# Multi-Client Attribute-Based and Predicate Encryption from Standard Assumptions

David Pointcheval<sup>1,2</sup> <sup>(D)</sup> and Robert Schädlich<sup>2</sup> <sup>(D)</sup>

<sup>1</sup> Cosmian, Paris, France

<sup>2</sup> DIENS, École normale superieure, CNRS, Inria, PSL University, Paris, France

**Abstract.** Multi-input Attribute-Based Encryption (ABE) is a generalization of key-policy ABE where attributes can be independently encrypted across several ciphertexts, and a joint decryption of these ciphertexts is possible if and only if the combination of attributes satisfies the policy of the decryption key. We extend this model by introducing a new primitive that we call Multi-Client ABE (MC-ABE), which provides the usual enhancements of multi-client functional encryption over multi-input functional encryption. Specifically, we separate the secret keys that are used by the different encryptors and consider the case that some of them may be corrupted by the adversary. Furthermore, we tie each ciphertext to a label and enable a joint decryption of C-ABE for various policy classes based on SXDH. Notably, we can deal with policies that are not a conjunction of local policies, which has been a limitation of previous constructions from standard assumptions.

Subsequently, we introduce the notion of Multi-Client Predicate Encryption (MC-PE) which, in contrast to MC-ABE, does not only guarantee message-hiding but also attribute-hiding. We present a new compiler that turns any constant-arity MC-ABE into an MC-PE for the same arity and policy class. Security is proven under the LWE assumption.

This is the full version of [PS25], DOI  $10.1007/978-3-031-78020-2_2$ , published by Springer in the proceedings of TCC 2024.

# Contents

1	Introduction					
	1.1	Related Work	3			
	1.2	Our Results	4			
	1.3	Relation to Witness Encryption	5			
<b>2</b>	Technical Overview					
	2.1	Construction of MC-ABE	8			
	2.2	MC-PE from MC-ABE and Lockable Obfuscation	12			
3	Preliminaries					
	3.1	Notational Conventions	15			
	3.2	Pairing Groups and Hardness Assumptions	16			
	3.3	Monotone Access Structures and Linear Secret Sharing Schemes	17			
	3.4	Function-Hiding Slotted Inner-Product Functional Encryption	18			
	3.5	Identity-Based Encryption	19			
	3.6	Lockable Obfuscation	20			
4	$\mathbf{M}\mathbf{u}$	lti-Client Attribute-Based and Predicate Encryption	<b>21</b>			
5	Construction of MC-ABE					
	5.1	MC-ABE Without Repetitions	24			
	5.2	MC-ABE With Repetitions	28			
6	Cor	nstruction of MC-PE	36			
R	efere	nces	43			

### 1 Introduction

Attribute-Based Encryption. Attribute-based encryption (ABE) [SW05, GPSW06] is a powerful generalization of classical public-key encryption that enables fine-grained access control on encrypted data. In (key-policy) ABE, a ciphertext  $CT_x$  encrypting a message  $\mu$  is generated with respect to a public attribute x while a secret decryption key  $DK_f$  is generated with respect to a policy f. The decryption key  $DK_f$  is authorized to decrypt the ciphertext  $CT_x$  if and only if the attribute x satisfies the policy f, *i.e.* f(x) = 1. Security requires indistinguishability in the presence of collusion attacks. That is, for any attribute xand any pair of messages  $(\mu^0, \mu^1)$ , ciphertexts corresponding to  $(x, \mu^0)$  and to  $(x, \mu^1)$  are indistinguishable, even for adversaries possessing a set of decryption keys  $\{DK_{f_i}\}_i$  unless one of the keys  $DK_{f_i}$  is individually authorized to decrypt. A strengthening of this notion, traditionally referred to as Predicate Encryption (PE) [BW07, SBC+07], requires ciphertexts to not only hide messages but also their associated attributes.

**Decentralized Encryption.** Until recently, ABE and PE were solely studied in the centralized setting, *i.e.* the considered policies f have arity one. The notions of Multi-Input ABE (MI-ABE) [BJK<sup>+</sup>18, AYY22] and Multi-Input PE (MI-PE) [FFMV23] overcome this limitation by considering n encryptors who each encrypt their inputs  $(x_1, \mu_1), \ldots, (x_n, \mu_n)$  independently using uncorrelated random coins. The key generator provides decryption keys for arity n functions f, and a joint decryption recovers  $(\mu_1, \ldots, \mu_n)$  if and only if  $f(x_1, \ldots, x_n) = 1$ . As in the single-input case, the security model of MI-ABE only guarantees to hide the message encoded in a ciphertext while MI-PE hides in addition the associated attribute.

