE for L does not automatically extend to the multi-key setting, and we need an extra copy of the indices to further isolate the interaction between the challenge ciphertext and *the secret key being modified* from the other secret keys.

Construction 37 (KP-ABE for L)**.** We describe the construction for any fixed value of the security parameter λ and suppress the appearance of λ below for simplicity of notations. Let the function class be

$$\mathcal{F} = \left\{ M|_{N,T,S} : \mathbb{Z}_p^N \to \mathbb{Z}_p \,\middle|\, M \in \mathsf{TM}, N, T, S \geq 1, p \text{ prime} \right\}.$$

Let $(\mathsf{Garble}, \mathsf{Eval})$ be the AKGS for $\mathcal{F}$ in Construction 30, $\mathcal{G}$ pairing groups of order p, and $(\mathsf{IPFE.Setup}, \mathsf{IPFE.KeyGen}, \mathsf{IPFE.Enc}, \mathsf{IPFE.Dec})$ a slotted IPFE based on $\mathcal{G}$. We construct an ABE scheme for the following singleton predicate space $\mathcal{P}$:

$$\begin{aligned}
&X = \left\{ (\mathbf{x}, 1^T, 1^{2^S}) \,\middle|\, \mathbf{x} \in \{0,1\}^N \text{ for some } N \geq 1,\ T, S \geq 1 \right\}, \quad Y = \mathsf{TM}, \\
&P : X \times Y \to \{0,1\},\ \left((\mathbf{x}, 1^T, 1^{2^S}), M \right) \mapsto M|_{N,T,S}(\mathbf{x}) \text{ for } \mathbf{x} \in \{0,1\}^N, \\
&\mathcal{P} = \{P\}.
\end{aligned}$$

The ABE scheme $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ operates as follows:

- $\mathsf{Setup}(P)$ takes the only predicate as input. It generates IPFE master public/secret key pair $(\mathsf{msk}, \mathsf{mpk}) \xleftarrow{\$} \mathsf{IPFE.Setup}(\mathfrak{s}_{\text{pub}}, \mathfrak{s}_{\text{priv}})$ for the following slots:

  $$\begin{aligned}
  \mathfrak{s}_{\text{pub}} &= \left\{ \mathsf{pad}^{\mathsf{ct}}, \mathsf{pad}^{\mathsf{sk}}, \mathsf{init}^{\text{pub}}, \mathsf{rand}^{\text{pub}}, \mathsf{acc}^{\text{pub}} \right\} \cup \left\{ \mathsf{tb}_\tau^{\text{pub}} \,\middle|\, \tau \in \mathcal{T} \right\}, \\
  \mathfrak{s}_{\text{copy}} &= \left\{ \mathsf{init}^{\text{copy}}, \mathsf{rand}^{\text{copy}}, \mathsf{acc}^{\text{copy}} \right\} \cup \left\{ \mathsf{tb}_\tau^{\text{copy}} \,\middle|\, \tau \in \mathcal{T} \right\}, \\
  \mathfrak{s}_{\text{priv}} &= \mathfrak{s}_{\text{copy}} \cup \mathfrak{s}_{\text{1-ABE}} \cup \left\{ \mathsf{pad}^{\text{copy}}, \mathsf{pad}^{\mathsf{temp}} \right\},
  \end{aligned}$$

  where $\mathfrak{s}_{\text{1-ABE}}$ is the index set defined in Construction 33. The algorithm returns $(\mathsf{mpk}, \mathsf{msk})$ as the master public/secret key pair.

  Note: *The indices* $\mathsf{pad}^{\mathsf{ct}}, \mathsf{pad}^{\mathsf{sk}}$ *are used to compute* $h + \mu$ *in the exponent, where* h *(resp.* μ*) is the random pad in* ct *(resp.* sk*) embedded at* $\mathsf{pad}^{\mathsf{ct}}$ *(resp.* $\mathsf{pad}^{\mathsf{sk}}$*). The values in* $\mathfrak{s}_{\text{priv}}$ *are set to* 0 *by the honest algorithms and reserved for the security proof. The names of the other indices follow the convention in 1-ABE.*

- $\mathsf{KeyGen}(\mathsf{msk}, M)$ takes the master secret key and a Turing machine $M = (Q, \mathbf{y}_{\text{acc}}, \delta) \in \mathsf{TM} = Y$ as input. It computes the transition blocks $\mathbf{M}_\tau$ for $\tau \in \mathcal{T}$ from δ, samples $\mu \xleftarrow{\$} \mathbb{Z}_p, \mathbf{r}_{\text{f}} \xleftarrow{\$} \mathbb{Z}_p^Q$, and sets the vectors $\mathbf{v}_{\text{pad}}, \mathbf{v}_{\text{init}}$ and $\mathbf{v}_q \in \mathbb{Z}_p^{\mathfrak{s}}$ for $q \in [Q]$ as follows:

vector	$\mathsf{pad}^{\mathsf{ct}}$	$\mathsf{pad}^{\mathsf{sk}}$	$\mathsf{init}^{\text{pub}}$	$\mathsf{rand}^{\text{pub}}$	$\mathsf{acc}^{\text{pub}}$	$\mathsf{tb}_\tau^{\text{pub}}$	in $\mathfrak{s}_{\text{priv}}$
$\mathbf{v}_{\text{pad}}$	1	μ	0	0	0	0	0
$\mathbf{v}_{\text{init}}$	0	0	$\mathbf{r}_{\text{f}}[1]$	0	0	0	
$\mathbf{v}_q$	0	0	0	$-\mathbf{r}_{\text{f}}[q]$	$\mu \mathbf{y}_{\text{acc}}[q]$	$(\mathbf{M}_\tau \mathbf{r}_{\text{f}})[q]$	

The algorithm generates an IPFE secret key for each vector defined above:

$$\begin{aligned}\mathsf{isk}_{\mathrm{pad}} &\xleftarrow{\$} \mathsf{IPFE.KeyGen}(\mathsf{msk}, [\![\mathbf{v}_{\mathrm{pad}}]\!]_2),\\ \mathsf{isk}_{\mathrm{init}} &\xleftarrow{\$} \mathsf{IPFE.KeyGen}(\mathsf{msk}, [\![\mathbf{v}_{\mathrm{init}}]\!]_2),\\ \text{for } q \in [Q]\colon\quad \mathsf{isk}_q &\xleftarrow{\$} \mathsf{IPFE.KeyGen}(\mathsf{msk}, [\![\mathbf{v}_q]\!]_2).\end{aligned}$$

It returns $\mathsf{sk} = (\mathsf{isk}_{\mathrm{pad}}, M, \mathsf{isk}_{\mathrm{init}}, \mathsf{isk}_1, \ldots, \mathsf{isk}_Q)$.

Note: *Same as in 1-ABE for* L, KeyGen *creates the garbling with secrets* $\alpha = \mu, \beta = 0$.

- $\mathsf{Enc}\big(\mathsf{mpk}, (\mathbf{x}, 1^T, 1^{2^S}), g\big)$ takes the master public key, an input $\mathbf{x} \in \{0,1\}^N$ for some $N \geq 1$, time/space complexity bounds $T, S \geq 1$ (encoded as 1^T and 1^{2^S}), and a message $g \in M = G_{\mathrm{T}}$ as input. It samples $h \xleftarrow{\$} \mathbb{Z}_p$, $s \xleftarrow{\$} \mathbb{Z}_p$, $\mathbf{r}_{\mathrm{x}} \xleftarrow{\$} \mathbb{Z}_p^{[0..T]\times[N]\times[S]\times\{0,1\}^S}$ and sets the vectors $\mathbf{u}_{\mathrm{pad}}, \mathbf{u}_{\mathrm{init}}$ and $\mathbf{u}_{t,i,j,\mathbf{W}} \in \mathbb{Z}_p^{\mathfrak{s}_{\mathrm{pub}}}$ for $t \in [T+1], i \in [N], j \in [S], \mathbf{W} \in \{0,1\}^S$ as follows:

vector	$\mathsf{pad}^{\mathsf{ct}}$	$\mathsf{pad}^{\mathsf{sk}}$	$\mathsf{init}^{\mathrm{pub}}$	the other indices
$\mathbf{u}_{\mathrm{pad}}$	h	s	0	0
$\mathbf{u}_{\mathrm{init}}$	0	0	$s\mathbf{r}_{\mathrm{x}}[(0,1,1,\mathbf{0}_S)]$	

vector	$\mathsf{rand}^{\mathrm{pub}}$	$\mathsf{acc}^{\mathrm{pub}}$	$\mathsf{tb}_\tau^{\mathrm{pub}}$	the other indices
$t \leq T$: $\mathbf{u}_{t,i,j,\mathbf{W}}$	$s\mathbf{r}_{\mathrm{x}}[(t-1,i,j,\mathbf{W})]$	0	$s\mathfrak{c}_\tau(\mathbf{x};t,i,j,\mathbf{W};\mathbf{r}_{\mathrm{x}})$	0
$\mathbf{u}_{T+1,i,j,\mathbf{W}}$	$s\mathbf{r}_{\mathrm{x}}[(T,i,j,\mathbf{W})]$	s	0	

The algorithm then generates IPFE ciphertexts for the vectors defined above:

$$\begin{aligned}\mathsf{ict}_{\mathrm{pad}} &\xleftarrow{\$} \mathsf{IPFE.SlotEnc}(\mathsf{mpk}, [\![\mathbf{u}_{\mathrm{pad}}]\!]_1), \qquad \mathsf{ict}_{\mathrm{init}} \xleftarrow{\$} \mathsf{IPFE.SlotEnc}(\mathsf{mpk}, [\![\mathbf{u}_{\mathrm{init}}]\!]_1),\\ \text{for } t \in [T+1], i \in [N], j \in [S], \mathbf{W} \in \{0,1\}^S\colon\quad \mathsf{ict}_{t,i,j,\mathbf{W}} &\xleftarrow{\$} \mathsf{IPFE.SlotEnc}(\mathsf{mpk}, [\![\mathbf{u}_{t,i,j,\mathbf{W}}]\!]_1).\end{aligned}$$

It returns $\mathsf{ct} = \big(g + [\![h]\!]_{\mathrm{T}}, \mathsf{ict}_{\mathrm{pad}}, (\mathbf{x}, T, S), \mathsf{ict}_{\mathrm{init}}, (\mathsf{ict}_{t,i,j,\mathbf{W}})_{t\in[T+1],i\in[N],j\in[S],\mathbf{W}\in\{0,1\}^S}\big)$.

- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$ takes a secret key sk and a ciphertext ct as input. It parses

$$\begin{aligned}&\mathsf{sk} \text{ as } \big(\mathsf{isk}_{\mathrm{pad}}, M, \mathsf{isk}_{\mathrm{init}}, \mathsf{isk}_1, \ldots, \mathsf{isk}_Q\big) \qquad \text{and}\\ &\mathsf{ct} \text{ as } \big([\![z]\!]_{\mathrm{T}}, \mathsf{ict}_{\mathrm{pad}}, (\mathbf{x}, T, S), \mathsf{ict}_{\mathrm{init}}, (\mathsf{ict}_{t,i,j,\mathbf{W}})_{t\in[T+1],i\in[N],j\in[S],\mathbf{W}\in\{0,1\}^S}\big),\end{aligned}$$

where $M = (Q, \mathbf{y}_{\mathrm{acc}}, \delta) \in \mathsf{TM}$ and $\mathbf{x} \in \{0,1\}^N$. The algorithm returns $\perp$ if $M|_{N,T,S}(\mathbf{x}) = 0$. Otherwise, it does the following (recall that $\mathcal{C}_{M,N,S} = [N] \times [S] \times \{0,1\}^S \times [Q]$):

$$\begin{aligned}[\![z']\!]_{\mathrm{T}} &\leftarrow \mathsf{IPFE.Dec}(\mathsf{isk}_{\mathrm{pad}}, \mathsf{ict}_{\mathrm{pad}}), \quad [\![\ell_{\mathrm{init}}]\!]_{\mathrm{T}} \leftarrow \mathsf{IPFE.Dec}(\mathsf{isk}_{\mathrm{init}}, \mathsf{ict}_{\mathrm{init}}),\\ \text{for } t \in [T+1], (i,j,\mathbf{W},q) \in \mathcal{C}_{M,N,S}\colon\quad [\![\ell_{t,i,j,\mathbf{W},q}]\!]_{\mathrm{T}} &\leftarrow \mathsf{IPFE.Dec}(\mathsf{isk}_q, \mathsf{ict}_{t,i,j,\mathbf{W}}),\\ [\![\mu']\!]_{\mathrm{T}} &\leftarrow \mathsf{Eval}\big((M, 1^N, 1^T, 1^{2^S}, p), \mathbf{x}, [\![\ell_{\mathrm{init}}]\!]_{\mathrm{T}}, ([\![\ell_{t,z}]\!]_{\mathrm{T}})_{t\in[T+1],z\in\mathcal{C}_{M,N,S}}\big).\end{aligned}$$

The algorithm returns $[\![z]\!]_{\mathrm{T}} + [\![\mu']\!]_{\mathrm{T}} - [\![z']\!]_{\mathrm{T}}$ as the decrypted message.

Note: *By combining the arguments for the correctness of 1-ABE for* L *(Construction 33) and KP-ABE for ABPs (Construction 26), it is readily verified that* $\mu' = s\mu M|_{N,T,S}(\mathbf{x}) = s\mu$ *and* $z' = h + s\mu$. *Therefore,* $[\![z]\!]_{\mathrm{T}} + [\![\mu']\!]_{\mathrm{T}} - [\![z']\!]_{\mathrm{T}} = g + [\![h]\!]_{\mathrm{T}} + [\![s\mu]\!]_{\mathrm{T}} - [\![h + s\mu]\!]_{\mathrm{T}} = g$, *i.e.,* Dec *correctly recovers the message.*

Theorem 38. *Suppose in Construction 37, the* SXDH *assumption holds in $\mathcal{G}$ and the slotted IPFE is* function-hiding*, then the construction is a secure ABE scheme.*

Proof. Let $\mathcal{A}$ be any efficient adversary. We want to show that the distinguishing advantage of $\mathcal{A}$ against $\mathsf{Exp}^0_{\mathrm{CPA}}$ and $\mathsf{Exp}^1_{\mathrm{CPA}}$ is negligible. Suppose $\mathcal{A}$ makes Φ secret key queries, it receives sk_φ encoding half a garbling for machine M_φ in the public slot and ct encrypting message g_b and encoding half a garbling for input string $\mathbf{x} \in \{0,1\}^N$ and time/space bounds T, S in the public slot, where M_φ's, $\mathbf{x}, T, S$ are chosen adaptively by $\mathcal{A}$ subject to the constraint that $M_\varphi|_{N,T,S}(\mathbf{x}) = 0$ for all $\varphi \in [\Phi]$. The goal of the proof is thus to show the message g_b is computationally hidden.

At a high level, the proof involves three steps, similar to those in the proof of IND-CPA security of KP-ABE for ABPs (Theorem 27):

- First, the garblings (in both ct and sk_φ) as well as the secret key pads μ_φ are removed from $\mathfrak{s}_{\mathrm{pub}}$. Instead, ct embeds in $\mathfrak{s}_{\mathrm{copy}}$ (half of) the garbling without s, and each sk_φ embeds in $\mathfrak{s}_{\mathrm{copy}}$ (half) a garbling generated with pad $\widehat{\mu}^0_\varphi$ and randomness $\widehat{\mathbf{r}}_{\varphi,\mathrm{f}}$ *independent* of those in $\mathfrak{s}_{\mathrm{pub}}$.

- Next, we go through a loop of Φ iterations. In the φ^{th} iteration, we replace $\widehat{\mu}^0_\varphi$ in $\mathbf{v}_{\varphi,\mathrm{pad}}$ by an independent random value $\widehat{\mu}^1_\varphi$ (inconsistent with $\widehat{\mu}^0_\varphi$ used to generate $\mathbf{v}_{\varphi,\mathrm{init}}$ and $\mathbf{v}_{\varphi,q}$'s). However, this is not as simple as the case of KP-ABE for ABPs, because the security of 1-ABE for L (Theorem 34) does not automatically generalize to the case of multiple secret key queries, which can be regarded as a consequence of reusing the randomness in the ciphertext *vectors*. We employ the trick of *temporary* mode to resolve the problem. Roughly speaking, $\mathfrak{s}_{\text{1-ABE}}$ is used for the interaction between sk_φ (the secret key being modified) and ct, while $\mathfrak{s}_{\mathrm{copy}}$ is used for the interaction between all other secret keys and ct. The ciphertext uses two sets of *independent* randomness — $\mathbf{r}_{\mathrm{x}}$ in $\mathfrak{s}_{\mathrm{copy}}$ and $\widehat{\mathbf{r}}_{\mathrm{x}}$ in $\mathfrak{s}_{\text{1-ABE}}$ — so that the proof of Theorem 34 can be invoked in $\mathfrak{s}_{\text{1-ABE}}$.

- When all the secret keys have their pads $\widehat{\mu}^0_\varphi$ replaced by $\widehat{\mu}^1_\varphi$, we move the random pad h from the ciphertext into all the secret keys, by which point h will be perfectly hidden by the pads $\widehat{\mu}^1_\varphi$ and perfectly hide the message.

We start with the first step, which is the first step in Theorem 27 with DDH instead of MDDH:

- Hybrid H^b_0 proceeds identically to $\mathsf{Exp}^b_{\mathrm{CPA}}$, where the ict's (in ct) are generated using $\mathsf{SlotEnc}$. In Figure 10, this is depicted as "$\perp$" in the $\mathbf{u}$'s.

- Hybrid H^b_1 proceeds identically to H^b_0, except that the ict's are generated using Enc with $\mathbf{u}|_{\mathfrak{s}_{\mathrm{priv}}}$ set to $\mathbf{0}$. By the slot-mode correctness of the slotted IPFE, we have $\mathsf{H}^b_1 \equiv \mathsf{H}^b_0$.

- Hybrid H^b_2 proceeds identically to H^b_1, except with how the (secret key) pads and the labels are computed (as inner products) is changed — instead of multiplying with s in the ciphertext, s is multiplied in the secret key (which prepares the secret key randomness to be rerandomized):

  - In the ciphertext vectors, values at $\mathsf{pad}^{\mathsf{sk}}, \mathsf{init}^{\mathrm{pub}}, \mathsf{rand}^{\mathrm{pub}}, \mathsf{acc}^{\mathrm{pub}}$ and $\mathsf{tb}^{\mathrm{pub}}_\tau$'s (in the public slot) are *moved* to $\mathsf{pad}^{\mathrm{copy}}, \mathfrak{s}_{\mathrm{copy}}$ (in the private slot), and the multiplier s is removed.
  - In the secret key vectors, values at $\mathsf{pad}^{\mathsf{sk}}, \mathsf{init}^{\mathrm{pub}}, \mathsf{rand}^{\mathrm{pub}}, \mathsf{acc}^{\mathrm{pub}}$ and $\mathsf{tb}^{\mathrm{pub}}_\tau$'s (in the public slot) are *copied* to $\mathsf{pad}^{\mathrm{copy}}, \mathfrak{s}_{\mathrm{copy}}$ (in the private slot), and s is multiplied to the copy in the private slot.

hybrid	vector	$\mathsf{pad}^{\mathsf{ct}}$, $\mathsf{pad}^{\mathsf{sk}}$	$\mathsf{init}^{\mathrm{pub}}$,$\mathsf{rand}^{\mathrm{pub}}$, $\mathsf{acc}^{\mathrm{pub}}$,$\mathsf{tb}_\tau^{\mathrm{pub}}$	$\mathsf{pad}^{\mathrm{copy}}$	in $\mathfrak{S}_{\mathrm{copy}}$
$\mathsf{H}_0^b \equiv \mathsf{Exp}_{\mathrm{CPA}}^b$	$\mathbf{v}_{\varphi,\mathrm{pad}}$	$1,\ \mu_\varphi$		0	
	$\mathbf{v}_{\varphi,\mathrm{init}}, \mathbf{v}_{\varphi,q}$		$\propto (\mu_\varphi, \mathbf{r}_{\varphi,\mathrm{f}})$		0
	$\mathbf{u}_{\mathrm{pad}}$	$h,\ s$		$\perp$	
	$\mathbf{u}_{\mathrm{init}}, \mathbf{u}_{t,i,j,\mathbf{W}}$		$\propto (s, s\mathbf{r}_{\mathrm{x}})$		$\perp$
H_1^b	$\mathbf{v}_{\varphi,\mathrm{pad}}$	$1,\ \mu_\varphi$		0	
	$\mathbf{v}_{\varphi,\mathrm{init}}, \mathbf{v}_{\varphi,q}$		$\propto (\mu_\varphi, \mathbf{r}_{\varphi,\mathrm{f}})$		0
	$\mathbf{u}_{\mathrm{pad}}$	$h,\ s$		0	
	$\mathbf{u}_{\mathrm{init}}, \mathbf{u}_{t,i,j,\mathbf{W}}$		$\propto (s, s\mathbf{r}_{\mathrm{x}})$		0
H_2^b	$\mathbf{v}_{\varphi,\mathrm{pad}}$	$1,\ \mu_\varphi$		$s\mu_\varphi$	
	$\mathbf{v}_{\varphi,\mathrm{init}}, \mathbf{v}_{\varphi,q}$		$\propto (\mu_\varphi, \mathbf{r}_{\varphi,\mathrm{f}})$		$\propto (s\mu_\varphi, s\mathbf{r}_{\varphi,\mathrm{f}})$
	$\mathbf{u}_{\mathrm{pad}}$	$h,\ 0$		1	
	$\mathbf{u}_{\mathrm{init}}, \mathbf{u}_{t,i,j,\mathbf{W}}$		0		$\propto (1, \mathbf{r}_{\mathrm{x}})$
$\mathsf{H}_3^b \equiv \mathsf{H}_{4,1}^b$	$\mathbf{v}_{\varphi,\mathrm{pad}}$	$1,\ \mu_\varphi$		$\widehat{\mu}_\varphi^0$	
	$\mathbf{v}_{\varphi,\mathrm{init}}, \mathbf{v}_{\varphi,q}$		$\propto (\mu_\varphi, \mathbf{r}_{\varphi,\mathrm{f}})$		$\propto (\widehat{\mu}_\varphi^0, \widehat{\mathbf{r}}_{\varphi,\mathrm{f}})$
	$\mathbf{u}_{\mathrm{pad}}$	$h,\ 0$		1	
	$\mathbf{u}_{\mathrm{init}}, \mathbf{u}_{t,i,j,\mathbf{W}}$		0		$\propto (1, \mathbf{r}_{\mathrm{x}})$
$\mathsf{H}_{4,1\sim\Phi+1}^b$			.....		
$\mathsf{H}_5^b \equiv \mathsf{H}_{4,\Phi+1}^b$	$\mathbf{v}_{\varphi,\mathrm{pad}}$	$1,\ \mu_\varphi$		$\widehat{\mu}_\varphi^1 \xleftarrow{\$} \mathbb{Z}_p$	
	$\mathbf{v}_{\varphi,\mathrm{init}}, \mathbf{v}_{\varphi,q}$		$\propto (\mu_\varphi, \mathbf{r}_{\varphi,\mathrm{f}})$		$\propto (\widehat{\mu}_\varphi^0, \widehat{\mathbf{r}}_{\varphi,\mathrm{f}})$
	$\mathbf{u}_{\mathrm{pad}}$	$h,\ 0$		1	
	$\mathbf{u}_{\mathrm{init}}, \mathbf{u}_{t,i,j,\mathbf{W}}$		0		$\propto (1, \mathbf{r}_{\mathrm{x}})$
H_6^b	$\mathbf{v}_{\varphi,\mathrm{pad}}$	$1,\ \mu_\varphi$		$h + \widehat{\mu}_\varphi^1$	
	$\mathbf{v}_{\varphi,\mathrm{init}}, \mathbf{v}_{\varphi,q}$		$\propto (\mu_\varphi, \mathbf{r}_{\varphi,\mathrm{f}})$		$\propto (\widehat{\mu}_\varphi^0, \widehat{\mathbf{r}}_{\varphi,\mathrm{f}})$
	$\mathbf{u}_{\mathrm{pad}}$	$0,\ 0$		1	
	$\mathbf{u}_{\mathrm{init}}, \mathbf{u}_{t,i,j,\mathbf{W}}$		0		$\propto (1, \mathbf{r}_{\mathrm{x}})$

For brevity, the vectors for computing the labels are not spelled out.
The shorthand "$\propto \mathbf{z}$" means that the components there are linear in $\mathbf{z}$ and efficiently computable given $\mathbf{z}$ in the exponent, and that there is only one natural way of computing them (cf. Construction 37).

Figure 10: The first/last few hybrids in the proof of IND-CPA security of our KP-ABE scheme for L.

The hybrid is illustrated in Figure 10. Since the inner products remain the same and the secret key vectors have their values in the public slot unchanged, we have $\mathsf{H}_2^b \approx \mathsf{H}_1^b$ by the function-hiding property of IPFE. An alternative view of this hybrid is that in each secret key sk_φ, two (half) garblings are embedded, one generated for μ_φ using randomness $\mathbf{r}_{\varphi,\mathrm{f}}$ in the public slot, the other generated for $s\mu_\varphi$ using randomness $s\mathbf{r}_{\varphi,\mathrm{f}}$ in the private slot.

- Hybrid H_3^b proceeds identically to H_2^b, except that the garbling in the private slots of the secret key vectors are generated for independent pad $\widehat{\mu}_\varphi^0$ with independent randomness $\widehat{\mathbf{r}}_{\varphi,\mathrm{f}}$. Note that the only difference between H_2^b and H_3^b is that in the former, the randomness and the pads of the garblings in the private slot are *random multiples* of those in the public slot, and that in the latter, those in the private slot are independently sampled. The DDH assumption in G_2 says

$$\left\{[\![\mu_\varphi, \mathbf{r}_{\varphi,\mathrm{f}};\ s\mu_\varphi, s\mathbf{r}_{\varphi,\mathrm{f}}]\!]_2\right\}_{\varphi\in[\Phi]} \approx \left\{[\![\mu_\varphi, \mathbf{r}_{\varphi,\mathrm{f}};\ \widehat{\mu}_\varphi^0, \widehat{\mathbf{r}}_{\varphi,\mathrm{f}}]\!]_2\right\}_{\varphi\in[\Phi]}.$$

  Given $[\![\mu_\varphi, \mathbf{r}_{\varphi,\mathrm{f}}]\!]_2$ and either $[\![s\mu_\varphi, s\mathbf{r}_{\varphi,\mathrm{f}}]\!]_2$ or $[\![\widehat{\mu}_\varphi^0, \widehat{\mathbf{r}}_{\varphi,\mathrm{f}}]\!]_2$, the secret key vectors can be efficiently computed *in the exponent*, and everything else can be efficiently computed. Therefore, $\mathsf{H}_2^b \approx \mathsf{H}_3^b$ by the DDH assumption in G_2.

We have completed the first step of the proof. Now we proceed to the second step, in which we switch $\widehat{\mu}_\varphi^0$ to independent random values $\widehat{\mu}_\varphi^1$ (inconsistent with $\mathbf{v}_{\varphi,\mathrm{init}}$ and $\mathbf{v}_{\varphi,q}$'s) one by one:

- Hybrid $\mathsf{H}_{4,\varphi}^b$ proceeds identically to H_3^b, except that for $\varphi' < \varphi$ (the first $\varphi - 1$ secret keys), $\mathbf{v}_{\varphi',\mathrm{pad}}$ (in $\mathsf{isk}_{\varphi',\mathrm{pad}}$) stores an independent random value $\widehat{\mu}_\varphi^1 \xleftarrow{\$} \mathbb{Z}_p$ at $\mathsf{pad}^{\mathrm{copy}}$.

Observe that $\mathsf{H}_{4,1}^b \equiv \mathsf{H}_3^b$. We show $\mathsf{H}_{4,\varphi}^b \approx \mathsf{H}_{4,\varphi+1}^b$ for all $\varphi \in [\Phi], b \in \{0,1\}$ as a separate claim:

Claim 39. $\mathsf{H}_{4,\varphi}^b \approx \mathsf{H}_{4,\varphi+1}^b$ *for all* $\varphi \in [\Phi], b \in \{0,1\}$.

For now, we focus on the last step:

- Hybrid H_5^b proceeds identically to H_3^b, except that all the pads $\widehat{\mu}_\varphi^0$ in $\mathbf{v}_{\varphi,\mathrm{pad}}[\mathsf{pad}^{\mathrm{copy}}]$ are replaced by independent random values $\widehat{\mu}_\varphi^1 \xleftarrow{\$} \mathbb{Z}_p$. The hybrid is shown in Figure 10. Observe that $\mathsf{H}_5^b \equiv \mathsf{H}_{4,\Phi+1}^b$.

  Note that in this hybrid, the inner products of $\mathbf{u}_{\mathrm{pad}}$ and $\mathbf{v}_{\varphi,\mathrm{pad}}$'s are $h + \widehat{\mu}_\varphi^1$, which hide h. However, h still appears in $\mathbf{u}_{\mathrm{pad}}$. In the next hybrid, we remove h from $\mathbf{u}_{\mathrm{pad}}$ and directly hardwire the inner products $h + \widehat{\mu}_\varphi^1$ in $\mathbf{v}_{\varphi,\mathrm{pad}}$'s so that h becomes information-theoretically hidden.

- Hybrid H_6^b proceeds identically to H_5^b, except that the random pad h is removed from $\mathbf{u}_{\mathrm{pad}}[\mathsf{pad}^{\mathsf{ct}}]$ and all the $\mathbf{v}_{\varphi,\mathrm{pad}}$ stores $h + \widehat{\mu}_\varphi^1$ at $\mathsf{pad}^{\mathrm{copy}}$. Since the inner products and the public slot of the secret key vectors remain unchanged, we have $\mathsf{H}_5^b \approx \mathsf{H}_6^b$ by the function-hiding property of IPFE.

We have $\mathsf{H}_6^0 \equiv \mathsf{H}_6^1$ because h is perfectly hidden by each $\widehat{\mu}_\varphi^1$ and perfectly hides the message. Therefore, by a hybrid argument, we conclude $\mathsf{Exp}_{\mathrm{CPA}}^0 \approx \mathsf{Exp}_{\mathrm{CPA}}^1$. □

Proof (Claim 39). Fix some $\varphi \in [\Phi], b \in \{0,1\}$ and we want to show $\mathsf{H}_{4,\varphi}^b \approx \mathsf{H}_{4,\varphi+1}^b$, for which we must modify sk_φ. There are three key ideas in the proof, which are logically chained in an anadiplosis fashion:

- To modify sk_φ, we temporarily use $\mathsf{pad}^{\mathsf{temp}}$ and $\mathfrak{s}_{\text{1-ABE}}$ exclusively for the interaction between ct and sk_φ, and use $\mathsf{pad}^{\text{copy}}$ and $\mathfrak{s}_{\text{copy}}$ for the interaction between ct and the other secret keys, similar to the *temporary* mode in the proof of Claim 36. With the interaction between ct and sk_φ isolated, we could hope to syntactically invoke 1-ABE security for them.

- To invoke 1-ABE security in $\mathfrak{s}_{\text{1-ABE}}$, we must make sure that in $\mathfrak{s}_{\text{1-ABE}}$, the ciphertext vectors use randomness independent of those in $\mathfrak{s}_{\text{copy}}$. This is done by invoking the DDH assumption twice in a row to make ct use $\mathbf{r}_{\text{x}}$ in $\mathsf{pad}^{\text{copy}}, \mathfrak{s}_{\text{copy}}$ and independent $\widehat{\mathbf{r}}_{\text{x}}$ in $\mathsf{pad}^{\mathsf{temp}}, \mathfrak{s}_{\text{1-ABE}}$, similar to how the labels are simulated in the proof of Theorem 34 by invoking the DDH assumption twice in a row.

- To invoke the DDH assumption to rerandomize $\mathbf{r}_{\text{x}}$, there needs to be a random value multiplied to it. However, in $\mathsf{H}^b_{4,\varphi}$, nothing is multiplied to μ's, $\mathbf{r}_{\text{f}}$'s, $\widehat{\mu}$'s, $\widehat{\mathbf{r}}_{\text{f}}$'s nor $\mathbf{r}_{\text{x}}$. This can be resolved by a change of variable introducing a negligible statistical error:

$$(\widehat{\mu}^0_\varphi, \widehat{\mu}^1_\varphi, \widehat{\mathbf{r}}_{\varphi,\text{f}}) \longleftrightarrow (\widehat{s}\,\widehat{\mu}^0_\varphi, \widehat{s}\,\widehat{\mu}^1_\varphi, \widehat{s}\,\widehat{\mathbf{r}}_{\varphi,\text{f}}) \text{ for } \widehat{s} \xleftarrow{\$} \mathbb{Z}_p.$$

  Having changed the variable, we move $\widehat{s}$ to $\mathbf{r}_{\text{x}}$ using the function-hiding property of IPFE, after which DDH can be invoked.

We now implement these ideas in the following hybrids $\mathsf{G}^\beta_0, \ldots, \mathsf{G}^\beta_6$:

- Hybrid G^β_0 proceeds identically to $\mathsf{H}^b_{4,\varphi+\beta}$, where the first $\varphi - 1$ secret keys have their $\widehat{\mu}^0_{\varphi'}$ replaced by $\widehat{\mu}^1_{\varphi'}$ in $\mathbf{v}_{\varphi',\text{pad}}[\mathsf{pad}^{\text{copy}}]$, the φ^{th} secret key stores either $\widehat{\mu}^0_\varphi$ (consistent with $\mathbf{v}_{\varphi,\text{init}}$ and $\mathbf{v}_{\varphi,q}$'s) or $\widehat{\mu}^1_\varphi$ (inconsistent). The hybrid is shown in Figure 11.

- Hybrid G^β_1 proceeds identically to G^β_0, except that a random multiplier $\widehat{s} \xleftarrow{\$} \mathbb{Z}_p$ is multiplied to the values at $\mathsf{pad}^{\text{copy}}, \mathfrak{s}_{\text{copy}}$ for sk_φ. Conditioned on $\widehat{s} \neq 0$ (which happens with overwhelming probability), G^β_1 and G^β_0 are identically distributed. Therefore, $\mathsf{G}^\beta_1 \approx_s \mathsf{G}^\beta_0$ taking into account the case $\widehat{s} = 0$.

- Hybrid G^β_2 proceeds identically to G^β_1, except that the interaction between ct and sk_φ is isolated from those between ct and the other secret keys, and that s is multiplied in ct instead of sk_φ:

  - In the vectors in *the φ^{th} secret key* sk_φ, values at $\mathsf{pad}^{\text{copy}}, \mathfrak{s}_{\text{copy}}$ are *moved* to $\mathsf{pad}^{\mathsf{temp}}, \mathfrak{s}_{\text{1-ABE}}$, and the multiplier $\widehat{s}$ is removed.
  - In the ciphertext vectors, values at $\mathsf{pad}^{\text{copy}}, \mathfrak{s}_{\text{copy}}$ are *copied* to $\mathsf{pad}^{\mathsf{temp}}, \mathfrak{s}_{\text{1-ABE}}$, and $\widehat{s}$ is multiplied to the new copy.

  As illustrated in Figure 11, the inner products and the public slots of the secret key vectors remain unchanged, so $\mathsf{G}^\beta_2 \approx \mathsf{G}^\beta_1$ by the function-hiding property of IPFE.

- Hybrid G^β_3 proceeds identically to G^β_2, except that in the ciphertext vectors, the $\widehat{s}\,\mathbf{r}_{\text{x}}$ in $\mathfrak{s}_{\text{1-ABE}}$ is replaced by an independent and uniformly random $\widehat{\mathbf{s}}$. The indistinguishability between G^β_2 and G^β_3 follows by the DDH assumption in G_1:

$$[\![\mathbf{r}_{\text{x}}, \widehat{s}, \widehat{s}\,\mathbf{r}_{\text{x}}]\!]_1 \approx [\![\mathbf{r}_{\text{x}}, \widehat{s}, \widehat{\mathbf{s}}]\!]_1 \text{ for } \mathbf{r}_{\text{x}}, \widehat{\mathbf{s}} \xleftarrow{\$} \mathbb{Z}_p^{[0..T]\times[N]\times[S]\times\{0,1\}^S}, \widehat{s} \xleftarrow{\$} \mathbb{Z}_p.$$

hybrid	vector	in $\mathfrak{s}_{\mathrm{pub}}$	$\mathsf{pad}^{\mathrm{copy}}$	in $\mathfrak{s}_{\mathrm{copy}}$	$\mathsf{pad}^{\mathrm{temp}}$	in $\mathfrak{s}_{\text{1-ABE}}$
$\mathsf{G}_0^\beta \equiv \mathsf{H}_{4,\varphi+\beta}^b$	$\varphi'<\varphi$ $\begin{cases}\mathbf{v}_{\varphi',\mathrm{pad}}\\ \mathbf{v}_{\varphi',\mathrm{init}}, \mathbf{v}_{\varphi',q}\end{cases}$	$\mu, \mathbf{r}_{\mathrm{f}}$'s	$\widehat{\mu}^1_{\varphi'} \xleftarrow{\$} \mathbb{Z}_p$	$\propto (\widehat{\mu}^0_{\varphi'}, \widehat{\mathbf{r}}_{\varphi',\mathrm{f}})$	0	0
	$\mathbf{v}_{\varphi,\mathrm{pad}}$		$\widehat{\mu}^\beta_\varphi$		0	
	$\mathbf{v}_{\varphi,\mathrm{init}}, \mathbf{v}_{\varphi,q}$	$\left(\begin{smallmatrix}\text{independent}\\ \text{of } \widehat{\mu}, \widehat{\mathbf{r}}_{\mathrm{f}}\text{'s}\end{smallmatrix}\right)$		$\propto (\widehat{\mu}^0_\varphi, \widehat{\mathbf{r}}_{\varphi,\mathrm{f}})$		0
	$\varphi'>\varphi$ $\begin{cases}\mathbf{v}_{\varphi',\mathrm{pad}}\\ \mathbf{v}_{\varphi',\mathrm{init}}, \mathbf{v}_{\varphi',q}\end{cases}$	and h	$\widehat{\mu}^0_{\varphi'}$	$\propto (\widehat{\mu}^0_{\varphi'}, \widehat{\mathbf{r}}_{\varphi',\mathrm{f}})$	0	0
	$\mathbf{u}_{\mathrm{pad}}$		1		0	
	$\mathbf{u}_{\mathrm{init}}, \mathbf{u}_{t,i,j,\mathbf{W}}$			$\propto (1, \mathbf{r}_{\mathrm{x}})$		0
G_1^β	$\mathbf{v}_{\varphi,\mathrm{pad}}$	$\mu, \mathbf{r}_{\mathrm{f}}$'s and h	$\boxed{\widehat{s}\,\widehat{\mu}^\beta_\varphi}$	$\boxed{\propto (\widehat{s}\,\widehat{\mu}^0_\varphi, \widehat{s}\,\widehat{\mathbf{r}}_{\varphi,\mathrm{f}})}$	$\boxed{0}$	$\boxed{0}$
	$\mathbf{v}_{\varphi,\mathrm{init}}, \mathbf{v}_{\varphi,q}$					
	$\mathbf{u}_{\mathrm{pad}}$		1		$\boxed{0}$	
	$\mathbf{u}_{\mathrm{init}}, \mathbf{u}_{t,i,j,\mathbf{W}}$			$\propto (1, \mathbf{r}_{\mathrm{x}})$		$\boxed{0}$
G_2^β	$\mathbf{v}_{\varphi,\mathrm{pad}}$	$\mu, \mathbf{r}_{\mathrm{f}}$'s and h	$\boxed{0}$		$\boxed{\widehat{\mu}^\beta_\varphi}$	
	$\mathbf{v}_{\varphi,\mathrm{init}}, \mathbf{v}_{\varphi,q}$			$\boxed{0}$		$\boxed{\propto (\widehat{\mu}^0_\varphi, \widehat{\mathbf{r}}_{\varphi,\mathrm{f}})}$
	$\mathbf{u}_{\mathrm{pad}}$		1		$\boxed{\widehat{s}}$	
	$\mathbf{u}_{\mathrm{init}}, \mathbf{u}_{t,i,j,\mathbf{W}}$			$\propto (1, \mathbf{r}_{\mathrm{x}})$		$\boxed{\propto (\widehat{s}, \boxed{\widehat{s}\,\mathbf{r}_{\mathrm{x}}})}$
G_3^β	$\mathbf{v}_{\varphi,\mathrm{pad}}$	$\mu, \mathbf{r}_{\mathrm{f}}$'s and h	0		$\widehat{\mu}^\beta_\varphi$	
	$\mathbf{v}_{\varphi,\mathrm{init}}, \mathbf{v}_{\varphi,q}$			0		$\propto (\widehat{\mu}^0_\varphi, \widehat{\mathbf{r}}_{\varphi,\mathrm{f}})$
	$\mathbf{u}_{\mathrm{pad}}$		1		$\widehat{s}$	
	$\mathbf{u}_{\mathrm{init}}, \mathbf{u}_{t,i,j,\mathbf{W}}$			$\propto (1, \mathbf{r}_{\mathrm{x}})$		$\propto (\widehat{s}, \boxed{\widehat{\mathbf{s}}})$
G_4^β	$\mathbf{v}_{\varphi,\mathrm{pad}}$	$\mu, \mathbf{r}_{\mathrm{f}}$'s and h	0		$\boxed{\widehat{\mu}^\beta_\varphi}$	
	$\mathbf{v}_{\varphi,\mathrm{init}}, \mathbf{v}_{\varphi,q}$			0		$\boxed{\propto (\widehat{\mu}^0_\varphi, \widehat{\mathbf{r}}_{\varphi,\mathrm{f}})}$
	$\mathbf{u}_{\mathrm{pad}}$		1		$\boxed{\widehat{s}}$	
	$\mathbf{u}_{\mathrm{init}}, \mathbf{u}_{t,i,j,\mathbf{W}}$			$\propto (1, \mathbf{r}_{\mathrm{x}})$		$\boxed{\propto (\widehat{s}, \boxed{\widehat{s}\,\widehat{\mathbf{r}}_{\mathrm{x}}})}$
G_5^β	$\mathbf{v}_{\varphi,\mathrm{pad}}$	$\mu, \mathbf{r}_{\mathrm{f}}$'s and h	0		$\boxed{\widehat{s}\,\widehat{\mu}^\beta_\varphi}$	
	$\mathbf{v}_{\varphi,\mathrm{init}}, \mathbf{v}_{\varphi,q}$			0		$\boxed{\propto (\widehat{s}\,\widehat{\mu}^0_\varphi, \widehat{s}\,\widehat{\mathbf{r}}_{\varphi,\mathrm{f}})}$
	$\mathbf{u}_{\mathrm{pad}}$		1		$\boxed{1}$	
	$\mathbf{u}_{\mathrm{init}}, \mathbf{u}_{t,i,j,\mathbf{W}}$			$\propto (1, \mathbf{r}_{\mathrm{x}})$		$\boxed{\propto (1, \widehat{\mathbf{r}}_{\mathrm{x}})}$
G_6^β	$\mathbf{v}_{\varphi,\mathrm{pad}}$	$\mu, \mathbf{r}_{\mathrm{f}}$'s and h	0		$\boxed{\widehat{\mu}^\beta_\varphi}$	
	$\mathbf{v}_{\varphi,\mathrm{init}}, \mathbf{v}_{\varphi,q}$			0		$\boxed{\propto (\widehat{\mu}^0_\varphi, \widehat{\mathbf{r}}_{\varphi,\mathrm{f}})}$
	$\mathbf{u}_{\mathrm{pad}}$		1		1	
	$\mathbf{u}_{\mathrm{init}}, \mathbf{u}_{t,i,j,\mathbf{W}}$			$\propto (1, \mathbf{r}_{\mathrm{x}})$		$\propto (1, \widehat{\mathbf{r}}_{\mathrm{x}})$

For brevity, the vectors for computing the labels are not spelled out, and the secret key vectors other than those for sk_φ are suppressed except in G_0^β.
See Figure 10 for the meaning of "$\propto \mathbf{z}$".

Figure 11: The hybrids for showing $\mathsf{H}_{4,\varphi}^b \approx \mathsf{H}_{4,\varphi+1}^b$ in the proof of IND-CPA security of our KP-ABE scheme for L.

- Hybrid G_4^β proceeds identically to G_3^β, except that in the ciphertext vectors, the $\widehat{\mathbf{s}}$ in $\mathfrak{s}_{\text{1-ABE}}$ is replaced by $\widehat{s}\,\widehat{\mathbf{r}}_{\mathrm{x}}$. The indistinguishability between G_3^β and G_4^β again follows by the DDH assumption in G_1:

$$[\![\widehat{s}, \widehat{\mathbf{s}}]\!]_1 \approx [\![\widehat{s}, \widehat{s}\,\widehat{\mathbf{r}}_{\mathrm{x}}]\!]_1 \text{ for } \widehat{\mathbf{s}}, \widehat{\mathbf{r}}_{\mathrm{x}} \xleftarrow{\$} \mathbb{Z}_p^{[0..T]\times[N]\times[S]\times\{0,1\}^S}, \widehat{s} \xleftarrow{\$} \mathbb{Z}_p.$$

- Hybrid G_5^β proceeds identically to G_4^β, except that the multiplier $\widehat{s}$ is moved from $\mathbf{u}$'s back to $\mathbf{v}_\varphi$'s as depicted in Figure 11. The indistinguishability between the two hybrids follows by the function-hiding property of IPFE.

- Hybrid G_6^β proceeds identically to G_5^β, except that the multiplier $\widehat{s}$ is removed. We have $\mathsf{G}_6^\beta \approx_s \mathsf{G}_5^\beta$.

Lastly, observe that in G_6^β, only the ciphertext vectors and the vectors for the φ^{th} secret key have non-zero values in $\mathfrak{s}_{\text{1-ABE}}$, and those values are generated exactly as specified by the 1-ABE for L (Construction 33). We use syntactically the same proof of the security of the 1-ABE scheme for L (Theorem 34) to argue $\mathsf{G}_6^0 \approx \mathsf{G}_6^1$:

- Each step in the proof of Theorem 34 uses the function-hiding property, the DDH assumption or the special piecewise security, all of which are still valid when translated into the case of G_6^0 and G_6^1 in the only natural way. More specifically, *i*) the function-hiding property is unaffected by the public slot, $\mathfrak{s}_{\text{copy}}, \text{pad}^{\text{copy}}$ and the additional vectors (in the other secret keys), as the values being manipulated during the proof are all in $\mathfrak{s}_{\text{1-ABE}}$ in the private slot, and the additional vectors (having values 0 in $\mathfrak{s}_{\text{1-ABE}}$) cannot "detect" the changes in $\mathfrak{s}_{\text{1-ABE}}$; and *ii*) the applications of the DDH assumptions and the special piecewise security still go through, because the garbling in $\mathfrak{s}_{\text{1-ABE}}$ is generated using randomness independent of everything else in G_6^β.

- In the translated version of the proof of Theorem 34, $\mathbf{v}_{\varphi,\text{pad}}[\mathsf{pad}^{\mathsf{temp}}]$ is always set to $\widehat{\mu}_\varphi^\beta$. In the final hybrids, both $\widehat{\mu}_\varphi^0$ and $\widehat{\mu}_\varphi^1$ are uniformly random and independent of everything else, so they can replace each other.

Therefore, $\mathsf{G}_6^0 \approx \mathsf{G}_6^1$, and by a hybrid argument, we conclude $\mathsf{H}_{4,\varphi}^b \equiv \mathsf{G}_0^0 \approx \mathsf{G}_0^1 \equiv \mathsf{H}_{4,\varphi+1}^b$. □

Remarks. The natural way to generalize Construction 37 to MDDH_k is to let the public slot store k independent garblings and let the private slot provide space for $k+1$ garblings — among the $k+1$ copies in the private slot, k of them are the counterpart of $\mathfrak{s}_{\text{copy}}$, and the last one of them is the counterpart of $\mathfrak{s}_{\text{1-ABE}}$. Of course, the underlying 1-ABE scheme must also be based on MDDH_k. This natural generalization uses vectors of dimension $\Theta(k^2)$, because each copy of the garbling already needs $\Theta(k)$ indices.

A simple optimization that shortens the vectors to only $\Theta(k)$ is to realize that the rerandomization of the (half) garblings in the security proof of ABE can be done without creating k copies of 1-ABE, essentially because each 1-ABE already uses k copies of random values to jointly generate the label functions. For this optimization to work, the copy in $\mathfrak{s}_{\text{pub}}$ needs to embed k independent pads in the secret keys and k independent random multipliers in the ciphertext, and decapsulation gives the inner product of the two. For the other two "compartments", $\mathfrak{s}_{\text{copy}}$ and $\mathfrak{s}_{\text{1-ABE}}$, each only needs to provide space for one pad.

The ABE scheme still uses 3 copies, and we use $(\mathbf{X} \parallel \mathbf{Y} \parallel \mathbf{Z})$ to represent an ABE key/ciphertext generated with randomness $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ for the $\mathfrak{s}_{\text{pub}}, \mathfrak{s}_{\text{copy}}, \mathfrak{s}_{\text{1-ABE}}$ copies ($\mathbf{0}$ means no copy is there; pads

are omitted). Roughly speaking, the first step (isolating the interaction between sk's and ct from the other ciphertexts) is done as follows:

Real	Hybrid 1	Hybrid 2	Hybrid 3
sk_φ: $(\mathbf{R}_{\mathrm{f},\varphi} \Vert \mathbf{0} \Vert \mathbf{0})$	sk_φ: $(\mathbf{R}_{\mathrm{f},\varphi} \Vert \mathbf{0} \Vert \mathbf{0})$	sk_φ: $(\mathbf{R}_{\mathrm{f},\varphi} \Vert \mathbf{A}^\intercal \mathbf{R}_{\mathrm{f},\varphi} \Vert \mathbf{0})$	sk_φ: $(\mathbf{R}_{\mathrm{f},\varphi} \Vert \widehat{\mathbf{R}}_{\mathrm{f},\varphi} \Vert \mathbf{0})$
ct: $(\mathbf{R}_{\mathrm{x}} \Vert \mathbf{0} \Vert \mathbf{0})$	ct: $(\mathbf{A}\mathbf{R}_{\mathrm{x}} \Vert \mathbf{0} \Vert \mathbf{0})$	ct: $(\mathbf{0} \Vert \mathbf{R}_{\mathrm{x}} \Vert \mathbf{0})$	ct: $(\mathbf{0} \Vert \mathbf{R}_{\mathrm{x}} \Vert \mathbf{0})$

Here, $\mathbf{R}_{\mathrm{f},\varphi}, \widehat{\mathbf{R}}_{\mathrm{f},\varphi} \xleftarrow{\$} \mathbb{Z}_p^{k \times Q_\varphi}, \mathbf{R}_{\mathrm{x}} \xleftarrow{\$} \mathbb{Z}_p^{[k] \times ([0..T] \times [N] \times [S] \times \{0,1\}^S)}$ and $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_p^{k \times k}$. The first transition is a change of variable introducing only a negligible statistical error. The second one is due to the function-hiding property of IPFE. The third reduces to $\mathrm{MDDH}^k_{k,\Phi+Q_1+\cdots+Q_\Phi}$.

The second step is to change the pads in $\mathfrak{s}_{\mathrm{copy}}$ to be inconsistent one by one. Similar to the proof of Claim 39, we achieve this via a loop of Φ iterations, one for each secret key. In the φ^{th} iteration, we *i*) introduce a random multiplier (this time a $k \times k$ matrix) in $\mathfrak{s}_{\mathrm{copy}}$ of sk_φ, *ii*) isolate the interaction between ct and sk_φ into $\mathfrak{s}_{\text{1-ABE}}$ with the multiplier moved into ct, *iii*) switch the randomness in $\mathfrak{s}_{\text{1-ABE}}$ of ct to be sampled independently from that in $\mathfrak{s}_{\mathrm{copy}}$ in ct, and *iv*) invoke 1-ABE security.

The last step is to remove the ciphertext pad from ct and hide it using the inconsistent pads in sk's, which is straightforward.

7.4 Extension to NL

Applying our construction of KP-ABE for L to non-deterministic Turing machines yields a KP-ABE for NL. In this section, we discuss what modifications need to be made to handle NL.

Non-Deterministic Turing Machines. The definitions of non-deterministic Turing machines (NTMs) and time/space bounded computation are the same as Definition 12, except with these changes:

- The transition criterion δ can be any relation between (i.e., any subset of the Cartesian product of) $[Q] \times \{0,1\}^2$ and $[Q] \times \{0,1\} \times \{0,\pm 1\}^2$, where $\big((q,x,w),(q',w',\Delta i,\Delta j)\big) \in \delta$ means that if the current state is q, the input tape symbol under scan is x and the work tape symbol under scan is w, then it is *valid* to transit into state q', overwrite w with w', and move the input and work tape pointer by offsets Δi and Δj respectively.

- The definition of *hanging in accepting states* is that for all $q \in [Q]$ such that $\mathbf{y}_{\mathrm{acc}}[q] = 1$ and all $x, w \in \{0,1\}$,

$$\delta \cap \big(\{(q,x,w)\} \times ([Q] \times \{0,1\} \times \{0,\pm 1\}^2)\big) = \big\{\big((q,x,w),(q,w,0,0)\big)\big\}.$$

- In the definition of acceptance,

$$\begin{aligned} & \delta(q_t, \mathbf{x}[i_t], \mathbf{W}_t[j_t]) = (q_{t+1}, \mathbf{W}_{t+1}[j_t], i_{t+1} - i_t, j_{t+1} - j_t) \\ \text{is changed to} \quad & \big((q_t, \mathbf{x}[i_t], \mathbf{W}_t[j_t]), (q_{t+1}, \mathbf{W}_{t+1}[j_t], i_{t+1} - i_t, j_{t+1} - j_t)\big) \in \delta. \end{aligned}$$

Note that in the case of non-deterministic machines, the machine might move off the tapes with certain choices of transitions, but that does not necessarily lead to rejection — it is just that particular path that is silently dropped. As long as one valid path does not exceed the space bound and land in an accepting state after T steps, the input is accepted.

Arithmetizing NTM Computation. The same matrix multiplication formula (thus the same AKGS) works for non-deterministic computation. We just need to use the non-deterministic version of the transition matrix, which share the same block structure as the deterministic ones.

Let $M = (Q, \mathbf{y}_{\text{acc}}, \delta)$ be an NTM. The *transition blocks* of M are

$$\mathbf{M}_{x,w,w',\Delta i,\Delta j} \in \mathbb{Z}_p^{Q\times Q} \text{ for } x, w, w' \in \{0,1\}, \Delta i, \Delta j \in \{0, \pm 1\}:$$
$$\mathbf{M}_{x,w,w',\Delta i,\Delta j}[q, q'] = \begin{cases} 1, & \text{if } \big((q, x, w), (q', w', \Delta i, \Delta j)\big) \in \delta; \\ 0, & \text{otherwise.} \end{cases}$$

The transition matrix of M for input of length N and space bound S is $\mathbf{M}_{N,S} \in \{0,1\}^{\mathcal{C}_{M,N,S}\times\mathcal{C}_{M,N,S}}$:

$$\mathbf{M}_{N,S}(\mathbf{x})[(i, j, \mathbf{W}, \sqcup), (i', j', \mathbf{W}', \sqcup)] = \begin{cases} \mathbf{M}_{\mathbf{x}[i],\mathbf{W}[j],\mathbf{W}'[j],i'-i,j'-j}, & \text{if } i'-i, j'-j \in \{0,\pm 1\} \text{ and} \\ & \quad \mathbf{W}[j''] = \mathbf{W}'[j''] \text{ for all } j'' \neq j; \\ \mathbf{0}, & \text{otherwise.} \end{cases}$$

It is readily verified that in $\mathbf{M}_{N,S}(\mathbf{x})[(i, j, \mathbf{W}, \sqcup), (\sqcup, \sqcup, \sqcup, \sqcup)]$, each transition block appears at most once. The matrix multiplication formula finds the number of accepting paths:

Lemma 40. *Let $M = (Q, \mathbf{y}_{\text{acc}}, \delta)$ be an NTM, $\mathbf{x} \in \{0,1\}^N$ for some $N \geq 1$ and $T, S \geq 1$. The number of valid computation paths reaching internal configuration $(i, j, \mathbf{W}, q)$ when running M starting from the initial configuration $(1, 1, \mathbf{0}_S, 1)$ with input $\mathbf{x} \in \{0,1\}^N$ and space bound S for T steps is*

$$\Big(\mathbf{e}_{(1,1,\mathbf{0}_S,1)}^{\mathsf{T}}\big(\mathbf{M}_{N,S}(\mathbf{x})\big)^T\Big)^{\mathsf{T}}[(i, j, \mathbf{W}, q)].$$

Moreover, computing $\mathbf{e}_{(1,1,\mathbf{0}_S,1)}^{\mathsf{T}}\big(\mathbf{M}_{N,S}(\mathbf{x})\big)^T\big(\mathbf{1}_{[N]\times[S]\times\{0,1\}^S} \otimes \mathbf{y}_{\text{acc}}\big)$ over the integers yields the number of accepting computation paths when running M starting from the initial configuration $(1, 1, \mathbf{0}_S, 1)$ with input $\mathbf{x} \in \{0,1\}^N$ within time T and space S, which is non-zero if and only if M accepts $\mathbf{x}$ within time T and space S.

Given the above lemma, we arithmetize time/space bounded computation of M by defining

$$M|_{N,T,S}(\mathbf{x}) = \mathbf{e}_{(1,1,\mathbf{0}_S,1)}^{\mathsf{T}}\big(\mathbf{M}_{N,S}(\mathbf{x})\big)^T\big(\mathbf{1}_{[N]\times[S]\times\{0,1\}^S} \otimes \mathbf{y}_{\text{acc}}\big) \quad \text{over } \mathbb{Z}_p \text{ for } \mathbf{x} \in \mathbb{Z}_p^N, \tag{10}$$

which evaluates to 0 if (but not necessarily only if) $\mathbf{x} \in \{0,1\}^N$ is not accepted by M within time T and space S.

Discussion on Correctness. The arithmetization is precise for the complexity class $\mathsf{modZ}_p\mathsf{L}$ [BDHM92] (restricted logspace counting class mod p). Each language $L \in \mathsf{modZ}_p\mathsf{L}$ is associated with a non-deterministic logspace Turing machine M such that

$$\begin{aligned} \mathbf{x} \in L \quad &\Longrightarrow \quad \#[\text{accepting paths of } M(\mathbf{x})] \not\equiv 0 \pmod p, \\ \mathbf{x} \notin L \quad &\Longrightarrow \quad \#[\text{accepting paths of } M(\mathbf{x})] = 0 \quad \text{(as an integer)}. \end{aligned}$$

As a special case, $\mathsf{UL} \subseteq \mathsf{modZ}_p\mathsf{L}$ for all p, where UL means *unambiguous* logspace — these are languages recognized by a logspace NTM such that for any input, there is at most 1 accepting path.

However, it is not known whether $\mathsf{NL} \subseteq \mathsf{modZ}_p\mathsf{L}$ for any p. Indeed, if the number of accepting paths is a non-zero multiple of p, the computation will be mistakenly rejected under our arithmetization. We circumvent this correctness issue by sampling an instance of pairing groups with *entropic* order.

Definition 41. *A* family of pairing groups *is $\mathcal{G}$ with an efficient algorithm $\mathcal{G}.\mathsf{Setup}(1^\lambda)$ (called* pairing group sampler*) that samples an instance $(p, G_1, G_2, G_\mathrm{T}, g_1, g_2, e)$ from some distribution of pairing groups such that*

- *G_1, G_2, G_T are groups of the same prime order p;*
- *G_1, G_2 are generated by g_1, g_2, respectively;*
- *$e : G_1 \times G_2 \to G_\mathrm{T}$ is bilinear and non-degenerate; and*
- *the group operations and e are efficiently computable, given the instance description.*

The family has entropic order *if the min-entropy of p is $\omega(\log \lambda)$:*

$$H_\lambda = -\log_2 \max_{p^\star} \Pr\left[(p, G_1, G_2, G_\mathrm{T}, g_1, g_2, e) \xleftarrow{\$} \mathsf{Setup}(1^\lambda) \ : \ p = p^\star\right].$$

Bracket notations are meaningful once an instance is fixed. The definitions of MDDH and DDH (Definitions 1 and 3) are changed accordingly so that the groups are sampled by $\mathcal{G}.\mathsf{Setup}$ and the description is given to the distinguisher along with the challenge. Moreover, the definition of IPFE (Definition 6) can easily generalize to work with families of pairing groups (still with perfect correctness). The construction (see Appendix A) implying Lemma 8 also works with families of pairing groups and retains perfect correctness.

For 1-ABE and ABE, we relax the correctness requirement so that decapsulation or decryption only needs to succeed with overwhelming probability for polynomial-sized inputs.

Definition 42 (1-ABE, altered)**.** *Let $\mathcal{G}$ be a pairing group sampler. A* 1-ABE scheme based on $\mathcal{G}$ *has the same syntax as in Definition 23, except that*

- Setup *additionally takes an instance of pairing groups as input.*
- *The* correctness *requirement is that for all sequence $\{z_\lambda = (P_\lambda \in \mathcal{P}_\lambda, y_\lambda \in Y_{P_\lambda}, x_\lambda \in X_{P_\lambda})\}_{\lambda \in \mathbb{N}}$ such that the description of z_λ is polynomially long in λ and $P_\lambda(x_\lambda, y_\lambda) = 1$ for all $\lambda \in \mathbb{N}$, the following probability is negligible in λ:*

$$\Pr\left[\begin{array}{l} \mathsf{groups} = (p, G_1, G_2, G_\mathrm{T}, g_1, g_2, e) \\ \qquad \xleftarrow{\$} \mathcal{G}.\mathsf{Setup}(1^\lambda) \\ \mathsf{msk} \xleftarrow{\$} \mathsf{Setup}(1^\lambda, \mathsf{groups}, P_\lambda) \\ \mu \xleftarrow{\$} \mathbb{Z}_p \\ \mathsf{sk} \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, \mathsf{msk}, y, \mu) \\ \mathsf{ct} \xleftarrow{\$} \mathsf{Enc}(1^\lambda, \mathsf{msk}, x) \end{array} : \mathsf{Dec}(1^\lambda, \mathsf{sk}, \mathsf{ct}) \neq [\![\mu]\!]_\mathrm{T}\right].$$

In the security experiments, the adversary additionally receives the instance description used to set up the 1-ABE scheme before making any queries.

It is also straightforward to relax the correctness requirement of ABE likewise. We now proceed to AKGS, 1-ABE and ABE schemes for NL.

AKGS, 1-ABE and ABE for NL. We simply plug in the matrix multiplication formula for NTM into AKGS, 1-ABE and ABE constructions.

Construction 43 (AKGS, 1-ABE and ABE for NL)**.** Let $\mathcal{G}$ be a family of pairing groups. The AKGS, 1-ABE and ABE for NL is obtained by plugging Equation (10) into Constructions 30 (AKGS), 33 (1-ABE) and 37 (ABE) with (secret-key or slotted) IPFE based on $\mathcal{G}$. Formally, 1-ABE and ABE work for the following singleton predicate space:

$$X = \left\{(\mathbf{x}, 1^T, 1^{2^S}) \,\middle|\, \mathbf{x} \in \{0,1\}^N \text{ for some } N \geq 1,\ T, S \geq 1\right\}, \quad Y = \left\{M \,\middle|\, M \text{ is an NTM}\right\},$$

$$P : X \times Y \to \{0,1\},\ \left((\mathbf{x}, 1^T, 1^{2^S}), M\right) \mapsto \begin{cases} 1, & \text{if } M \text{ accepts } \mathbf{x} \text{ within} \\ & \quad \text{time } T \text{ and space } S; \\ 0, & \text{otherwise;} \end{cases}$$

$$\mathcal{P}_\lambda = \{P\}, \quad \mathcal{P} = \{\mathcal{P}_\lambda\}_{\lambda \in \mathbb{N}}.$$

In 1-ABE and ABE, Dec returns $\perp$ if $M|_{N,T,S}(\mathbf{x}) \equiv 0 \pmod p$; otherwise, it recovers μ or $\overline{\mu}$ by dividing the Eval result by $M|_{N,T,S}(\mathbf{x})$. In ABE, the message space is chosen so that it is efficiently encodable and decodable in the target group given the instance description.

Theorem 44. *The AKGS in Construction 43 is* linear, correct, *and* special piecewise secure *with the same ordering of label functions and randomizers as described in Theorem 31.*

Suppose in Construction 43, $\mathcal{G}$ has entropic order, SXDH *holds in $\mathcal{G}$ and the IPFE schemes are function-hiding, then the constructed 1-ABE and ABE are correct (in the relaxed sense) and secure.*

Proof. The AKGS linearity, correctness and special piecewise security (and ordering) are straightforward. The security of 1-ABE and ABE also generalizes from the proof of Theorems 34 and 38 — by Lemma 40, $M|_{N,T,S}$ for non-deterministic Turing machine M always evaluates to 0 if the input is not accepted within the time/space bounds. We focus on the (relaxed) correctness of 1-ABE and ABE, which amounts to showing that $M|_{N,T,S}(\mathbf{x}) \equiv 0 \pmod p$ with only negligible probability for accepting computations.

Let the sequence in the correctness test be[17]

$$z_\lambda = (P,\ M_\lambda,\ \mathbf{x}_\lambda, 1^{T_\lambda}, 1^{2^{S_\lambda}}), \quad P \text{ is the only predicate}, \quad M_\lambda = (Q_\lambda, \mathbf{y}_{\mathrm{acc},\lambda}, \delta_\lambda), \quad \mathbf{x}_\lambda \in \{0,1\}^{N_\lambda}.$$

Since z_λ can only be polynomially long, the number of possible transitions (i.e., $|\delta_\lambda|$) and the time constraint T_λ are bounded by polynomial of λ, which implies that

$$\begin{aligned} A_\lambda &\overset{\text{def}}{=\!=} \#[\text{accepting paths of } M(\mathbf{x}) \text{ in } T \text{ steps}] \\ &\leq \#[\text{valid computation paths of } M(\mathbf{x}) \text{ in } T \text{ steps}] \\ &\leq |\delta_\lambda|^{T_\lambda} \leq 2^{c+\lambda^c} \end{aligned}$$

for some constant $c > 0$. In particular, this means A_λ has no more than $c + \lambda^c$ prime factors. Let the min-entropy of group order p be $H_\lambda = \omega(\log \lambda)$. By our assumption, M_λ accepts $\mathbf{x}_\lambda$ within time T_λ and space S_λ, so $A_\lambda > 0$ and

$$\Pr\left[(p, \ldots) \xleftarrow{\$} \mathcal{G}.\mathsf{Setup}(1^\lambda) \,:\, p \mid A_\lambda\right] \leq \sum_{\substack{p' \text{ prime} \\ p' \mid A_\lambda}} \Pr\left[(p, \ldots) \xleftarrow{\$} \mathcal{G}.\mathsf{Setup}(1^\lambda) \,:\, p = p'\right] \leq (c + \lambda^c) 2^{-H_\lambda}$$

is negligible in λ. □

Remarks. If the non-deterministic machine in question is *unambiguous*, the schemes can be instantiated with a sequence of pairing groups with perfect correctness.

[17] In case of ABE, there is also a message, but it does not affect the argument.

7.5 ABE for DFA/NFA

A deterministic finite automaton (DFA) can be converted (in polynomial time) to a deterministic Turing machine with space complexity 1 and time complexity N, where N is the length of the input. Similarly, an NFA can be converted to an NTM with space complexity 1 and time complexity N. Therefore, ABE schemes for DFA and NFA are just special cases of our ABE for L and NL.

As an optimization, DFA and NFA do not need to keep track of the input tape pointer in their internal configurations — it is always the current time step. This corresponds to a simpler formula (thus a simpler AKGS) for arithmetizing DFA and NFA. We formally define NFA (with DFA as a special kind of NFA), present the AKGS for it, and discuss how to construct 1-ABE and ABE for NFA. Again, we only consider the binary alphabet. The schemes readily extend to handle any polynomial-sized alphabet fixed at set-up time.

Definition 45. *A* non-deterministic finite automaton *is a tuple $(Q, \mathbf{y}_{\mathrm{acc}}, \delta)$, where $Q \geq 1$ is the number of states (we use $[Q]$ as the set of states and 1 the initial state), $\mathbf{y}_{\mathrm{acc}} \in \{0,1\}^Q$ indicates whether each state is accepting, and δ is a relation (state transition relation) between $[Q] \times \{0,1\}$ and $[Q]$. For $\mathbf{x} \in \{0,1\}^N$ for some $N \geq 1$, the NFA* accepts $\mathbf{x}$ *if there exists $q_0, \ldots, q_N \in [Q]$ (called an* accepting path*) such that*

$$q_0 = 1, \quad \big((q_{i-1}, \mathbf{x}[i]), q_i\big) \in \delta, \quad \mathbf{y}_{\mathrm{acc}}[q_N] = 1.$$

A path without the last condition is a computation path*. An NFA is* deterministic*, if δ is a function relation. An NFA is* unambiguous*, if for any input $\mathbf{x}$, there is at most 1 accepting path.*

Clearly, a DFA is always unambiguous.

Transition Matrix and Blocks. We use $\mathbf{e}_q \in \{0,1\}^Q$ to represent the current state of an NFA. For an NFA $M = (Q, \mathbf{y}_{\mathrm{acc}}, \delta)$, its *transition matrix* is

$$\mathbf{M}(x)[q, q'] = \begin{cases} 1, & \text{if } \big((q, x), q'\big) \in \delta; \\ 0, & \text{otherwise.} \end{cases}$$

For all $q \in [Q]$ and $x \in \{0,1\}$, consider $\mathbf{c}^\intercal = \mathbf{e}_q^\intercal \mathbf{M}(x)$ — we have $\mathbf{c} \in \{0,1\}^Q$ and $\mathbf{c}[q'] = 1$ if and only if q' is a valid state after the NFA reads x in state q. Inductively, $\mathbf{e}_q^\intercal \mathbf{M}(x_1) \cdots \mathbf{M}(x_n)$ is a vector that counts the number of computation paths reaching each state starting from state q after reading $x_1, \ldots, x_n$. Let the *transition blocks* be $\mathbf{M}_x = \mathbf{M}(x)$ for $x \in \{0,1\}$, then $\mathbf{M}(x) = (1-x)\mathbf{M}_0 + x\mathbf{M}_1$. We arithmetize the computation of NFA by defining

$$M|_N(\mathbf{x}) = \mathbf{e}_1^\intercal \prod_{i=1}^{N} \big((1 - \mathbf{x}[i])\mathbf{M}_0 + \mathbf{x}[i]\mathbf{M}_1\big) \cdot \mathbf{y}_{\mathrm{acc}} \quad \text{over } \mathbb{Z}_p \text{ for } \mathbf{x} \in \mathbb{Z}_p^N. \tag{11}$$

In case the NFA is unambiguous, $M|_N$ is binary over $\{0,1\}^N$ and indicates whether M accepts the input.

AKGS for NFA. Garbling $M|_N$ using the recursive mechanism for garbling matrix multiplication yields a special piecewise secure AKGS for NFA.

Construction 46. Let $\mathcal{F} = \{(M, 1^N, p) \mid M \text{ is an NFA, } p \text{ prime}\}$ be the function class, i.e., $M|_N$ over $\mathbb{Z}_p$ is encoded as $(M, 1^N, p)$. The AKGS $(\mathsf{Garble}, \mathsf{Eval})$ for $\mathcal{F}$ operates as follows:

- $\mathsf{Garble}\big((M, 1^N, p), \alpha, \beta\big)$ takes the NFA and the secrets as input. It computes the transition blocks $\mathbf{M}_0, \mathbf{M}_1$ for M, samples $\mathbf{r}_0, \ldots, \mathbf{r}_N \xleftarrow{\$} \mathbb{Z}_p^Q$, and defines the label functions by

$$\begin{aligned} L_{\mathrm{init}}(\mathbf{x}) &= \beta + \mathbf{e}_1^\mathsf{T}\mathbf{r}_0, \\ \text{for } i \in [N]: \quad \big(L_{i,q}(\mathbf{x})\big)_{q\in[Q]} &= -\mathbf{r}_{i-1} + \big((1-\mathbf{x}[i])\mathbf{M}_0 + \mathbf{x}[i]\mathbf{M}_1\big)\mathbf{r}_i, \\ \big(L_{N+1,q}(\mathbf{x})\big)_{q\in[Q]} &= -\mathbf{r}_N + \alpha\mathbf{y}_{\mathrm{acc}}. \end{aligned}$$

  The algorithm collects the coefficient vectors and outputs them.

- $\mathsf{Eval}\big((M, 1^N, p), \mathbf{x}, \ell_{\mathrm{init}}, (\ell_{i,q})_{i\in[N+1],q\in[Q]}\big)$ takes the NFA, the input string $\mathbf{x} \in \mathbb{Z}_p^N$ and the labels as input. It computes the transition blocks $\mathbf{M}_0, \mathbf{M}_1$ of M, sets $\boldsymbol{\ell}_i = (\ell_{i,q})_{q\in[Q]}$ for $i \in [N+1]$, and computes and returns

$$\ell_{\mathrm{init}} + \mathbf{e}_1^\mathsf{T} \sum_{i=1}^{N+1} \prod_{j=1}^{i-1} \big((1-\mathbf{x}[j])\mathbf{M}_0 + \mathbf{x}[j]\mathbf{M}_1\big) \cdot \boldsymbol{\ell}_i.$$

  Note: *It is straightforward to verify the construction satisfies the syntax of a linear AKGS. The evaluation correctness can be readily verified (similar to Construction 30).*

Remarks. Construction 46 coincides with the secret-sharing scheme for DFA by Waters [Wat12]. However, extending it to NFA via path-tracking makes the secret-sharing scheme insecure due to *backtracking* attacks, as observed in [Wat12]. In contrast, generalizing it to NFA via path-counting using the matrix multiplication formula retains the special piecewise security.

Theorem 47. *Construction 46 is special piecewise secure with L_{init} being the first label function, the other label functions sorted in increasing order of i, and the randomness sorted in the same order as the label functions.*

ABE for NFA. ABE for NFA can be obtained following the same blueprint of Constructions 33 and 37 and using a family of pairing groups with entropic order. With the pseudorandomness set to $\mathbf{r}_i = \mathbf{r}_{\mathrm{x}}[i]\mathbf{r}_{\mathrm{f}}$ for $\mathbf{r}_{\mathrm{x}} \xleftarrow{\$} \mathbb{Z}_p^{[0..N]}$ sampled at encryption time and $\mathbf{r}_{\mathrm{f}} \xleftarrow{\$} \mathbb{Z}_p^Q$ sampled at key generation time, the labels for $\alpha = \mu, \beta = 0$ can be computed as

$$\begin{aligned} \ell_{\mathrm{init}} &= \mathbf{r}_0[1] = \mathbf{r}_{\mathrm{x}}[0]\mathbf{r}_{\mathrm{f}}[1] = \langle \mathbf{u}_{\mathrm{init}}, \mathbf{v}_{\mathrm{init}} \rangle, \\ \text{for } i \in [N], q \in [Q]: \quad \ell_{i,q} &= -\mathbf{r}_{i-1}[q] + \big(((1-\mathbf{x}[i])\mathbf{M}_0 + \mathbf{x}[i]\mathbf{M}_1)\mathbf{r}_i\big)[q] \\ &= -\mathbf{r}_{\mathrm{x}}[i-1]\mathbf{r}_{\mathrm{f}}[q] + \mathbf{r}_{\mathrm{x}}[i](1-\mathbf{x}[i])(\mathbf{M}_0\mathbf{r}_{\mathrm{f}})[q] + \mathbf{r}_{\mathrm{x}}[i]\mathbf{x}[i](\mathbf{M}_1\mathbf{r}_{\mathrm{f}})[q] \\ &= \langle \mathbf{u}_i, \mathbf{v}_q \rangle, \\ \text{for } q \in [Q]: \quad \ell_{N+1,q} &= -\mathbf{r}_N[q] + \mu\mathbf{y}_{\mathrm{acc}}[q] = -\mathbf{r}_{\mathrm{x}}[N]\mathbf{r}_{\mathrm{f}}[q] + \mu\mathbf{y}_{\mathrm{acc}}[q] = \langle \mathbf{u}_{N+1}, \mathbf{v}_q \rangle, \end{aligned}$$

where the vectors are as follows:

vector	init	rand	acc	tb_0	tb_1	the other indices
$\mathbf{u}_{\mathrm{init}}$	$\mathbf{r}_{\mathrm{x}}[0]$	0	0	0	0	
$\mathbf{v}_{\mathrm{init}}$	$\mathbf{r}_{\mathrm{f}}[1]$	0	0	0	0	
$i \in [N]$: $\mathbf{u}_i$	0	$\mathbf{r}_{\mathrm{x}}[i-1]$	0	$\mathbf{r}_{\mathrm{x}}[i](1-\mathbf{x}[i])$	$\mathbf{r}_{\mathrm{x}}[i]\mathbf{x}[i]$	0
$\mathbf{u}_{N+1}$	0	$\mathbf{r}_{\mathrm{x}}[N]$	1	0	0	
$q \in [Q]$: $\mathbf{v}_q$	0	$-\mathbf{r}_{\mathrm{f}}[q]$	$\mu\mathbf{y}_{\mathrm{acc}}[q]$	$(\mathbf{M}_0\mathbf{r}_{\mathrm{f}})[q]$	$(\mathbf{M}_1\mathbf{r}_{\mathrm{f}})[q]$	

As is the case for the other 1-ABE schemes, 1-ABE for NFA will need extra indices for the security proof, the exact number of which can be figured out easily but tediously. Likewise, the ABE scheme will need more copies of the indices for computing the labels (cf. Construction 37).

Corollary 48. *Assuming a family of pairing groups with entropic order in which* SXDH *holds, there is a* compact *and* adaptively secure *ABE for NFA (satisfying the relaxed correctness requirement). If the NFA are restricted to unambiguous ones, the scheme can be instantiated in a sequence of pairing groups with perfect correctness. Moreover,* SXDH *can be relaxed to* MDDH_k *in both of the source groups for any integer* $k \geq 1$.

Acknowledgments. The authors were supported by NSF grants[18] CNS-1528178, CNS-1929901, CNS-1936825 (CAREER). The authors thank Hoeteck Wee for helpful discussions and the anonymous reviewers for insightful comments.

References

[AC17] Shashank Agrawal and Melissa Chase. Simplifying design and analysis of complex predicate encryption schemes. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 627–656. Springer, Heidelberg, April / May 2017.

[ACC+16] Prabhanjan Ananth, Yu-Chi Chen, Kai-Min Chung, Huijia Lin, and Wei-Kai Lin. Delegating RAM computations with adaptive soundness and privacy. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 3–30. Springer, Heidelberg, October / November 2016.

[AFS18] Prabhanjan Ananth, Xiong Fan, and Elaine Shi. Towards attribute-based encryption for RAMs from LWE: Sub-linear decryption, and more. Cryptology ePrint Archive, Report 2018/273, 2018. https://eprint.iacr.org/2018/273.

[AIK04] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0. In *45th FOCS*, pages 166–175. IEEE Computer Society Press, October 2004.

[AIK11] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. How to garble arithmetic circuits. In Rafail Ostrovsky, editor, *52nd FOCS*, pages 120–129. IEEE Computer Society Press, October 2011.

[ALS16] Shweta Agrawal, Benoît Libert, and Damien Stehlé. Fully secure functional encryption for inner products, from standard assumptions. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 333–362. Springer, Heidelberg, August 2016.

[AM18] Shweta Agrawal and Monosij Maitra. FE and iO for turing machines from minimal assumptions. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 473–512. Springer, Heidelberg, November 2018.

[18] 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.

[AMY19a] Shweta Agrawal, Monosij Maitra, and Shota Yamada. Attribute based encryption (and more) for nondeterministic finite automata from LWE. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 765–797. Springer, Heidelberg, August 2019.

[AMY19b] Shweta Agrawal, Monosij Maitra, and Shota Yamada. Attribute based encryption for deterministic finite automata from DLIN. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part II*, volume 11892 of *LNCS*, pages 91–117. Springer, Heidelberg, December 2019.

[AS16] Prabhanjan Vijendra Ananth and Amit Sahai. Functional encryption for turing machines. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part I*, volume 9562 of *LNCS*, pages 125–153. Springer, Heidelberg, January 2016.

[AS17] Prabhanjan Ananth and Amit Sahai. Projective arithmetic functional encryption and indistinguishability obfuscation from degree-5 multilinear maps. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 152–181. Springer, Heidelberg, April / May 2017.

[Att14] Nuttapong Attrapadung. Dual system encryption via doubly selective security: Framework, fully secure functional encryption for regular languages, and more. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 557–577. Springer, Heidelberg, May 2014.

[Att16] Nuttapong Attrapadung. Dual system encryption framework in prime-order groups via computational pair encodings. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 591–623. Springer, Heidelberg, December 2016.

[Att17] Nuttapong Attrapadung. Dual system framework in multilinear settings and applications to fully secure (compact) ABE for unbounded-size circuits. In Serge Fehr, editor, *PKC 2017, Part II*, volume 10175 of *LNCS*, pages 3–35. Springer, Heidelberg, March 2017.

[BDHM92] Gerhard Buntrock, Carsten Damm, Ulrich Hertrampf, and Christoph Meinel. Structure and importance of logspace-MOD class. *Mathematical Systems Theory*, 25(3):223–237, September 1992.

[BG98] Amos Beimel and Anna Gal. On arithmetic branching programs. In *IN PROC. OF THE 13TH ANNUAL IEEE CONFERENCE ON COMPUTATIONAL COMPLEXITY*, pages 68–80, 1998.

[BGG+14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 533–556. Springer, Heidelberg, May 2014.

[BGL+15] Nir Bitansky, Sanjam Garg, Huijia Lin, Rafael Pass, and Sidharth Telang. Succinct randomized encodings and their applications. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 439–448. ACM Press, June 2015.

[BJK15] Allison Bishop, Abhishek Jain, and Lucas Kowalczyk. Function-hiding inner product encryption. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 470–491. Springer, Heidelberg, November / December 2015.

[BL15] Xavier Boyen and Qinyi Li. Attribute-based encryption for finite automata from LWE. In Man Ho Au and Atsuko Miyaji, editors, *ProvSec 2015*, volume 9451 of *LNCS*, pages 247–267. Springer, Heidelberg, November 2015.

[BMZ19] James Bartusek, Fermi Ma, and Mark Zhandry. The distinction between fixed and random generators in group-based assumptions. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 801–830. Springer, Heidelberg, August 2019.

[BSW07] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *2007 IEEE Symposium on Security and Privacy (SP '07)*, pages 321–334, May 2007.

[CGKW18] Jie Chen, Junqing Gong, Lucas Kowalczyk, and Hoeteck Wee. Unbounded ABE via bilinear entropy expansion, revisited. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 503–534. Springer, Heidelberg, April / May 2018.

[CGW15] Jie Chen, Romain Gay, and Hoeteck Wee. Improved dual system ABE in prime-order groups via predicate encodings. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 595–624. Springer, Heidelberg, April 2015.

[CGW18] Jie Chen, Junqing Gong, and Hoeteck Wee. Improved inner-product encryption with adaptive security and full attribute-hiding. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 673–702. Springer, Heidelberg, December 2018.

[CHJV15] Ran Canetti, Justin Holmgren, Abhishek Jain, and Vinod Vaikuntanathan. Succinct garbling and indistinguishability obfuscation for RAM programs. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 429–437. ACM Press, June 2015.

[DDM16] Pratish Datta, Ratna Dutta, and Sourav Mukhopadhyay. Functional encryption for inner product with full function privacy. In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *PKC 2016, Part I*, volume 9614 of *LNCS*, pages 164–195. Springer, Heidelberg, March 2016.

[EHK+13] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. An algebraic framework for Diffie-Hellman assumptions. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 129–147. Springer, Heidelberg, August 2013.

[GKP+13a] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. How to run turing machines on encrypted data. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 536–553. Springer, Heidelberg, August 2013.

[GKP+13b] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 555–564. ACM Press, June 2013.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 89–98. ACM Press, October / November 2006. Available as Cryptology ePrint Archive Report 2006/309.

[GV15] Sergey Gorbunov and Dhinakaran Vinayagamurthy. Riding on asymmetry: Efficient ABE for branching programs. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 550–574. Springer, Heidelberg, November / December 2015.

[GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 545–554. ACM Press, June 2013.

[GWW19] Junqing Gong, Brent Waters, and Hoeteck Wee. ABE for DFA from k-lin. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 732–764. Springer, Heidelberg, August 2019.

[HKS15] Dennis Hofheinz, Jessica Koch, and Christoph Striecks. Identity-based encryption with (almost) tight security in the multi-instance, multi-ciphertext setting. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 799–822. Springer, Heidelberg, March / April 2015.

[IK97] Yuval Ishai and Eyal Kushilevitz. Private simultaneous messages protocols with applications. In *In Proc. of 5th ISTCS*, pages 174–183, 1997.

[IK00] Yuval Ishai and Eyal Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *41st FOCS*, pages 294–304. IEEE Computer Society Press, November 2000.

[IK02] Yuval Ishai and Eyal Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In Peter Widmayer, Francisco Triguero Ruiz, Rafael Morales Bueno, Matthew Hennessy, Stephan Eidenbenz, and Ricardo Conejo, editors, *ICALP 2002*, volume 2380 of *LNCS*, pages 244–256. Springer, Heidelberg, July 2002.

[IW14] Yuval Ishai and Hoeteck Wee. Partial garbling schemes and their applications. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *ICALP 2014, Part I*, volume 8572 of *LNCS*, pages 650–662. Springer, Heidelberg, July 2014.

[JKK+17] Zahra Jafargholi, Chethan Kamath, Karen Klein, Ilan Komargodski, Krzysztof Pietrzak, and Daniel Wichs. Be adaptive, avoid overcommitting. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 133–163. Springer, Heidelberg, August 2017.

[KLM+18] Sam Kim, Kevin Lewi, Avradip Mandal, Hart Montgomery, Arnab Roy, and David J. Wu. Function-hiding inner product encryption is practical. In Dario Catalano and Roberto De Prisco, editors, *SCN 18*, volume 11035 of *LNCS*, pages 544–562. Springer, Heidelberg, September 2018.

[KLMM19] Lucas Kowalczyk, Jiahui Liu, Tal Malkin, and Kailash Meiyappan. Mitigating the one-use restriction in attribute-based encryption. In Kwangsu Lee, editor, *ICISC 18*, volume 11396 of *LNCS*, pages 23–36. Springer, Heidelberg, November 2019.

[KLW15] Venkata Koppula, Allison Bishop Lewko, and Brent Waters. Indistinguishability obfuscation for turing machines with unbounded memory. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 419–428. ACM Press, June 2015.

[KNTY19] Fuyuki Kitagawa, Ryo Nishimaki, Keisuke Tanaka, and Takashi Yamakawa. Adaptively secure and succinct functional encryption: Improving security and efficiency, simultaneously. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 521–551. Springer, Heidelberg, August 2019.

[KSW13] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. *Journal of Cryptology*, 26(2):191–224, April 2013.

[KW19] Lucas Kowalczyk and Hoeteck Wee. Compact adaptively secure ABE for NC^1 from k-lin. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 3–33. Springer, Heidelberg, May 2019.

[Lin17] Huijia Lin. Indistinguishability obfuscation from SXDH on 5-linear maps and locality-5 PRGs. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 599–629. Springer, Heidelberg, August 2017.

[LOS+10] Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 62–91. Springer, Heidelberg, May / June 2010.

[LV16] Huijia Lin and Vinod Vaikuntanathan. Indistinguishability obfuscation from DDH-like assumptions on constant-degree graded encodings. In Irit Dinur, editor, *57th FOCS*, pages 11–20. IEEE Computer Society Press, October 2016.

[LW10] Allison B. Lewko and Brent Waters. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 455–479. Springer, Heidelberg, February 2010.

[LW11] Allison B. Lewko and Brent Waters. Unbounded HIBE and attribute-based encryption. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 547–567. Springer, Heidelberg, May 2011.

[LW12] Allison B. Lewko and Brent Waters. New proof methods for attribute-based encryption: Achieving full security through selective techniques. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 180–198. Springer, Heidelberg, August 2012.

[Nis91] Noam Nisan. Lower bounds for non-commutative computation (extended abstract). In *23rd ACM STOC*, pages 410–418. ACM Press, May 1991.

[OT09] Tatsuaki Okamoto and Katsuyuki Takashima. Hierarchical predicate encryption for inner-products. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 214–231. Springer, Heidelberg, December 2009.

[OT10] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 191–208. Springer, Heidelberg, August 2010.

[OT12] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure unbounded inner-product and attribute-based encryption. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 349–366. Springer, Heidelberg, December 2012.

[SSW09] Emily Shen, Elaine Shi, and Brent Waters. Predicate privacy in encryption systems. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 457–473. Springer, Heidelberg, March 2009.

[SW08] Elaine Shi and Brent Waters. Delegating capabilities in predicate encryption systems. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP 2008, Part II*, volume 5126 of *LNCS*, pages 560–578. Springer, Heidelberg, July 2008.

[TAO16] Junichi Tomida, Masayuki Abe, and Tatsuaki Okamoto. Efficient functional encryption for inner-product values with full-hiding security. In Matt Bishop and Anderson C. A. Nascimento, editors, *ISC 2016*, volume 9866 of *LNCS*, pages 408–425. Springer, Heidelberg, September 2016.

[Wat09] Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 619–636. Springer, Heidelberg, August 2009.

[Wat12] Brent Waters. Functional encryption for regular languages. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 218–235. Springer, Heidelberg, August 2012.

[Wee14] Hoeteck Wee. Dual system encryption via predicate encodings. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 616–637. Springer, Heidelberg, February 2014.

[Wee17] Hoeteck Wee. Attribute-hiding predicate encryption in bilinear groups, revisited. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 206–233. Springer, Heidelberg, November 2017.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986.

A Construction of Function-Hiding Slotted IPFE

In this section, we construct the function-hiding slotted IPFE promised in Lemma 8 by applying the double encryption technique [LV16,Lin17] to the adaptively secure public-key encryption scheme in [ALS16,Wee17].

First, we recall the adaptively secure public-key IPFE scheme in [ALS16,Wee17] in our syntax. Note that it suffices to consider index sets being $\mathfrak{s} = [n]$.

Construction 49 ([ALS16,Wee17])**.** We describe the construction for any fixed value of the security parameter λ and suppress the appearance of λ below for simplicity of notations. Let $\mathcal{G}$ be pairing groups of order p such that the MDDH_k assumption holds in G_1. The public-key IPFE scheme $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ based on $\mathcal{G}$ operates as follows:

- $\mathsf{Setup}(1^n)$ takes the dimension of vectors in unary (i.e., $[n]$ is encoded as 1^n) as input and generates the master public/secret key pair by

$$\begin{aligned} &\mathbf{A} \xleftarrow{\$} \mathbb{Z}_p^{k\times(k+1)}, && \mathbf{W} \xleftarrow{\$} \mathbb{Z}_p^{(k+1)\times n}, \\ &\mathsf{mpk} = ([\![\mathbf{A}]\!]_1, [\![\mathbf{AW}]\!]_1), && \mathsf{msk} = \mathbf{W}. \end{aligned}$$

- $\mathsf{KeyGen}(\mathsf{msk}, [\![\mathbf{v}]\!]_2)$ takes the master secret key $\mathsf{msk} = \mathbf{W}$ and a vector $\mathbf{v} \in \mathbb{Z}_p^n$ (encoded in G_2) as input. It returns $\mathsf{sk} = (-\mathbf{W}[\![\mathbf{v}]\!]_2, [\![\mathbf{v}]\!]_2)$ as the secret key for $\mathbf{v}$.

- $\mathsf{Enc}(\mathsf{mpk}, [\![\mathbf{u}]\!]_1)$ takes the master public key $\mathsf{mpk} = ([\![\mathbf{A}]\!]_1, [\![\mathbf{AW}]\!]_1)$ and a vector $\mathbf{u} \in \mathbb{Z}_p^n$ (encoded in G_1) as input. It generates the ciphertext for $\mathbf{u}$ by

$$\mathbf{s} \xleftarrow{\$} \mathbb{Z}_p^k, \quad \mathsf{ct} = (\mathbf{s}^\mathsf{T}[\![\mathbf{A}]\!]_1, \mathbf{s}^\mathsf{T}[\![\mathbf{AW}]\!]_1 + [\![\mathbf{u}^\mathsf{T}]\!]_1).$$

- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$ takes a secret key and a ciphertext as input. It parses sk as $([\![\mathbf{y}]\!]_2, [\![\mathbf{w}]\!]_2)$ and ct as $([\![\mathbf{z}^\mathsf{T}]\!]_1, [\![\mathbf{x}^\mathsf{T}]\!]_1)$ and returns $[\![\mathbf{z}^\mathsf{T}]\!]_1[\![\mathbf{y}]\!]_2 + [\![\mathbf{x}^\mathsf{T}]\!]_1[\![\mathbf{w}]\!]_2$ as the decryption result.

  Note: *The correctness of the scheme is readily verified by the equation*

$$\mathbf{z}^\mathsf{T}\mathbf{y} + \mathbf{x}^\mathsf{T}\mathbf{w} = (\mathbf{s}^\mathsf{T}\mathbf{A})(-\mathbf{Wv}) + (\mathbf{s}^\mathsf{T}\mathbf{AW} + \mathbf{u}^\mathsf{T})\mathbf{v} = \mathbf{u}^\mathsf{T}\mathbf{v}.$$

The (standard) IND-CPA security of a public-key IPFE scheme can be defined using the experiments for function-hiding security of slotted IPFE (Definition 7) by setting $\mathfrak{s}_{\mathrm{priv}} = \varnothing$ and $\mathfrak{s}_{\mathrm{pub}} = \mathfrak{s}$ — this means $\mathbf{v}_j^0 = \mathbf{v}_j^1$ in the challenges, i.e., only the ciphertext vector may vary. The above construction satisfies this notion.

Lemma 50 ([ALS16,Wee17])**.** *Construction 49 is IND-CPA secure.*

Remarks. In [ALS16,Wee17], Construction 49 is presented in a (not necessarily pairing) group in which MDDH_k holds. In particular, the secret key vector and the secret key are vectors encoded in $\mathbb{Z}_p$ instead of G_2. We encode them in G_2 only to adhere to the formalism used in this work. Moreover, for any group G_1 of order p, we can build a "pairing group" by setting $G_2 = \mathbb{Z}_p$ (additive group of integers modulo p) and $e(g, h) = g^h$. The additional benefit of using this formalism, as we shall see soon, is that certain preconditions of applying the technique in [LV16,Lin17] can be simplified.

We note that Construction 49 has four special properties:

- A secret key is a vector (of dimension $(k+1)+n$) encoded in G_2.
- A ciphertext is a vector (of dimension $(k+1)+n$) encoded in G_1.
- The secret key for $\mathbf{0}$ is $\mathbf{0}$ (encoded in G_2) (or rather, any distribution known with only mpk).
- The decryption algorithm simply computes the inner product of the secret key and the ciphertext using the pairing operation.

The first two properties suggest that it is syntactically possible to nest IPFE schemes. The last property guarantees that nesting preserves correctness. The key idea of the double encryption technique in [LV16,Lin17] is to nest IPFE schemes, one using G_1 for ciphertexts and the other using G_1 for secret keys, so that both secret keys and ciphertexts in the resulting scheme are protected by one instance of public-key IPFE.

Definition 51. *Let $\mathcal{G} = (G_1, G_2, G_T, g_1, g_2, e)$ be pairing groups. Its* opposite *is*

$$\mathcal{G}^{\mathrm{op}} = (G_2, G_1, G_T, g_2, g_1, e^{\mathrm{op}}), \quad e^{\mathrm{op}}(h_2, h_1) = e(h_1, h_2).$$

Clearly, given a sequence of pairing groups, taking the opposite yields another sequence of pairing groups whose operations are still efficient (using the algorithms for the original sequence). Armed with this notion, we are ready to apply the double encryption technique tailored to our needs.

Construction 52 (slotted IPFE [LV16,Lin17])**.** We describe the construction for any fixed value of the security parameter λ and suppress the appearance of λ below for simplicity of notations. Let $\mathcal{G}$ be pairing groups of order p such that the MDDH_k assumption holds in *both* G_1 *and* G_2, and let $\mathcal{G}^{\mathrm{op}}$ be its opposite. Instantiate Construction 49 to obtain

$$\begin{aligned} &\text{(outer scheme)} \quad (\mathsf{Setup}_{\mathsf{out}}, \mathsf{KeyGen}_{\mathsf{out}}, \mathsf{Enc}_{\mathsf{out}}, \mathsf{Dec}_{\mathsf{out}}) \quad \text{based on } \mathcal{G} \\ &\text{and (inner scheme)} \quad (\mathsf{Setup}_{\mathsf{in}}, \mathsf{KeyGen}_{\mathsf{in}}, \mathsf{Enc}_{\mathsf{in}}, \mathsf{Dec}_{\mathsf{in}}) \quad \text{based on } \mathcal{G}^{\mathrm{op}}. \end{aligned}$$

The slotted IPFE scheme based on $\mathcal{G}$ operates as follows:

- $\mathsf{Setup}(1^{n_{\mathrm{pub}}}, 1^{n_{\mathrm{priv}}})$ takes the public/private slot lengths in unary as input (i.e., $\mathfrak{s}_{\mathrm{pub}} = [n_{\mathrm{pub}}]$ and $\mathfrak{s}_{\mathrm{priv}} = [n_{\mathrm{pub}} + n_{\mathrm{priv}}] \setminus [n_{\mathrm{pub}}]$). It generates master public/secret key pairs for both inner and outer public-key IPFE schemes:

  $$(\mathsf{mpk}_{\mathsf{in}}, \mathsf{msk}_{\mathsf{in}}) \xleftarrow{\$} \mathsf{Setup}_{\mathsf{in}}(1^{2n_{\mathrm{priv}}}), \quad (\mathsf{mpk}_{\mathsf{out}}, \mathsf{msk}_{\mathsf{out}}) \xleftarrow{\$} \mathsf{Setup}_{\mathsf{out}}(1^{n_{\mathrm{pub}}+(k+1)+2n_{\mathrm{priv}}}).$$

  The algorithm returns $\mathsf{mpk} = \mathsf{mpk}_{\mathsf{out}}$ and $\mathsf{msk} = (\mathsf{mpk}_{\mathsf{in}}, \mathsf{msk}_{\mathsf{in}}, \mathsf{mpk}_{\mathsf{out}}, \mathsf{msk}_{\mathsf{out}})$.

  Note: *We explain the choice of parameters. The inner scheme is used to encrypt the private slot, and only the first n_{priv} components are used in the honest algorithms — the extra n_{priv} components provide programming space in the security proof. The outer scheme is used to encrypt the whole vector. The first n_{pub} components store the public slot, and the other $(k+1)+2n_{\mathrm{priv}}$ components store the* secret key/ciphertext *of the private slot.*

  For notational simplicity, we will suppress the master secret (resp. public) key as an input to key generation (resp. encryption) algorithms of the inner/outer schemes — there is only one natural choice for each of them.

- $\mathsf{KeyGen}(\mathsf{msk}, [\![\mathbf{v}]\!]_2)$ takes the master secret key msk and a vector $\mathbf{v} \in \mathbb{Z}_p^{n_{\mathrm{pub}}+n_{\mathrm{priv}}}$ (encoded in G_2) as input. It separates $\mathbf{v}$ by slots, i.e., $\mathbf{v} = (\mathbf{v}_{\mathrm{pub}}, \mathbf{v}_{\mathrm{priv}})$ with $\mathbf{v}_{\mathrm{pub}} \in \mathbb{Z}_p^{n_{\mathrm{pub}}}$ and $\mathbf{v}_{\mathrm{priv}} \in \mathbb{Z}_p^{n_{\mathrm{priv}}}$, and generates the secret key by

  $$\mathsf{sk} \xleftarrow{\$} \mathsf{KeyGen}_{\mathsf{out}}\big([\![\mathbf{v}_{\mathrm{pub}}]\!]_2, \mathsf{Enc}_{\mathsf{in}}([\![\mathbf{v}_{\mathrm{priv}}]\!]_2, [\![\mathbf{0}_{n_{\mathrm{priv}}}]\!]_2)\big).$$

- $\mathsf{Enc}(\mathsf{msk}, [\![\mathbf{u}]\!]_1)$ takes the master secret key msk and a vector $\mathbf{u} \in \mathbb{Z}_p^{n_{\text{pub}}+n_{\text{priv}}}$ (encoded in G_1) as input. It separates $\mathbf{u}$ by slots into $\mathbf{u}_{\text{pub}} \in \mathbb{Z}_p^{n_{\text{pub}}}$ and $\mathbf{u}_{\text{priv}} \in \mathbb{Z}_p^{n_{\text{priv}}}$, and generates the ciphertext by

$$\mathsf{ct} \xleftarrow{\$} \mathsf{Enc}_{\mathsf{out}}\big([\![\mathbf{u}_{\text{pub}}]\!]_1, \mathsf{KeyGen}_{\mathsf{in}}([\![\mathbf{u}_{\text{priv}}]\!]_1, [\![\mathbf{0}_{n_{\text{priv}}}]\!]_1)\big).$$

- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$ returns $\mathsf{Dec}_{\mathsf{out}}(\mathsf{sk}, \mathsf{ct})$.

  Note: *We verify the correctness of the scheme. By the correctness of the outer scheme,*

  $$\mathsf{Dec}_{\mathsf{out}}(\mathsf{sk}, \mathsf{ct}) = [\![\langle \mathbf{u}_{\text{pub}}, \mathbf{v}_{\text{pub}} \rangle + \langle \mathbf{x}, \mathbf{y} \rangle]\!]_{\mathrm{T}},$$

  where $\mathbf{x}, \mathbf{y}$ are the vectors encoded in the secret key/ciphertext of the inner scheme. By the correctness of the inner scheme and the fact that decryption computes $\langle \mathbf{x}, \mathbf{y} \rangle$ in the exponent in the target group, we have

  $$\langle \mathbf{x}, \mathbf{y} \rangle = \langle (\mathbf{u}_{\text{priv}}, \mathbf{0}_{n_{\text{priv}}}), (\mathbf{v}_{\text{priv}}, \mathbf{0}_{n_{\text{priv}}}) \rangle = \langle \mathbf{u}_{\text{priv}}, \mathbf{v}_{\text{priv}} \rangle.$$

  Combining the two gives $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) = [\![\langle \mathbf{u}_{\text{pub}}, \mathbf{v}_{\text{pub}} \rangle + \langle \mathbf{u}_{\text{priv}}, \mathbf{v}_{\text{priv}} \rangle]\!]_{\mathrm{T}} = [\![\langle \mathbf{u}, \mathbf{v} \rangle]\!]_{\mathrm{T}}$.

- $\mathsf{SlotEnc}(\mathsf{mpk}, [\![\mathbf{u}_{\text{pub}}]\!]_1)$ takes the master public key mpk and a vector $\mathbf{u}_{\text{pub}} \in \mathbb{Z}_p^{n_{\text{pub}}}$ (encoded in G_1) as input. It generates the ciphertext by

  $$\mathsf{ct} \xleftarrow{\$} \mathsf{Enc}_{\mathsf{out}}\big([\![\mathbf{u}_{\text{pub}}]\!]_1, [\![\mathbf{0}_{(k+1)+2n_{\text{priv}}}]\!]_1\big).$$

  Note: $\mathsf{SlotEnc}$ *and* Enc *generates identically distributed ciphertexts if* $\mathbf{u}_{\text{priv}} = \mathbf{0}_{n_{\text{priv}}}$ — *this is because the secret key for the zero vector in Construction 49 is the zero vector. This construction can be instantiated as long as the distribution of the secret key for the zero vector is efficiently sampleable given* $\mathsf{mpk}_{\mathsf{in}}$ *(and* mpk *will include* $\mathsf{mpk}_{\mathsf{in}}$*).*

Theorem 53 (Lemma 8)**.** *Construction 52 is function-hiding (Definition 7).*

Proof. We show $\mathsf{Exp}^0_{\mathrm{FH}} \approx \mathsf{Exp}^1_{\mathrm{FH}}$ by going through several hybrids responding to the challenges in different ways. For $\mathbf{z}_1 \in \mathbb{Z}_p^{n_{\text{pub}}}$ and $\mathbf{z}_2, \mathbf{z}_3 \in \mathbb{Z}_p^{n_{\text{priv}}}$, we write

$$\begin{aligned} & \mathsf{sk} \sim \mathbf{z}_1\ \mathbf{z}_2\ \mathbf{z}_3 \quad \text{for} \quad \mathsf{sk} \xleftarrow{\$} \mathsf{KeyGen}_{\mathsf{out}}\big([\![\mathbf{z}_1]\!]_2, \mathsf{Enc}_{\mathsf{in}}([\![\mathbf{z}_2]\!]_2, [\![\mathbf{z}_3]\!]_2)\big) \\ \text{and} \quad & \mathsf{ct} \sim \mathbf{z}_1\ \mathbf{z}_2\ \mathbf{z}_3 \quad \text{for} \quad \mathsf{ct} \xleftarrow{\$} \mathsf{Enc}_{\mathsf{out}}\big([\![\mathbf{z}_1]\!]_1, \mathsf{KeyGen}_{\mathsf{in}}([\![\mathbf{z}_2]\!]_1, [\![\mathbf{z}_3]\!]_1)\big). \end{aligned}$$

Consider the hybrids illustrated in Figure 12. Observe the vectors encoded by the inner scheme:

$$\begin{aligned} & & \text{in } \mathsf{sk}_{i,\mathsf{in}}\ (\text{in } \mathsf{ct}_{i,\mathsf{out}}) & & & & \text{in } \mathsf{ct}_{j,\mathsf{in}}\ (\text{in } \mathsf{sk}_{j,\mathsf{out}}) \\ \mathsf{H}_0 \to \mathsf{H}_1\text{:} \quad & \langle (& \mathbf{u}^0_{i,\text{priv}} & , & \underline{\mathbf{0}} &), (& \mathbf{v}^0_{j,\text{priv}} & , \mathbf{0} \to \mathbf{v}^1_{j,\text{priv}} &) \rangle, \\ \mathsf{H}_2 \to \mathsf{H}_3\text{:} \quad & \langle (& \underline{\mathbf{0}} & , & \mathbf{u}^1_{i,\text{priv}} &), (& \mathbf{v}^0_{j,\text{priv}} \to \mathbf{v}^1_{j,\text{priv}} & , \mathbf{v}^1_{j,\text{priv}} &) \rangle, \\ \mathsf{H}_4 \to \mathsf{H}_5\text{:} \quad & \langle (& \mathbf{u}^1_{i,\text{priv}} & , & \underline{\mathbf{0}} &), (& \mathbf{v}^1_{j,\text{priv}} & , \mathbf{v}^1_{j,\text{priv}} \to \mathbf{0} &) \rangle. \end{aligned}$$

For those three transitions, we only modify the ciphertext vectors of the inner scheme, and the modified part only multiplies with $\mathbf{0}$ (underlined) in the secret keys (of the inner scheme). Therefore, $\mathsf{H}_0 \approx \mathsf{H}_1$, $\mathsf{H}_2 \approx \mathsf{H}_3$ and $\mathsf{H}_4 \approx \mathsf{H}_5$ reduce to the IND-CPA security of the inner scheme. (The reduction will run $\mathsf{Setup}_{\mathsf{out}}$ on its own.)

hybrid	respond to the challenges by...			
$\mathsf{Exp}^0_{\mathrm{FH}} \equiv \mathsf{H}_0$	$\mathsf{sk}_j \sim$	$\mathbf{v}_{j,\mathrm{pub}}$	$\mathbf{v}^0_{j,\mathrm{priv}}$	$\boxed{\mathbf{0}}$
	$\mathsf{ct}_i \sim$	$\mathbf{u}^0_{i,\mathrm{pub}}$	$\mathbf{u}^0_{i,\mathrm{priv}}$	$\mathbf{0}$
H_1	$\mathsf{sk}_j \sim$	$\mathbf{v}_{j,\mathrm{pub}}$	$\mathbf{v}^0_{j,\mathrm{priv}}$	$\boxed{\mathbf{v}^1_{j,\mathrm{priv}}}$
	$\mathsf{ct}_i \sim$	$\boxed{\mathbf{u}^0_{i,\mathrm{pub}}}$	$\boxed{\mathbf{u}^0_{i,\mathrm{priv}}}$	$\boxed{\mathbf{0}}$
H_2	$\mathsf{sk}_j \sim$	$\mathbf{v}_{j,\mathrm{pub}}$	$\boxed{\mathbf{v}^0_{j,\mathrm{priv}}}$	$\mathbf{v}^1_{j,\mathrm{priv}}$
	$\mathsf{ct}_i \sim$	$\boxed{\mathbf{u}^1_{i,\mathrm{pub}}}$	$\boxed{\mathbf{0}}$	$\boxed{\mathbf{u}^1_{i,\mathrm{priv}}}$
H_3	$\mathsf{sk}_j \sim$	$\mathbf{v}_{j,\mathrm{pub}}$	$\boxed{\mathbf{v}^1_{j,\mathrm{priv}}}$	$\mathbf{v}^1_{j,\mathrm{priv}}$
	$\mathsf{ct}_i \sim$	$\mathbf{u}^1_{i,\mathrm{pub}}$	$\boxed{\mathbf{0}}$	$\boxed{\mathbf{u}^1_{i,\mathrm{priv}}}$
H_4	$\mathsf{sk}_j \sim$	$\mathbf{v}_{j,\mathrm{pub}}$	$\mathbf{v}^1_{j,\mathrm{priv}}$	$\boxed{\mathbf{v}^1_{j,\mathrm{priv}}}$
	$\mathsf{ct}_i \sim$	$\mathbf{u}^1_{i,\mathrm{pub}}$	$\boxed{\mathbf{u}^1_{i,\mathrm{priv}}}$	$\boxed{\mathbf{0}}$
$\mathsf{Exp}^1_{\mathrm{FH}} \equiv \mathsf{H}_5$	$\mathsf{sk}_j \sim$	$\mathbf{v}_{j,\mathrm{pub}}$	$\mathbf{v}^1_{j,\mathrm{priv}}$	$\boxed{\mathbf{0}}$
	$\mathsf{ct}_i \sim$	$\mathbf{u}^1_{i,\mathrm{pub}}$	$\mathbf{u}^1_{i,\mathrm{priv}}$	$\mathbf{0}$

Figure 12: Hybrids for proving the function-hiding property of Construction 52.

The other two transitions follow by the IND-CPA security of the outer scheme, taking advantage of the fact that the decryption of the inner scheme is computing the inner product of the secret key and the ciphertext — this means the outer decryption implicitly decrypts the inner instance. In H_1, the inner products (for the outer scheme) are

$$\begin{aligned}
&\left\langle \left(\mathbf{u}^0_{i,\mathrm{pub}}, \mathsf{KeyGen}_{\mathsf{in}}(\mathbf{u}^0_{i,\mathrm{priv}}, \mathbf{0})\right), \left(\mathbf{v}_{j,\mathrm{pub}}, \mathsf{Enc}_{\mathsf{in}}(\mathbf{v}^0_{j,\mathrm{priv}}, \mathbf{v}^1_{j,\mathrm{priv}})\right)\right\rangle \\
&= \langle \mathbf{u}^0_{i,\mathrm{pub}}, \mathbf{v}_{j,\mathrm{pub}}\rangle + \left\langle \mathsf{KeyGen}_{\mathsf{in}}(\mathbf{u}^0_{i,\mathrm{pub}}, \mathbf{0}), \mathsf{Enc}_{\mathsf{in}}(\mathbf{v}^0_{j,\mathrm{priv}}, \mathbf{v}^1_{j,\mathrm{priv}})\right\rangle \\
(\mathsf{Dec}_{\mathsf{in}} \text{ is inner product}) \quad &= \langle \mathbf{u}^0_{i,\mathrm{pub}}, \mathbf{v}_{j,\mathrm{pub}}\rangle + \langle (\mathbf{u}^0_{i,\mathrm{pub}}, \mathbf{0}), (\mathbf{v}^0_{j,\mathrm{priv}}, \mathbf{v}^1_{j,\mathrm{priv}})\rangle \\
&= \langle \mathbf{u}^0_{i,\mathrm{pub}}, \mathbf{v}_{j,\mathrm{pub}}\rangle + \langle \mathbf{u}^0_{i,\mathrm{priv}}, \mathbf{v}^0_{j,\mathrm{priv}}\rangle.
\end{aligned}$$

In H_2, the inner products are $\langle \mathbf{u}^1_{i,\mathrm{pub}}, \mathbf{v}_{j,\mathrm{pub}}\rangle + \langle \mathbf{u}^1_{i,\mathrm{priv}}, \mathbf{v}^1_{j,\mathrm{priv}}\rangle$, which remain unchanged by the constraint. Moreover, as far as the outer scheme is concerned, only the ciphertext vectors are changed, so $\mathsf{H}_1 \approx \mathsf{H}_2$ reduces to the IND-CPA security of the outer scheme. Same for H_3 and H_4. (The reduction will run $\mathsf{Setup}_{\mathsf{in}}$ on its own.)

By a hybrid argument, we conclude $\mathsf{Exp}^0_{\mathrm{FH}} \equiv \mathsf{H}_0 \approx \mathsf{H}_5 \equiv \mathsf{Exp}^1_{\mathrm{FH}}$. □

B Key Delegation

Key delegation in the context of ABE is the ability to securely create a more restrictive key from a secret key *without* the master secret key. This ability is characterized by a syntactical change of KeyGen and a generalized correctness requirement with respect to arbitrarily delegated keys. The experiments in the basic security notion (IND-CPA) are also adapted to take key delegation into account. In this section, we define the notion of key delegation and IND-CPA-DLG security for ABE, and show to how to tweak our basic KP-ABE scheme for ABPs to achieve these notions.

Definition 54 (key delegation)**.** *In an ABE for message space $\mathcal{M}$ and predicate space $\mathcal{P}$ that* supports delegation*, the key generation algorithm* KeyGen *takes a generalized syntax and the correctness requirement is strengthened:*

- $\mathsf{KeyGen}(1^\lambda, \mathsf{mpk}, \mathsf{sk}_{D:y_1,\ldots,y_D}, y_{D+1})$ *takes the master* ***public*** *key, a secret key $\mathsf{sk}_{D:y_1,\ldots,y_D}$, and a new policy $y_{D+1} \in Y_P$ as input, where P is the predicate in use. Here, D represents the number of times the key is delegated and $y_1, \ldots, y_D$ the previously imposed policies. The master secret key* msk *is alternatively denoted by sk_0, i.e., a key delegated for 0 times. The algorithm outputs a delegated key $\mathsf{sk}_{D+1:y_1,\ldots,y_{D+1}}$.*
- *Correctness holds with respect to arbitrarily delegated keys. Formally, for all $\lambda \in \mathbb{N}$, all $P \in \mathcal{P}_\lambda$, all $x \in X_P, g \in M_\lambda$, and all $D \in \mathbb{N}, y_1, \ldots, y_D \in Y_P$ such that $P(x, y_1) = \cdots = P(x, y_D) = 1$, it holds that*

$$\Pr\left[\begin{array}{l}(\mathsf{mpk}, \mathsf{msk} \stackrel{\mathrm{def}}{=\!=} \mathsf{sk}_0) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, P) \\ \quad d \in [D]\colon \quad \mathsf{sk}_d \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, \mathsf{mpk}, \mathsf{sk}_{d-1}, y_d) \\ \qquad\qquad \mathsf{ct} \xleftarrow{\$} \mathsf{Enc}(1^\lambda, \mathsf{mpk}, x, g)\end{array} : \mathsf{Dec}(1^\lambda, \mathsf{sk}_D, \mathsf{ct}) = g\right] = 1.$$

In the security game taking delegation into account [SW08], the adversary, instead of making key queries, makes key delegation and key revelation queries. The former creates a delegated key, and the latter reveals a delegated key to the adversary. Security should hold as long as the challenge attribute is not permitted by any individual secret key that is *revealed* to the adversary. Note that in IND-CPA-DLG, merely creating a secret key does not impose restrictions on the challenge attribute.

Definition 55 (IND-CPA-DLG [SW08])**.** *Let* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *be an ABE for message space* $\mathcal{M}$ *and predicate space* $\mathcal{P}$ *supporting delegation. The scheme is* IND-CPA-DLG secure *if* $\mathsf{Exp}^0_{\text{CPA-DLG}} \approx \mathsf{Exp}^1_{\text{CPA-DLG}}$*, where* $\mathsf{Exp}^b_{\text{CPA-DLG}}$ *is defined as follows:*

- **Setup.** Run $\mathcal{A}(1^\lambda)$ and receive a predicate $P \in \mathcal{P}_\lambda$ from it. Run $(\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, P)$, define $\mathsf{sk}_0 = \mathsf{msk}$, $Z_0 = ()$, and let $H \leftarrow \{0\}, R \leftarrow \varnothing$. Return mpk to $\mathcal{A}$.

  Note: *Elements in* H *are* key handles*, each of which corresponds to a (delegated) secret key. The set* R *keeps track of the handles whose corresponding keys are revealed. For each handle* $h \in H$*, the list* $Z_h \in Y_P^*$ *consists of the policies imposed on* sk_h*.*

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

  - $\mathcal{A}$ can submit $h \in H, y \in Y_P$ to delegate sk_h with y. Upon this query, let $h' \leftarrow |H|$ and set $H \leftarrow H \cup \{h'\}$. Define $Z_{h'} = (Z_h, y)$ and run $\mathsf{sk}_{h'} \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, \mathsf{mpk}, \mathsf{sk}_h, y)$. Resume the experiment without returning any information to $\mathcal{A}$.
  - $\mathcal{A}$ can submit $h \in H$ to have sk_h revealed. Upon this query, let $R \leftarrow R \cup \{h\}$ and return sk_h to $\mathcal{A}$.

- **Challenge.** $\mathcal{A}$ submits $x \in X_P, g_0, g_1 \in M_\lambda$. Run $\mathsf{ct} \xleftarrow{\$} \mathsf{Enc}(1^\lambda, \mathsf{mpk}, x, g_b)$ and return ct to $\mathcal{A}$.

- **Query II.** Same as Query I.

- **Guess.** $\mathcal{A}$ outputs a bit b'. The outcome of the experiment is b' if for all $h \in R$, there exists y in Z_h such that $P(x, y) = 0$. Otherwise, the outcome is 0.

Our approach to ABE supporting delegation builds on top of the basic ABE and adds two ingredients: *i*) secret-sharing the pads to support conjunction of policies; *ii*) key-homomorphism from IPFE to support generating delegated keys without master secret key. We first discuss them separately, then put them together to obtain the ABE scheme supporting delegation.

B.1 1-ABE Supporting Conjunctions

To construct ABE supporting delegation, secret keys must be capable of expressing arbitrary conjunction of policies in Y_P. We start by showing how to tweak 1-ABE (Construction 24) to achieve this. To begin, we make these modifications to 1-ABE and its security definition:

- A 1-ABE supporting conjunction for predicate space $\mathcal{P}$ has the same syntax as a usual 1-ABE, except that KeyGen now takes in an arbitrary number of policies $y_1, \ldots, y_D \in Y_P$, and generates a key for them, and that Dec must decapsulate the pad only when $P(x, y_1) = \cdots = P(x, y_D) = 1$.

- In 1-key 1-ciphertext security experiment, the adversary submits an arbitrary number of policies (in one shot) to request a secret key for the conjunction of them. The restriction on the challenge attribute is that there exists $y_d \in \{y_1, \ldots, y_D\}$ (among the submitted policies) such that $P(x, y_d) = 0$.

Tweaking Construction 24. To support conjunctions, KeyGen secret-shares the pad μ to encapsulate into $\mu_1, \ldots, \mu_D$ among the policies, and garbles y_d with pad μ_d. To decapsulate the pad, Dec first recovers each share μ_d in the exponent, then sums them to get μ.

Formally, recall that in Construction 24, the predicate space is induced by some function class $\mathcal{F}$ with AKGS $(\mathsf{Garble}, \mathsf{Eval})$:

$$X_n = \mathbb{Z}_p^n, \quad Y_n = \{f_{\neq 0}, f_{=0} \mid f \in \mathcal{F}, f : \mathbb{Z}_p^n \to \mathbb{Z}_p\},$$
$$\mathcal{P} = \{P_n : X_n \times Y_n \to \{0,1\}, (\mathbf{x}, y) \mapsto y(\mathbf{x}) \mid n \in \mathbb{N}\}.$$

$\mathsf{Setup}(1^n)$ simply runs $\mathsf{msk} \xleftarrow{\$} \mathsf{IPFE.Setup}(1^n)$ to set up a function-hiding secret-key IPFE scheme as before.

The tweaked version of $\mathsf{KeyGen}(\mathsf{msk}, y_1, \ldots, y_D, \mu)$ first samples $\mu_1, \ldots, \mu_{D-1}, \eta_1, \ldots, \eta_D \xleftarrow{\$} \mathbb{Z}_p$ and sets $\mu_D \leftarrow \mu - \sum_{d=1}^{D-1} \mu_d$. Next, it garbles the functions underlying each policy, i.e., for $d \in [D]$:

$$\begin{cases} \alpha_d \leftarrow \mu_d, \quad \beta_d \leftarrow 0, & \text{if } y_d = f_{d,\neq 0}; \\ \alpha_d \leftarrow \eta_d, \quad \beta_d \leftarrow \mu_d, & \text{if } y_d = f_{d,=0}; \end{cases} \qquad (\mathbf{L}_{d,1}, \ldots, \mathbf{L}_{d,m_d}) \xleftarrow{\$} \mathsf{Garble}(f_d, \alpha_d, \beta_d).$$

Then, the algorithm generates IPFE secret keys $\mathsf{isk}_{d,j} \xleftarrow{\$} \mathsf{IPFE.KeyGen}(\mathsf{msk}, \mathbf{v}_{d,j})$ to encrypt the coefficient vectors:

vector	const	coef_i	sim_1	$\mathsf{sim}_\star$
$\mathbf{v}_{d,j}$	$\mathbf{L}_{d,j}[\mathsf{const}]$	$\mathbf{L}_{d,j}[\mathsf{coef}_i]$	0	0

It returns $\mathsf{sk} = (y_1, \mathsf{isk}_{1,1}, \ldots, \mathsf{isk}_{1,m_1}; \ \ldots; \ y_D, \mathsf{isk}_{D,1}, \ldots, \mathsf{isk}_{D,m_D})$ as the secret key encapsulating μ.

Enc remains the same, and an ABE ciphertext consists of the attribute $\mathbf{x}$ itself and an IPFE ciphertext encrypting $(1, \mathbf{x}, 0, 0)$. The decapsulation algorithm $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$ first parses

$$\mathsf{sk} \text{ as } (y_1, \mathsf{isk}_{1,1}, \ldots, \mathsf{isk}_{1,m_1}; \ \ldots; \ y_D, \mathsf{isk}_{D,1}, \ldots, \mathsf{isk}_{D,m_D}),$$
$$\text{and } \mathsf{ct} \text{ as } (\mathbf{x}, \mathsf{ict}).$$

It returns $\perp$ if $y_d(\mathbf{x}) = 0$ for some $d \in [D]$. Otherwise, it recovers μ_d's in the exponent, i.e., for all $d \in [D]$:

$$\text{for } j \in [m_d]: \quad [\![\ell_{j,d}]\!]_{\mathrm{T}} \leftarrow \mathsf{IPFE.Dec}(\mathsf{isk}_{j,d}, \mathsf{ict}),$$
$$[\![\mu'_d]\!]_{\mathrm{T}} \leftarrow \begin{cases} \frac{1}{f_d(\mathbf{x})} \mathsf{Eval}(f_d, \mathbf{x}, [\![\ell_{d,1}]\!]_{\mathrm{T}}, \ldots, [\![\ell_{d,m_d}]\!]_{\mathrm{T}}), & \text{if } y_d = f_{d,\neq 0}; \\ \mathsf{Eval}(f_d, \mathbf{x}, [\![\ell_{d,1}]\!]_{\mathrm{T}}, \ldots, [\![\ell_{d,m_d}]\!]_{\mathrm{T}}), & \text{if } y_d = f_{d,=0}. \end{cases}$$

The algorithm returns $\sum_{d=1}^{D} [\![\mu'_d]\!]_{\mathrm{T}}$ as the decapsulated pad.

For correctness, following the same argument for Construction 24, we know that $\mu'_d = \mu_d$ for all $d \in [D]$, whence $\sum_{d=1}^{D} \mu'_d = \mu$.

Security Proof. We give the idea of the security proof and leave the formalism to the reader. Recall that in (the more interesting case of) the security proof for Construction 24 (Theorem 25), we first hardwire the reversely sampled first label into ict, then replace each subsequent label function encoded in isk_j by a randomly simulated label, until all the other labels are simulated and the first label is reversely sampled, at which point $\mathsf{ict}, \mathsf{isk}_j$'s are independent of the encapsulated pad.

The security of the tweaked version requires that the encapsulated pad be hidden when $y_d(\mathbf{x}) = 0$ for some $d \in [D]$, where $y_1, \ldots, y_D, \mathbf{x}$ are chosen adaptively by the adversary. We first consider a *selective-violation* version of the security experiments, where the adversary commits to an index

$d^* \in \mathbb{N}$ such that $y_{d^*}(\mathbf{x}) = 0$ for its later choices of $y_1, \ldots, y_D, \mathbf{x}$. The proof of selective-violation security follows by the same argument for Theorem 25 — we simulate the labels for f_{d^*} to show that the $(d^*)^{\text{th}}$ share μ_{d^*} is hidden, and so is the encapsulated pad.

Now, we can prove 1-key 1-ciphertext security of the tweaked construction by a random guessing argument and reduction to selective-violation security. For any adversary $\mathcal{A}$ against 1-key 1-ciphertext security, we construct $\mathcal{B}$ against selective-violation security. Suppose $\mathcal{A}$ makes the key query for at most D policies. $\mathcal{B}$ first guesses $d^* \xleftarrow{\$} [D]$, uses it as the commitment, and then runs $\mathcal{A}$. The new adversary $\mathcal{B}$ aborts if d^* is not *the smallest* d such that $y_d(\mathbf{x}) = 0$ for $y_1, \ldots, y_D, \mathbf{x}$ chosen adaptively by $\mathcal{A}$. This reduction loses a factor of D in the advantage.

B.2 Perfectly Key-Homomorphic IPFE

In our KP-ABE supporting delegation, the secret key has structures similar to that in the usual KP-ABE (an IPFE key for computing the sum of the pads) with the additional features in the tweaked 1-ABE (one set of IPFE keys for each policy for computing the labels). To be able to delegate, one must be able to generate new IPFE secret keys using the (new) master public key. Furthermore, we will *rerandomize* the delegated key so that it is identically distributed to a freshly generated one, with randomness independent of the key being delegated — this ensures that a delegated key leaks no information about the randomness in the key from which it is created, thus making security proof simpler. We rely on an additional property, called *key-homomorphism*, of the slotted IPFE scheme to achieve the above.

Definition 56 (perfect key-homomorphism)**.** *A slotted IPFE scheme* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{SlotEnc})$ *based on pairing groups of order* $p = p(\lambda)$ *is* (perfectly) key-homomorphic *if it is endowed with an efficient algorithm* $\mathsf{Subtract}(1^\lambda, \mathsf{mpk}, \mathsf{sk}_1, \mathsf{sk}_2)$ *that takes the master public key and two secret keys* $\mathsf{sk}_1, \mathsf{sk}_2$ *as input, and outputs a new secret key* sk_3 *encrypting the difference of the vectors encrypted in* $\mathsf{sk}_1, \mathsf{sk}_2$*. Formally, for all* $\lambda \in \mathbb{N}$*, all disjoint index sets* $\mathfrak{s}_{\text{pub}}, \mathfrak{s}_{\text{priv}}$ *(let* $\mathfrak{s} = \mathfrak{s}_{\text{pub}} \cup \mathfrak{s}_{\text{priv}}$*) and all* $\mathbf{v}_1, \mathbf{v}_2 \in \mathbb{Z}_{p(\lambda)}^{\mathfrak{s}}$*, the following distributions are identical:*

$$\left\{ \begin{array}{rl} (\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, \mathfrak{s}_{\text{pub}}, \mathfrak{s}_{\text{priv}}) & \\ \mathsf{sk}_1 \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, \mathsf{msk}, [\![\mathbf{v}_1]\!]_2) & \\ \mathsf{sk}_2 \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, \mathsf{msk}, [\![\mathbf{v}_2]\!]_2) & : (\mathsf{mpk}, \mathsf{msk}, \mathsf{sk}_1, \mathsf{sk}_2, \mathsf{sk}_3) \\ \mathsf{sk}_3 \xleftarrow{\$} \mathsf{Subtract}(1^\lambda, \mathsf{mpk}, \mathsf{sk}_1, \mathsf{sk}_2) & \end{array} \right\},$$

$$\left\{ \begin{array}{rl} (\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, \mathfrak{s}_{\text{pub}}, \mathfrak{s}_{\text{priv}}) & \\ \mathsf{sk}_1 \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, \mathsf{msk}, [\![\mathbf{v}_1]\!]_2) & \\ \mathsf{sk}_2 \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, \mathsf{msk}, [\![\mathbf{v}_2]\!]_2) & : (\mathsf{mpk}, \mathsf{msk}, \mathsf{sk}_1, \mathsf{sk}_2, \mathsf{sk}_3) \\ \mathsf{sk}_3 \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, \mathsf{msk}, [\![\mathbf{v}_1 - \mathbf{v}_2]\!]_2) & \end{array} \right\}.$$

Note that a key-homomorphic and function-hiding IPFE must use a randomized subtraction algorithm. For a key-homomorphic IPFE, given secret keys $\mathsf{sk}_1, \mathsf{sk}_2$ and coefficient $k \in \mathbb{Z}_p$, we abuse the notation $\mathsf{sk}_1 + k\mathsf{sk}_2$ for the implicit application of $\mathsf{Subtract}$ to compute a secret key for the linearly combined vector.

Achieving Key-Homomorphism. It turns out the IPFE scheme in Construction 52 can be made prefectly key-homomorphic by publishing some additional components in mpk. First, notice that the secret key in that scheme is linear in $(\mathbf{v}_{\text{pub}}, \mathbf{v}_{\text{priv}}, \mathbf{s})$, where $\mathbf{v}_{\text{pub}}, \mathbf{v}_{\text{priv}}$ are the public/private part of the vector and $\mathbf{s}$ is the randomness used by the (inner) scheme. Therefore, to subtract two

keys, it suffices to subtract the group elements and add a random key of $\mathbf{0}$. The internal workings opened up, Setup of the IPFE scheme samples

$$\mathbf{A}_{\mathsf{in}}, \mathbf{A}_{\mathsf{out}} \xleftarrow{\$} \mathbb{Z}_p^{k\times(k+1)}, \quad \mathbf{W}_{\mathsf{in}} \xleftarrow{\$} \mathbb{Z}_p^{(k+1)\times 2n_{\mathrm{priv}}}, \quad \mathbf{W}_{\mathsf{out}} \xleftarrow{\$} \mathbb{Z}_p^{(k+1)\times(n_{\mathrm{pub}}+(k+1)+2n_{\mathrm{priv}})}.$$

Unwinding the abstraction, a secret key for $\mathbf{0}$ in the scheme has the following form:

$$\begin{aligned}\mathsf{sk} &= \mathsf{KeyGen}_{\mathsf{out}}([\![\mathbf{0}_{n_{\mathrm{pub}}}]\!]_2, \mathsf{Enc}_{\mathsf{in}}([\![\mathbf{0}_{2n_{\mathrm{priv}}}]\!]_2)) = \mathsf{KeyGen}_{\mathsf{out}}([\![\mathbf{0}_{n_{\mathrm{pub}}}]\!]_2, [\![\mathbf{A}_{\mathsf{in}}^\mathsf{T}\mathbf{s}]\!]_2, [\![\mathbf{W}_{\mathsf{in}}^\mathsf{T}\mathbf{A}_{\mathsf{in}}^\mathsf{T}\mathbf{s}]\!]_2) \\ &= \left[\!\!\left[-\mathbf{W}_{\mathsf{out}} \begin{pmatrix} \mathbf{0}_{n_{\mathrm{pub}}} \\ \mathbf{A}_{\mathsf{in}}^\mathsf{T}\mathbf{s} \\ \mathbf{W}_{\mathsf{in}}^\mathsf{T}\mathbf{A}_{\mathsf{in}}^\mathsf{T}\mathbf{s} \end{pmatrix} \right]\!\!\right]_2 = \left[\!\!\left[\begin{pmatrix} -\mathbf{W}_{\mathsf{out}}\mathbf{Z} \\ \mathbf{Z} \end{pmatrix} \right]\!\!\right]_2 \mathbf{s},\end{aligned}$$

where $\mathbf{Z} = \begin{pmatrix} \mathbf{0}_{n_{\mathrm{pub}}\times k} \\ \mathbf{A}_{\mathsf{in}}^\mathsf{T} \\ \mathbf{W}_{\mathsf{in}}^\mathsf{T}\mathbf{A}_{\mathsf{in}}^\mathsf{T} \end{pmatrix} \in \mathbb{Z}_p^{(n_{\mathrm{pub}}+(k+1)+2n_{\mathrm{priv}})\times k}$ and $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_p^k$. Therefore, it suffices to set the master public key as (the newly added part is highlighted in the box)

$$\mathsf{mpk} = \Big(\underbrace{[\![\mathbf{A}_{\mathsf{out}}]\!]_1, [\![\mathbf{A}_{\mathsf{out}}\mathbf{W}_{\mathsf{out}}]\!]_1}_{\mathsf{mpk}_{\mathsf{out}}}, \boxed{[\![-\mathbf{W}_{\mathsf{out}}\mathbf{Z}]\!]_2, [\![\mathbf{Z}]\!]_2} \Big).$$

The master secret key stays the same as in Construction 52. $\mathsf{Subtract}(\mathsf{mpk}, \mathsf{sk}_1, \mathsf{sk}_2)$ computes

$$\underbrace{\mathsf{sk}_1 - \mathsf{sk}_2}_{\text{group operation}} + \begin{pmatrix} [\![-\mathbf{W}_{\mathsf{out}}\mathbf{Z}]\!]_2 \\ [\![\mathbf{Z}]\!]_2 \end{pmatrix} \mathbf{s} \quad \text{with} \quad \mathbf{s} \xleftarrow{\$} \mathbb{Z}_p^k$$

as the resultant key.

Security Revisited. We observe that the newly published information does not jeopardize the function-hiding property. Again we just give the ideas and leave the formalism to the reader.

Recall that in the security proof of Construction 52 (Theorem 53), between each pair of neighboring hybrids, we rely on either the IND-CPA security of the inner scheme or that of the outer scheme. It suffices to show that in either case, the reduction algorithm can efficiently obtain the newly added components, $[\![-\mathbf{W}_{\mathsf{out}}\mathbf{Z}]\!]_2$ and $[\![\mathbf{Z}]\!]_2$. First, note that the entries of $\mathbf{Z}$ are either 0 or from $\mathbf{A}_{\mathsf{in}}, \mathbf{A}_{\mathsf{in}}\mathbf{W}_{\mathsf{in}}$, the master public key of the inner scheme.

If the indistinguishability is reduced to the IND-CPA security of the inner scheme, the reduction algorithm receives $\mathsf{mpk}_{\mathsf{in}}$ from the CPA experiments of the inner scheme, from which it can form $[\![\mathbf{Z}]\!]_2$. It sets up the outer scheme by itself, so it knows $\mathbf{W}_{\mathsf{out}}$ and can compute $-\mathbf{W}_{\mathsf{out}}[\![\mathbf{Z}]\!]_2$ efficiently.

If the indistinguishability is reduced to the IND-CPA security of the outer scheme, the reduction algorithm sets up the inner scheme itself and knows $\mathbf{A}_{\mathsf{in}}, \mathbf{W}_{\mathsf{in}}$ thus $\mathbf{Z}$. It also receives $\mathsf{mpk}_{\mathsf{out}}$ from the CPA experiments of the outer scheme. The reduction algorithm then requests secret keys for each column of $[\![\mathbf{Z}]\!]_2$ from the outer scheme, which gives it exactly $[\![-\mathbf{W}_{\mathsf{out}}\mathbf{Z}]\!]_2$. This will not violate the constraints of the CPA experiments, because any vector for which the reduction algorithm requests a ciphertext will have inner product 0 with the columns of $\mathbf{Z}$ — recall that each column of $\mathbf{Z}$ is a possible vector fed into $\mathsf{KeyGen}_{\mathsf{out}}$ when generating a secret key of $\mathbf{0}$, that the reduction algorithm only requests ciphertexts for vectors that are potentially fed into $\mathsf{Enc}_{\mathsf{out}}$ when generating ciphertexts in the slotted scheme, and that the inner product must be 0 by the correctness of the slotted scheme.

B.3 KP-ABE Supporting Delegation

We now combine secret-sharing of the pads and key-homomorphism of IPFE to obtain ABE supporting delegation. Recall that in our basic ABE scheme (Construction 26), we use a slotted IPFE as the underlying primitive. An ABE ciphertext consists of the padded message, the attribute, and an IPFE ciphertext, i.e., $\mathsf{ct} = ([\![h]\!]_{\mathrm{T}} + g, \mathbf{x}, \mathsf{ict})$, where ict encrypts $\mathbf{u}$, consisting of h and random multiples of $(1, \mathbf{x})$. An ABE secret key consists of several IPFE secret keys, $\mathsf{sk}_y = (\mathsf{isk}_{\mathrm{pad}}, y, \mathsf{isk}_1, \ldots, \mathsf{isk}_m)$, where isk_j encodes the label functions $\mathbf{v}_j$ in the garblings of the function f underlying $y = f_{\neq 0}, f_{=0}$, and $\mathsf{isk}_{\mathrm{pad}}$ encodes a vector $\mathbf{v}_{\mathrm{pad}}$, decrypting ict to the sum of the pads. We refer to $\mathsf{isk}_{\mathrm{pad}}$ as the **pad key**, and isk_j's as the **label keys**. The vectors are summarized below:

vector	pad	const^t	coef_i^t	in $\mathfrak{s}_{\mathrm{priv}}$
$\mathbf{v}_{\mathrm{pad}}$	1	$\boldsymbol{\mu}[t]$	0	0
$\mathbf{v}_j$	0	$\mathbf{L}_j^t[\mathsf{const}]$	$\mathbf{L}_j^t[\mathsf{coef}_i]$	
$\mathbf{u}$	h	$\mathbf{s}[t]$	$\mathbf{s}[t]\mathbf{x}[i]$	0

Expressing conjunction is done by having multiple sets of *label keys*, each for a share of the pad in the *pad key*, as demonstrated in the tweaked version of 1-ABE. Delegating a key requires creating new *label keys* with non-zero (potentially arbitrary) values at $\mathsf{const}^t, \mathsf{coef}_i^t$ without using the master secret key. A natural idea is to publish a set of keys for a basis for these indices (called the **basis keys**) in the (new) master public key and use key-homomorphism to combine them as needed. These keys do not help computing the pad h by simply using them to decrypt ict.

Naïve Attempt. A naïve attempt is to publish the keys for the natural (standard) basis in these indices. However, doing so ruins the security proof (see Theorem 27). To see this, recall that the first step of the security proof is to move the computation of the labels from the public slot of ict to the private slot, and $\mathbf{s}$ from ict to the isk's. Let us consider adding the standard basis keys for const^t. In the security proof, when we modify ict, we must also modify the basis keys to keep the inner products the same. Pragmatically, the vectors will transform as follows:

vector	pad	const^t	coef_i^t	in $\mathfrak{s}_{\mathrm{priv}}$
$\mathbf{e}_{\mathsf{const}^{t'}}$	0	$\mathbb{1}[t = t']$	0	0
$\mathbf{v}_{\mathrm{pad}}$	1	$\boldsymbol{\mu}[t]$	0	
$\mathbf{v}_j$	0	$\mathbf{L}_j^t[\mathsf{const}]$	$\mathbf{L}_j^t[\mathsf{coef}_i]$	
$\mathbf{u}$	h	$\mathbf{s}[t]$	$\mathbf{s}[t]\mathbf{x}[i]$	0

$\longrightarrow$

pad	const^t	coef_i^t	const	coef_i	group
0	$\mathbb{1}[t = t']$	0	$\mathbf{s}[t']$	0	G_2
1	$\boldsymbol{\mu}[t]$	0	$\langle \boldsymbol{\mu}, \mathbf{s} \rangle$	0	
0	$\mathbf{L}_j^t[\mathsf{const}]$	$\mathbf{L}_j^t[\mathsf{coef}_i]$	$\cdots$	$\cdots$	
h	0	0	1	$\mathbf{x}[i]$	G_1

Here, the expression for the label keys are suppressed for brevity — they are the linear combinations of $\mathbf{L}_j^t$'s with coefficients $\mathbf{s}$ (combined over $t \in [k]$).

The next step in the security proof is to replace the pad $\langle \boldsymbol{\mu}, \mathbf{s} \rangle$ and the label functions in the private slot by independent randomly generated ones. For this, we rely on the MDDH_k assumption in G_2:

$$[\![\mathbf{A}, \mathbf{s}^\top \mathbf{A}]\!]_2 \approx [\![\mathbf{A}, \mathbf{c}^\top]\!]_2, \text{ where } \mathbf{A} \xleftarrow{\$} \mathbb{Z}_p^{k \times N}, \mathbf{s} \xleftarrow{\$} \mathbb{Z}_p^k, \mathbf{c} \xleftarrow{\$} \mathbb{Z}_p^N.$$

Here, N counts the total amount of randomness used in the secret keys, and $\mathbf{A}$ consists of the randomness used in the public slot garblings. However, this assumption can no longer be used if we want to publish the basis keys as above, because the right-hand side distribution does not contain $[\![\mathbf{s}]\!]_2$, which is necessary to generate the basis keys.

Random Basis. To mitigate the above issue, we notice that for the purpose of delegation, it suffices to publish a set of basis keys for *any* basis. The delegation procedure can use the freshly generated label functions to combine the basis keys, which corresponds to implicitly applying an invertible linear transformation over the label functions. The combined keys still follow the correct distribution (having the label functions from k independent garblings), thanks to the linearity of AKGS Garble.

If we publish the basis keys for a random basis $\mathbf{B} \in \mathbb{Z}_p^{k\times k}$, it is valid to invoke MDDH_k in G_2 to rerandomize the garblings in the private slot. In the first hybrid, the basis keys will have values from $\mathbf{B}$ in the public slot, and values from $\mathbf{s}^\mathsf{T}\mathbf{B}$ in the private slot. Now, we can apply MDDH_k to argue

$$[\![\mathbf{A}, \mathbf{B}, \mathbf{s}^\mathsf{T}\mathbf{A}, \mathbf{s}^\mathsf{T}\mathbf{B}]\!]_2 \approx [\![\mathbf{A}, \mathbf{B}, \mathbf{c}^\mathsf{T}, \mathbf{d}^\mathsf{T}]\!]_2, \text{ where } \mathbf{A} \xleftarrow{\$} \mathbb{Z}_p^{k\times N}, \mathbf{B} \xleftarrow{\$} \mathbb{Z}_p^{k\times k}, \mathbf{s} \xleftarrow{\$} \mathbb{Z}_p^{k}, \mathbf{c} \xleftarrow{\$} \mathbb{Z}_p^{N}, \mathbf{d} \xleftarrow{\$} \mathbb{Z}_p^{k},$$

and in the second hybrid, the basis keys will have values from $\mathbf{d}^\mathsf{T}$ in the private slot.

Construction 57 (KP-ABE supporting delegation)**.** We describe the construction for any fixed value of the security parameter λ and suppress the appearance of λ below for simplicity of notations. Let $(\mathsf{Garble}, \mathsf{Eval})$ be an AKGS for a function class $\mathcal{F}$, $\mathcal{G}$ pairing groups of order p such that MDDH_k holds in G_2, and $(\mathsf{IPFE.Setup}, \mathsf{IPFE.KeyGen}, \mathsf{IPFE.Enc}, \mathsf{IPFE.Dec}, \mathsf{IPFE.SlotEnc}, \mathsf{IPFE.Subtract})$ a *key-homomorphic* slotted IPFE based on $\mathcal{G}$. We construct an ABE scheme *supporting delegation* for message space $M = G_\mathrm{T}$, the target group of the pairing, and the predicate space $\mathcal{P}$ induced by $\mathcal{F}$:

$$X_n = \mathbb{Z}_p^n, \quad Y_n = \{f_{\neq 0}, f_{=0} \mid f \in \mathcal{F}, f : \mathbb{Z}_p^n \to \mathbb{Z}_p\},$$
$$\mathcal{P} = \{P_n : X_n \times Y_n \to \{0,1\}, (\mathbf{x}, y) \mapsto y(\mathbf{x}) \mid n \in \mathbb{N}\}.$$

For a better exposition, we simultaneously consider two ABE schemes. The helper scheme, $(\mathsf{Setup}_{\text{1-shot}}, \mathsf{KeyGen}_{\text{1-shot}}, \mathsf{Enc}, \mathsf{Dec})$, supports *conjunction* with a key generation algorithm conforming to the basic ABE syntax.[19] The main scheme, $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$, supports *delegation* and shares much of its codebase with the helper. They operate as follows:

- $\mathsf{Setup}_{\text{1-shot}}(1^n)$ takes the attribute length in unary (i.e., P_n is encoded as 1^n) as input. It generates IPFE master public/secret key pair $(\mathsf{impk}, \mathsf{imsk}) \xleftarrow{\$} \mathsf{IPFE.Setup}(\mathfrak{s}_\mathrm{pub}, \mathfrak{s}_\mathrm{priv})$ for the following slots:

  $$\mathfrak{s}_\mathrm{pub} = \{\mathsf{pad}, \mathsf{const}^t, \mathsf{coef}_i^t \mid t \in [k], i \in [n]\},$$
  $$\mathfrak{s}_\mathrm{priv} = \{\mathsf{const}, \mathsf{coef}_i, \mathsf{sim}_1, \mathsf{sim}_\star \mid i \in [n]\}.$$

  The algorithm samples $\mathbf{B} \xleftarrow{\$} \mathbb{Z}_p^{k\times k}$ conditioned on $\mathbf{B}$ being invertible and sets the basis vectors as follows:

vector	pad	const^t	coef_i^t	in $\mathfrak{s}_\mathrm{priv}$
$\mathbf{b}_{\mathsf{const}^{t'}}$	0	$\mathbf{B}[t,t']$	0	0
$\mathbf{b}_{\mathsf{coef}_{i'}^{t'}}$	0	0	$\mathbf{B}[t,t']\mathbb{1}[i=i']$	

  It then generates the *basis keys* as

  $$\mathsf{bsk}_z \xleftarrow{\$} \mathsf{IPFE.KeyGen}(\mathsf{imsk}, [\![\mathbf{b}_{\mathsf{const}^t}]\!]_2) \text{ for } z \in \{\mathsf{const}^t, \mathsf{coef}_i^t \mid i \in [n], t \in [k]\}.$$

[19] To keep the formalism happy, the predicates should be changed to work with X_n and Y_n^* by mapping $(\mathbf{x}, (y_1, \ldots, y_D))$ to the conjunction of $y_d(\mathbf{x})$'s.

The algorithm returns $\mathsf{mpk} = \left(\mathsf{impk}, (\mathsf{bsk}_z)_{z\in\{\mathsf{coef}^t,\mathsf{const}_i^t \mid i\in[n],t\in[k]\}}\right)$ and $\mathsf{msk}_{\text{1-shot}} = \mathsf{imsk}$.

Note: *This procedure generates the basis keys on top of the basic ABE scheme. As far as the helper scheme (supporting conjunction without delegation) is concerned, it is not necessary to include the basis keys. It is also not harmful, and we will be able to reuse the IND-CPA security of the helper scheme for proving the IND-CPA-DLG security of the main scheme if the basis keys are included.*

- $\mathsf{KeyGen}_{\text{1-shot}}(\mathsf{msk}_{\text{1-shot}}, y_1, \ldots, y_D)$ takes in the master secret key $\mathsf{msk}_{\text{1-shot}} = \mathsf{imsk}$ and D policies $y_1, \ldots, y_D \in Y_n$ as input. It starts by garbling the functions underlying the policies, i.e., for all $d \in [D]$:

$$\boldsymbol{\mu}_d, \boldsymbol{\eta}_d \xleftarrow{\$} \mathbb{Z}_p^k, \quad \begin{cases} \boldsymbol{\alpha}_d \leftarrow \boldsymbol{\mu}_d, & \boldsymbol{\beta}_d \leftarrow \mathbf{0}, \quad \text{if } y_d = f_{d,\neq 0}; \\ \boldsymbol{\alpha}_d \leftarrow \boldsymbol{\eta}_d, & \boldsymbol{\beta}_d \leftarrow \boldsymbol{\mu}_d, \quad \text{if } y_d = f_{d,=0}; \end{cases}$$
$$\text{for } t \in [k]: \quad (\mathbf{L}_{d,1}^t, \ldots, \mathbf{L}_{d,m_d}^t) \leftarrow \mathsf{Garble}(f_d, \boldsymbol{\alpha}_d[t], \boldsymbol{\beta}_d[t]; \mathbf{r}_{d,t}).$$

The algorithm sets $\boldsymbol{\mu} = \sum_{d=1}^{D} \boldsymbol{\mu}_d$ and assigns $\mathbf{v}_{\text{pad}}, \mathbf{v}_{d,j}$ for $d \in [D], j \in [m_d]$ as follows:

vector	pad	const^t	coef_i^t	in $\mathfrak{s}_{\text{priv}}$
$\mathbf{v}_{\text{pad}}$	1	$\boldsymbol{\mu}[t]$	0	0
$\mathbf{v}_{d,j}$	0	$\mathbf{L}_{d,j}^t[\mathsf{const}]$	$\mathbf{L}_{d,j}^t[\mathsf{coef}_i]$	

It generates the IPFE secret keys for those vectors and returns the secret key:

$$\mathsf{isk}_{\text{pad}} \xleftarrow{\$} \mathsf{IPFE.KeyGen}(\mathsf{imsk}, [\![\mathbf{v}_{\text{pad}}]\!]_2),$$
$$\text{for } d \in [D], j \in [m_d]: \quad \mathsf{isk}_{d,j} \xleftarrow{\$} \mathsf{IPFE.KeyGen}(\mathsf{imsk}, [\![\mathbf{v}_{d,j}]\!]_2),$$
$$\mathsf{sk}_{D:y_1,\ldots,y_D} = (\mathsf{isk}_{\text{pad}};\ y_1, \mathsf{isk}_{1,1}, \ldots, \mathsf{isk}_{1,m_1};\ \ldots;\ y_D, \mathsf{isk}_{D,1}, \ldots, \mathsf{isk}_{D,m_D}).$$

Note: $\mathsf{KeyGen}_{\text{1-shot}}$ *is a mixture of the basic ABE* KeyGen *(Construction 26) and the tweaked 1-ABE* KeyGen *(Appendix B.1) to support conjunctions.*

- $\mathsf{Setup}(1^n)$ takes the attribute length as input. It runs $(\mathsf{mpk}, \mathsf{msk}_{\text{1-shot}}) \xleftarrow{\$} \mathsf{Setup}_{\text{1-shot}}(1^n)$ and generates the master secret key as $\mathsf{msk} = \mathsf{KeyGen}_{\text{1-shot}}(\mathsf{msk}_{\text{1-shot}})$.

Note: *The master secret key is simply a secret key with* 0 *policies (and thus pads* $\mathbf{0}$*). The only component of* msk *is its* ***pad key****, which will decrypt any* ict *to the pad used to hide the message, giving unconditional decryption of ABE ciphertexts.*

- $\mathsf{KeyGen}(\mathsf{mpk}, \mathsf{sk}_{D:y_1,\ldots,y_D}, y_{D+1})$ takes the master public key, a secret key, and a policy $y_{D+1} \in Y_n$ as input. It parses mpk as $\left(\mathsf{impk}, (\mathsf{bsk}_z)_{z\in\{\mathsf{coef}^t,\mathsf{const}_i^t \mid i\in[n],t\in[k]\}}\right)$ and

$$\mathsf{sk}_{D:y_1,\ldots,y_D} \text{ as } \left(\mathsf{isk}_{\text{pad}}^{\text{old}};\ y_1, \mathsf{isk}_{1,1}^{\text{old}}, \ldots, \mathsf{isk}_{1,m_1}^{\text{old}};\ \ldots;\ y_D, \mathsf{isk}_{D,1}^{\text{old}}, \ldots, \mathsf{isk}_{D,m_D}^{\text{old}}\right).$$

The algorithm generates fresh garblings for all the $D+1$ policies, i.e., for all $d \in [D+1]$:

$$\boldsymbol{\mu}_d^{\text{new}}, \boldsymbol{\eta}_d^{\text{new}} \xleftarrow{\$} \mathbb{Z}_p^k, \quad \begin{cases} \boldsymbol{\alpha}_d^{\text{new}} \leftarrow \boldsymbol{\mu}_d^{\text{new}}, & \boldsymbol{\beta}_d^{\text{new}} \leftarrow \mathbf{0}, \quad \text{if } y_d = f_{d,\neq 0}; \\ \boldsymbol{\alpha}_d^{\text{new}} \leftarrow \boldsymbol{\eta}_d^{\text{new}}, & \boldsymbol{\beta}_d^{\text{new}} \leftarrow \boldsymbol{\mu}_d^{\text{new}}, \quad \text{if } y_d = f_{d,=0}; \end{cases}$$
$$\text{for } t \in [k]: \quad (\mathbf{L}_{d,1}^{t,\text{new}}, \ldots, \mathbf{L}_{d,m_d}^{t,\text{new}}) \leftarrow \mathsf{Garble}(f_d, \boldsymbol{\alpha}_d^{\text{new}}[t], \boldsymbol{\beta}_d^{\text{new}}[t]; \mathbf{r}_{d,t}^{\text{new}}).$$

It then uses IPFE key-homomorphism to combine the old and new label functions as well as the pads, forming the delegated key:

$$\mathsf{isk}_{\mathrm{pad}} \xleftarrow{\$} \mathsf{isk}_{\mathrm{pad}}^{\mathrm{old}} + \sum_{t=1}^{k} \boldsymbol{\mu}^{\mathrm{new}}[t]\mathsf{bsk}_{\mathsf{const}^t}, \text{ where } \boldsymbol{\mu}^{\mathrm{new}} \leftarrow \sum_{d=1}^{D+1} \boldsymbol{\mu}_d^{\mathrm{new}},$$

$$\text{for } d \in [D+1], j \in [m_d]: \quad \mathsf{isk}_{d,j}^{\mathrm{new}} \xleftarrow{\$} \sum_{t=1}^{k} \left(\mathbf{L}_{d,j}^{t,\mathrm{new}}[\mathsf{const}]\mathsf{bsk}_{\mathsf{const}^t} + \sum_{i=1}^{n} \mathbf{L}_{d,j}^{t,\mathrm{new}}[\mathsf{coef}_i]\mathsf{bsk}_{\mathsf{coef}_i^t} \right),$$

$$\mathsf{isk}_{d,j} \xleftarrow{\$} \begin{cases} \mathsf{isk}_{d,j}^{\mathrm{old}} + \mathsf{isk}_{d,j}^{\mathrm{new}}, & \text{if } d \in [D]; \\ \mathsf{isk}_{D+1,j}^{\mathrm{new}}, & \text{if } d = D+1; \end{cases}$$

$$\mathsf{sk}_{D+1:y_1,\ldots,y_{D+1}} = \left(\mathsf{isk}_{\mathrm{pad}};\ y_1, \mathsf{isk}_{1,1}, \ldots, \mathsf{isk}_{1,m_1};\ \ldots;\ y_{D+1}, \mathsf{isk}_{D+1,1}, \ldots, \mathsf{isk}_{D+1,m_{D+1}}\right).$$

Note: *We will show (Lemma 58) that the secret key output by* KeyGen *is identically distributed to one by* $\mathsf{KeyGen}_{\text{1-shot}}$ *for the same list of policies, provided that the input secret key is well-formed.*

- $\mathsf{Enc}(\mathsf{mpk}, \mathbf{x}, g)$ takes the master public key $\mathsf{mpk} = (\mathsf{impk}, \ldots)$, the attribute $\mathbf{x} \in X_n = \mathbb{Z}_p^n$, and the message $g \in G_{\mathrm{T}}$ as input. It generates the ciphertext in the same way as the basic ABE construction. The algorithm samples $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_p^k$ and sets $\mathbf{u}$ as follows:

vector	pad	const^t	coef_i^t
$\mathbf{u}$	h	$\mathbf{s}[t]$	$\mathbf{s}[t]\mathbf{x}[i]$

It returns the following ciphertext:

$$\mathsf{ict} \xleftarrow{\$} \mathsf{IPFE.SlotEnc}(\mathsf{impk}, [\![\mathbf{u}]\!]_1), \quad \mathsf{ct} = ([\![h]\!]_{\mathrm{T}} + g, \mathbf{x}, \mathsf{ict}).$$

- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$ parses ct as $([\![z]\!]_{\mathrm{T}}, \mathbf{x}, \mathsf{ict})$ and sk as $\left(\mathsf{isk}_{\mathrm{pad}};\ y_1, \mathsf{isk}_{1,1}, \ldots, \mathsf{isk}_{1,m_1};\ \ldots;\ y_D, \mathsf{isk}_{D,1}, \ldots, \mathsf{isk}_{D,m_D}\right)$. It returns $\perp$ if $y_d(\mathbf{x}) = 0$ for some $d \in [D]$. Otherwise, it does the following:

$$[\![z']\!]_{\mathrm{T}} \leftarrow \mathsf{IPFE.Dec}(\mathsf{isk}_{\mathrm{pad}}, \mathsf{ict}),$$

$$\text{for } d \in [D], j \in [m_d]: \quad [\![\ell_{d,j}]\!]_{\mathrm{T}} \leftarrow \mathsf{IPFE.Dec}(\mathsf{isk}_{d,j}, \mathsf{ict}),$$

$$[\![\mu'_d]\!]_{\mathrm{T}} \leftarrow \begin{cases} \frac{1}{f_d(\mathbf{x})}\mathsf{Eval}(f_d, \mathbf{x}, [\![\ell_{d,1}]\!]_{\mathrm{T}}, \ldots, [\![\ell_{d,m_d}]\!]_{\mathrm{T}}), & \text{if } y_d = f_{d,\neq 0}; \\ \mathsf{Eval}(f_d, \mathbf{x}, [\![\ell_{d,1}]\!]_{\mathrm{T}}, \ldots, [\![\ell_{d,m_d}]\!]_{\mathrm{T}}), & \text{if } y_d = f_{d,=0}. \end{cases}$$

The algorithm returns $[\![z]\!]_{\mathrm{T}} + \sum_{d=1}^{D} [\![\mu'_d]\!]_{\mathrm{T}} - [\![z']\!]_{\mathrm{T}}$ as the decrypted message.

Note: *By generalizing the argument for the correctness of the basic ABE scheme (Construction 26) in the same way as for the tweaked 1-ABE scheme, we know that* Dec *correctly recovers the message in the scheme* $(\mathsf{Setup}_{\text{1-shot}}, \mathsf{KeyGen}_{\text{1-shot}}, \mathsf{Enc}, \mathsf{Dec})$. *Once we show the equivalence between* $\mathsf{KeyGen}_{\text{1-shot}}$ *and* KeyGen*, we obtain the correctness of* (Setup, KeyGen, Enc, Dec).

Lemma 58. *Suppose in Construction 57, the IPFE scheme is* perfectly key-homomorphic*, then for all $\lambda \in \mathbb{N}, n \in \mathbb{N}, D \in \mathbb{N}$ and all $y_1, \ldots, y_D, y_{D+1} \in Y_{\lambda,n}$, the following distributions are identical:*

$$\left\{\begin{array}{rll} (\mathsf{mpk}, \mathsf{msk}_{1\text{-shot}}) \xleftarrow{\$} \mathsf{Setup}_{1\text{-shot}}(1^\lambda, 1^n) & & \\ \mathsf{sk}_D \xleftarrow{\$} \mathsf{KeyGen}_{1\text{-shot}}(\mathsf{msk}_{1\text{-shot}}, y_1, \ldots, y_D) & : & (\mathsf{mpk}, \mathsf{msk}_{1\text{-shot}}, \mathsf{sk}_D, \mathsf{sk}_{D+1}) \\ \mathsf{sk}_{D+1} \xleftarrow{\$} \mathsf{KeyGen}_{1\text{-shot}}(\mathsf{msk}_{1\text{-shot}}, y_1, \ldots, y_D, y_{D+1}) & & \end{array}\right\},$$

$$\left\{\begin{array}{rll} (\mathsf{mpk}, \mathsf{msk}_{1\text{-shot}}) \xleftarrow{\$} \mathsf{Setup}_{1\text{-shot}}(1^\lambda, 1^n) & & \\ \mathsf{sk}_D \xleftarrow{\$} \mathsf{KeyGen}_{1\text{-shot}}(\mathsf{msk}_{1\text{-shot}}, y_1, \ldots, y_D) & : & (\mathsf{mpk}, \mathsf{msk}_{1\text{-shot}}, \mathsf{sk}_D, \mathsf{sk}_{D+1}) \\ \mathsf{sk}_{D+1} \xleftarrow{\$} \mathsf{KeyGen}(\mathsf{mpk}, \mathsf{sk}_D, y_{D+1}) & & \end{array}\right\}.$$

Proof (sketch). Recall that KeyGen creates new garblings, and uses key-homomorphism and the basis keys to combine the new garblings into the old keys. Since key-homomorphism is perfect and every old key takes part in *some* operation to form the new key, it suffices to show that the vectors encrypted in the keys in sk_{D+1} follows the same distribution in the two cases.

Combining the linearity of Garble, $\mathbf{B}$ being invertible, and the fact that KeyGen generates fresh garblings for all the policies and combines the newly generated one (with a change of basis determined by $\mathbf{B}$) with the old ones (which are *valid* garblings), the vectors encrypted in sk_{D+1} generated by KeyGen encode independently generated garblings for independently sampled pads, which are the same as the ones created using $\mathsf{KeyGen}_{1\text{-shot}}$. The verification is easy in principle but tedious to practice. We omit it here. □

Lemma 59. *Suppose in Construction 57, the AKGS is* piecewise secure*, the* MDDH_k *assumption holds in G_2, and the slotted IPFE is* function-hiding*, then the helper scheme* $(\mathsf{Setup}_{1\text{-shot}}, \mathsf{KeyGen}_{1\text{-shot}}, \mathsf{Enc}, \mathsf{Dec})$ *is IND-CPA secure.*

Proof (sketch). The proof is similar to that of Theorem 27 (IND-CPA of the basic ABE), except that we need one extra preparation step and have to take care of the basis keys. We highlight them and leave the rest to the reader.

Recall that in $\mathsf{Exp}^b_{\mathrm{CPA}}$, the adversary receives mpk (consisting of impk and bsk_z's), sk_q for the conjunction of $y_{q,1}, \ldots, y_{q,D_q}$, and ct encrypting g_b under attribute $\mathbf{x}$, where $y_{1,1}, \ldots, y_{1,D_1}$; $\ldots$; $y_{Q,1}, \ldots, y_{Q,D_Q}$ and $g_0, g_1, \mathbf{x}$ are chosen adaptively by $\mathcal{A}$ subject to the constraint that each queried key should not decrypt the ciphertext, i.e., for all $q \in [Q]$, there exists $d_q \in [D_q]$ such that $y_{q,d_q}(\mathbf{x}) = 0$. The basis keys bsk_z's encrypt a basis determined by a random invertible matrix $\mathbf{B}$ for the indices $\mathsf{const}^t, \mathsf{coef}^t_i$. The adversary needs to distinguish the two cases.

As before, the proof goes by moving the computation of the garblings for the challenge ciphertext to the private slot, invoking MDDH_k to argue the garblings in the private slot are indistinguishable from independently generated garblings for independently sampled pads, and lastly reducing IND-CPA security to 1-ABE security.

We describe the *first few* hybrids (the indices are aligned with those in the proof of Theorem 27):

- Hybrid H^b_0 proceeds identically to $\mathsf{Exp}^b_{\mathrm{CPA}}$. As shown in Figure 13, ict in the challenge ciphertext is generated using $\mathsf{IPFE.SlotEnc}$ (with $\perp$ values in the private slot), the basis keys bsk_z and the secret keys sk_q are generated normally.

- Hybrid H^b_1 proceeds identically to H^b_0, except that the IPFE ciphertext in the challenge ciphertext is generated using $\mathsf{IPFE.Enc}$ with 0 values in the private slot. By the slot-mode correctness, we have $\mathsf{H}^b_0 \equiv \mathsf{H}^b_1$.

hybrid	vector	pad	const^t	coef_i^t	const	coef_i	$\mathsf{sim}_1, \mathsf{sim}_\star$
$\mathsf{H}_0^b \equiv \mathsf{Exp}_{\mathrm{CPA}}^b$	$\mathsf{bsk}_{\mathsf{const}^{t'}}$	0	$\mathbf{B}[t,t']$	0	0	0	0
	$\mathsf{bsk}_{\mathsf{coef}_{i'}^{t'}}$	0	0	$\mathbf{B}[t,t']\mathbb{1}[i=i']$	0	0	0
	sk_q: $\mathbf{v}_{q,\mathrm{pad}}$	1	$\boldsymbol{\mu}_q[t]$	0	0	0	0
	sk_q: $\mathbf{v}_{q,d,j}$	0	$\mathbf{L}_{q,d,j}^t[\mathsf{const}]$	$\mathbf{L}_{q,d,j}^t[\mathsf{coef}_i]$	0	0	0
	$\boxed{\mathsf{ct}}$: $\mathbf{u}$	h	$\mathbf{s}[t]$	$\mathbf{s}[t]\mathbf{x}[i]$	$\bot$	$\bot$	$\bot$
H_1^b	$\mathsf{bsk}_{\mathsf{const}^{t'}}$	0	$\mathbf{B}[t,t']$	0	$\boxed{0}$	0	0
	$\mathsf{bsk}_{\mathsf{coef}_{i'}^{t'}}$	0	0	$\mathbf{B}[t,t']\mathbb{1}[i=i']$	0	$\boxed{0}$	0
	sk_q: $\mathbf{v}_{q,\mathrm{pad}}$	1	$\boldsymbol{\mu}_q[t]$	0	$\boxed{0}$	0	0
	sk_q: $\mathbf{v}_{q,d,j}$	0	$\mathbf{L}_{q,d,j}^t[\mathsf{const}]$	$\mathbf{L}_{q,d,j}^t[\mathsf{coef}_i]$	$\boxed{0}$	$\boxed{0}$	0
	$\boxed{\mathsf{ct}}$: $\mathbf{u}$	h	$\boxed{\mathbf{s}[t]}$	$\boxed{\mathbf{s}[t]\mathbf{x}[i]}$	$\boxed{0}$	$\boxed{0}$	0
H_2^b	$\mathsf{bsk}_{\mathsf{const}^{t'}}$	0	$\boxed{\mathbf{B}[t,t']}$	0	$\boxed{(\mathbf{B}^\intercal\mathbf{s})[t']}$	0	0
	$\mathsf{bsk}_{\mathsf{coef}_{i'}^{t'}}$	0	0	$\boxed{\mathbf{B}[t,t']\mathbb{1}[i=i']}$	0	$\boxed{(\mathbf{B}^\intercal\mathbf{s})[t']\mathbb{1}[i=i']}$	0
	sk_q: $\mathbf{v}_{q,\mathrm{pad}}$	1	$\boldsymbol{\mu}_q[t]$	0	$\boxed{\overline{\mu}_q}$	0	0
	sk_q: $\mathbf{v}_{q,d,j}$	0	$\mathbf{L}_{q,d,j}^t[\mathsf{const}]$	$\mathbf{L}_{q,d,j}^t[\mathsf{coef}_i]$	$\boxed{\overline{\mathbf{L}}_{q,d,j}[\mathsf{const}]}$	$\boxed{\overline{\mathbf{L}}_{q,d,j}[\mathsf{coef}_i]}$	0
	ct: $\mathbf{u}$	h	$\boxed{0}$	$\boxed{0}$	$\boxed{1}$	$\boxed{\mathbf{x}[i]}$	0
$\mathsf{H}_{2.71828\cdots}^b$	$\mathsf{bsk}_{\mathsf{const}^{t'}}$	0	$\boxed{\mathbf{B}[t,t']}$	0	$\boxed{(\mathbf{B}^\intercal\mathbf{s})[t']}$	0	0
	$\mathsf{bsk}_{\mathsf{coef}_{i'}^{t'}}$	0	0	$\boxed{\mathbf{B}[t,t']\mathbb{1}[i=i']}$	0	$\boxed{(\mathbf{B}^\intercal\mathbf{s})[t']\mathbb{1}[i=i']}$	0
	sk_q: $\mathbf{v}_{q,\mathrm{pad}}$	1	$\boldsymbol{\mu}_q[t]$	0	$\boxed{\overline{\mu}_q}$	0	0
	sk_q: $\mathbf{v}_{q,d,j}$	0	$\mathbf{L}_{q,d,j}^t[\mathsf{const}]$	$\mathbf{L}_{q,d,j}^t[\mathsf{coef}_i]$	$\boxed{\overline{\mathbf{L}}_{q,d,j}[\mathsf{const}]}$	$\boxed{\overline{\mathbf{L}}_{q,d,j}[\mathsf{coef}_i]}$	0
	ct: $\mathbf{u}$	h	0	0	1	$\mathbf{x}[i]$	0
H_3^b	$\mathsf{bsk}_{\mathsf{const}^{t'}}$	0	$\mathbf{B}[t,t']$	0	$\boxed{\mathbf{d}[t']}$	0	0
	$\mathsf{bsk}_{\mathsf{coef}_{i'}^{t'}}$	0	0	$\mathbf{B}[t,t']\mathbb{1}[i=i']$	0	$\boxed{\mathbf{d}[t']\mathbb{1}[i=i']}$	0
	sk_q: $\mathbf{v}_{q,\mathrm{pad}}$	1	$\boldsymbol{\mu}_q[t]$	0	$\boxed{\widehat{\mu}_q}$	0	0
	sk_q: $\mathbf{v}_{q,d,j}$	0	$\mathbf{L}_{q,d,j}^t[\mathsf{const}]$	$\mathbf{L}_{q,d,j}^t[\mathsf{coef}_i]$	$\boxed{\widehat{\mathbf{L}}_{q,d,j}[\mathsf{const}]}$	$\boxed{\widehat{\mathbf{L}}_{q,d,j}[\mathsf{coef}_i]}$	0
	ct: $\mathbf{u}$	h	0	0	1	$\mathbf{x}[i]$	0

The hybrids are illustrated with ciphertext challenge coming after the key queries (as in the more difficult case for 1-ABE). In reality, the key queries can be made before and after the ciphertext challenge, and the proof is unaffected.
In H_2^b, the linearly combined $\overline{\mathbf{L}}_{q,j} = \sum_{t\in[k]} \mathbf{s}[t]\mathbf{L}_{q,j}^t$ are valid garblings for $\overline{\mu}_q = \langle \mathbf{s}, \boldsymbol{\mu}_q \rangle$.
In $\mathsf{H}_{2.71828\cdots}^b, \mathsf{H}_3^b$, the matrix $\mathbf{B}$ is uniformly random (not conditioned on being invertible).
In H_3^b, the garblings $\widehat{\mathbf{L}}_{q,j}$ are fresh and are for freshly sampled pads $\widehat{\mu}_q$.

Figure 13: The first few hybrids the proof of IND-CPA security of $(\mathsf{Setup}_{1\text{-shot}}, \mathsf{KeyGen}_{1\text{-shot}}, \mathsf{Enc}, \mathsf{Dec})$ in Construction 57 (cf. Figure 4).

- Hybrid H_2^b proceeds identically to H_1^b, except that we hardwire the linear combination coefficients $\mathbf{s}$ into the IPFE keys. We refer the readers to the proof of Theorem 27 for the handling of the ABE secret keys and the challenge ciphertext, and focus on the basis keys. In H_1^b, the inner product between the vector in the basis key $\mathsf{bsk}_{\mathsf{const}t'}$ and the challenge ciphertext was

$$\sum_{t=1}^{k} \mathbf{B}[t,t']\mathbf{s}[t] = \sum_{t=1}^{k} \mathbf{B}^\intercal[t',t]\mathbf{s}[t] = (\mathbf{B}^\intercal\mathbf{s})[t'].$$

  So it suffices to put $(\mathbf{B}^\intercal\mathbf{s})[t']$ at index const of the vector in $\mathsf{bsk}_{\mathsf{const}t'}$. Similarly, in $\mathsf{bsk}_{\mathsf{coef}_{i'}^{t'}}$, we should embed $(\mathbf{B}^\intercal\mathbf{s})[t']$ at index $\mathsf{coef}_{i'}$ to keep the inner product unchanged, as depicted in Figure 13. Since we make sure the inner products do not change, $\mathsf{H}_1^b \approx \mathsf{H}_2^b$ by the function-hiding property.

- Hybrid $\mathsf{H}_{2.71828\cdots}^b$ proceeds identically to H_2^b, except that we do not condition $\mathbf{B}$ on being invertible. This step does not appear in the proof for the basic ABE, and is a preparation step for invoking MDDH_k. A random square matrix over $\mathbb{Z}_p$ is invertible with overwhelming probability, so $\mathsf{H}_2^b \approx_s \mathsf{H}_{2.71828\cdots}^b$.

- Hybrid H_3^b proceeds identically to $\mathsf{H}_{2.71828\cdots}^b$, except that in the private slot of the secret keys, the garblings are generated independent of the public slot garblings and for freshly sampled pads, and that in the private slot of the basis keys, $\mathbf{B}^\intercal\mathbf{s}$ is replaced by $\mathbf{d} \xleftarrow{\$} \mathbb{Z}_p^k$. Similar to the proof of Theorem 27, we rely on the MDDH_k assumption in G_2 (IPFE secret key encoding group) to argue $\mathsf{H}_{2.71828\cdots}^b \approx \mathsf{H}_3^b$. The only difference is that we invoke $\mathrm{MDDH}_{k,N+k}$ instead of $\mathrm{MDDH}_{k,N}$, where N is the total number of random elements used in the public slot garblings. The MDDH_k assumption implies

$$[\![\mathbf{A}, \mathbf{B}, \mathbf{s}^\intercal\mathbf{A}, \mathbf{s}^\intercal\mathbf{B}]\!]_2 \approx [\![\mathbf{A}, \mathbf{B}, \mathbf{c}^\intercal, \mathbf{d}^\intercal]\!]_2 \quad \text{for} \quad \mathbf{A} \xleftarrow{\$} \mathbb{Z}_p^{k\times N}, \mathbf{B} \xleftarrow{\$} \mathbb{Z}_p^{k\times k}, \mathbf{s} \xleftarrow{\$} \mathbb{Z}_p^k, \mathbf{c} \xleftarrow{\$} \mathbb{Z}_p^N, \mathbf{d} \xleftarrow{\$} \mathbb{Z}_p^k.$$

  Given $\mathbf{A}, \mathbf{B}$ and $\mathbf{c}^\intercal, \mathbf{d}^\intercal$ that are either $\mathbf{s}^\intercal\mathbf{A}, \mathbf{s}^\intercal\mathbf{B}$ or random, we use $\mathbf{B}, \mathbf{d}$ to form the public/private slots of the basis keys, and $\mathbf{A}, \mathbf{c}$ as the randomness of the garblings in the public/private slots of the queried secret keys. The latter part is again attributed to the linearity of AKGS Garble. If $\mathbf{c}^\intercal, \mathbf{d}^\intercal$ are sampled as $\mathbf{s}^\intercal\mathbf{A}, \mathbf{s}^\intercal\mathbf{B}$, the created keys follow the distribution specified in $\mathsf{H}_{2.71828\cdots}^b$; otherwise, the distribution is that specified in H_3^b. Therefore, $\mathsf{H}_{2.71828\cdots}^b \approx \mathsf{H}_3^b$ by the MDDH_k assumption in G_2.

The proof then proceeds the same as that of Theorem 27, and we omit the rest. □

Corollary 60. *Suppose in Construction 57, the AKGS is* piecewise secure*, the* MDDH_k *assumption holds in* G_2*, and the slotted IPFE is* function-hiding *and* perfectly key-homomorphic*, then the main scheme* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is IND-CPA-DLG secure.*

Proof (sketch). We prove the corollary by reduction to the IND-CPA security of the helper scheme.

Recall that in the IND-CPA-DLG experiments, the adversary $\mathcal{A}$ can delegate and reveal secret keys, and request a ciphertext encrypting one of the two messages under some attribute. The policies to delegate, the secret keys to reveal, the messages and the attribute are chosen adaptively by $\mathcal{A}$, subject to the constraint that any revealed key has some policy that disallows the challenge attribute. The task of $\mathcal{A}$ is to tell which message is being encrypted.

By Lemma 58 (and a hybrid argument over D), instead of delegating the secret keys (from msk and other delegated keys) and revealing them as requested, it is identical to remember the policies

for the would-be-delegated keys and generate the secret key using $\mathsf{KeyGen}_{\text{1-shot}}$ and $\mathsf{msk}_{\text{1-shot}}$ when it is requested for revelation for the first time. With this change made, the IND-CPA-DLG security experiments of the main scheme is exactly the IND-CPA security experiments of the helper scheme (plus some bookkeeping), and we conclude security by Lemma 59. □

C Proof of Theorem 20

Proof. Suppose $(\mathsf{Garble}, \mathsf{Eval})$ is a piecewise secure AKGS.

We first show that ℓ_1 has non-zero coefficient in Eval. Consider a function $f : \mathbb{Z}_p^n \to \mathbb{Z}_p$ of garbling size m. By Lemma 17, for any α, β, the labels for a garbling of f with secrets α, β can be simulated as $\ell_2, \dots, \ell_m \xleftarrow{\$} \mathbb{Z}_p$ and $\ell_1 \xleftarrow{\$} \mathsf{RevSamp}(f, \mathbf{x}, \gamma, \ell_2, \dots, \ell_m)$. In particular, $\ell_2 = \cdots = \ell_m = 0$ is a possible assignment of the labels for $\alpha = 0, \beta = 1$ with positive probability. By the (perfect) correctness, for this assignment, we have

$$1 = \ell_1 \mathsf{Eval}(f, \mathbf{x}, 1, 0, \dots, 0) \quad \Longrightarrow \quad \mathsf{Eval}(f, \mathbf{x}, 1, 0, \dots, 0) \neq 0.$$

Now we show how to find the change of variable for the randomness to write the labels in the special form with randomizers. Let $\mathbf{r}$ be the randomness used by Garble and $\mathbf{L}_1, \dots, \mathbf{L}_m$ the coefficients of the label functions as functions of $\alpha, \beta, \mathbf{r}$. By the linearity requirement,

$$\mathbf{L}_j[z] = u_{j,z}\alpha + v_{j,z}\beta + \langle \mathbf{w}_{j,z}, \mathbf{r} \rangle$$

for some $u_{j,z}, v_{j,z}, \mathbf{w}_{j,z}$ determined by f. Consider the following process that computes a basis $\mathcal{B}$ for the dual space of $\mathbf{r}$:

$\mathcal{B} \leftarrow \varnothing$
for $j = m, m-1, m-2, \dots, 2$:
  for $i = 1, \dots, n$:
    if $\mathbf{w}_{j,\mathsf{coef}_i}$ is linearly independent of $\mathcal{B}$:
      $\mathcal{B} \leftarrow \mathcal{B} \cup \{\mathbf{w}_{j,\mathsf{coef}_i}\}$
  $\mathcal{B} \leftarrow \mathcal{B} \cup \{\mathbf{w}_{j,\mathsf{const}}\}$
complete $\mathcal{B}$ into a basis if it is not yet a basis

We claim that $\mathcal{B}$ is always a linearly independent set. Clearly, initially $\mathcal{B} = \varnothing$ is linearly independent. By how we add $\mathbf{w}_{j,\mathsf{coef}_i}$'s, none of these steps change $\mathcal{B}$ from being linearly independent to being linearly dependent. Consider each time we add some $\mathbf{w}_{j^*,\mathsf{const}}$ and suppose (for the sake of contradiction) that $\mathbf{w}_{j^*,\mathsf{const}}$ were already in the span of $\mathcal{B}$ by the time we add it, then we would have

$$\mathbf{w}_{j^*,\mathsf{const}} = \sum_{i=1}^{n} a_i \mathbf{w}_{j^*,\mathsf{coef}_i} + \sum_{j>j^*,z} b_{j,z}\mathbf{w}_{j,z} \quad \text{for some } a_i, b_{j,z} \in \mathbb{Z}_p.$$

Now consider garbling f with secrets $\alpha = \beta = 0$ and setting $\mathbf{x}[i] = -a_i$. Given $\mathbf{L}_{j^*+1}, \dots, \mathbf{L}_m$, the label ℓ_{j^*} would be determined, contradicting the marginal randomness property:

$$\begin{aligned}\ell_{j^*} &= \sum_{i=1}^{n} \mathbf{x}[i]\langle \mathbf{w}_{j^*,\mathsf{coef}_i}, \mathbf{r} \rangle + \sum_{i=1}^{n} a_i \langle \mathbf{w}_{j^*,\mathsf{coef}_i}, \mathbf{r} \rangle + \sum_{j>j^*,z} b_{j,z}\langle \mathbf{w}_{j^*,z}, \mathbf{r} \rangle \\ &= \sum_{j>j^*,z} b_{j,z}\mathbf{L}_j[z] \qquad (\text{since } \mathbf{x}[i] + a_i = 0). \qquad \lightning\end{aligned}$$

Therefore, no step of adding $\mathbf{w}_{j,\mathsf{const}}$ makes $\mathcal{B}$ linearly dependent.

Given the basis $\mathcal{B} = \{\mathbf{b}_t, \ldots, \mathbf{b}_1\}$ (with $\mathbf{b}_t$ being the *first* vector added to $\mathcal{B}$), we change the randomness to be $\widetilde{\mathbf{r}}$ for $\widetilde{\mathbf{r}}[t] = \langle \mathbf{b}_t, \mathbf{r} \rangle$, which amounts to a change of basis of $\mathbf{r}$ (to the dual basis of $\mathcal{B}$). Note that since $\mathcal{B}$ is efficiently computable, so is the bijection between $\mathbf{r}$ and $\widetilde{\mathbf{r}}$. The garbling scheme now uses $\widetilde{\mathbf{r}}$ as the randomness, and garbles by first computing $\mathbf{r}$ then using the underlying procedure with $\mathbf{r}$.

For all $j > 1$, the label function $L_j(\mathbf{x})$ is in the special form with randomizer $\widetilde{\mathbf{r}}[t_j]$ for some t_j — $\widetilde{\mathbf{r}}[t_j] = \langle \mathbf{w}_{j,\mathsf{const}}, \mathbf{r} \rangle$ appears in the constant term as $+\widetilde{\mathbf{r}}[t_j]$, the non-constant coefficients and the coefficients of any label function $\mathbf{L}_{j'}$ with $j' > j$ only use $\widetilde{\mathbf{r}}[t]$ with $t > t_j$. □