In practice, the multi-input versions of ABE and PE often seem more realistic as they allow data to be encrypted in different locations or at different points in time. As an example, consider a company that stores its client data in encrypted form on a server. Each employee has their own decryption key which they can use to decrypt parts of the data depending on the employee's role. At one point, the company decides to expand and opens several new branches across the country. Clients should be able to be served from each branch. This requires that data can be independently encrypted and uploaded to the central server while still being subject to the global access control for employees. To implement these requirements, we could use an MI-ABE or MI-PE.

Let us extend the above scenario as follows. First, suppose that one of the company's branches falls victim to a hacker attack. Since MI-ABE and MI-PE use the same master secret key across all encryption slots, an attack on a single branch threatens to compromise the security of the entire system. Instead, it would be better if a restricted form of security could be preserved even if a few branches are corrupted. Second, the company's data might be time-sensitive in a sense that, say, data from different years should not be authorized for a joint decryption. To prevent unintended decryptions and the resulting data leakage, we may wish to equip ciphertexts with timestamps and allow a joint decryption if and only if all involved ciphertexts share the same timestamp.

To deal with this extended scenario, we introduce two natural generalization of MI-ABE and MI-PE which we dub *Multi-Client ABE* (MC-ABE) and *Multi-Client PE* (MC-PE). In contrast to MI-ABE and MI-PE, our new notions implement two additional features. First, they separate the secret keys of the slots and guarantee security even if some of them are known to the adversary. Second, encryption proceeds with respect to a label that can be used to realize a timestamp.

#### 1.1 Related Work

The notion of MI-ABE had been studied first by Brakerski et al. [BJK<sup>+</sup>18] as a new pathway for achieving witness encryption. However, they did not consider strong security notions

nor did they provide any constructions. In [AYY22], Agrawal et al. provided the first constructions for 2-input ABE for NC<sup>1</sup> from LWE and a nonstandard assumption on pairings. They also gave heuristic constructions for 2-input ABE for P and 3-input ABE for NC<sup>1</sup>. Additionally, they gave a compiler that lifts a constant-input ABE scheme to a PE scheme for the same arity and policy class, using a sophisticated nesting technique of lockable obfuscation which can be based on LWE. In an independent work, Francati et al. [FFMV23] built MI-PE for *conjunctions* of (bounded) polynomial-depth circuits from LWE<sup>1</sup>. Notably, they can support a polynomial number of inputs. Furthermore, when restricting to constant arity, they provide a construction that remains secure under user corruptions. On the negative side, neither of their MI-PE constructions can be proven secure under collusions. Very recently, Agrawal et al. [ARYY23] presented a constant-arity MI-ABE for NC<sup>1</sup> whose security is based on *evasive* LWE which is a strong knowledge type assumption. When additionally assuming tensor LWE, they can upgrade their scheme to support arbitrary policies in P.

MI-ABE and MI-PE can both be viewed as a special case of the more general primitive Multi-Input Functional Encryption (MIFE) [GGG<sup>+</sup>14]. The notion of MIFE has been the subject of extensive studies resulting in large body of works with various trade-offs between expressiveness, security, underlying assumptions and efficiency, e.g. [GGG<sup>+</sup>14, AJ15, AGRW17, DOT18, ACF<sup>+</sup>18, CDG<sup>+</sup>18, Tom19, ABKW19, ABG19, LT19, AGT21, AGT22]. As MIFE for  $NC^1$  is known to imply indistinguishability Obfuscation (iO) [BGI+01, GGH<sup>+</sup>13] it remains an important area of research to build MIFE schemes for simpler function classes from assumptions not known to imply iO. Some of these function classes are still powerful enough to imply MI-ABE. Specifically, Nguyen et al. [NPP22] built the first attribute-based Multi-Client Functional Encryption (MCFE) for inner products, where policies are conjunctions of Linear Secret Sharings (LSS). As they consider MCFE instead of MIFE, their construction supports corruption of users and encryption with respect to labels. They make use of pairings and proof security under the SXDH assumption. However, their security model does not allow repetitions<sup>2</sup>. In [ATY23], Agrawal et al. presented the first attribute-based MIFE for attribute-weighted sums. The supported policies are conjunctions of  $NC^1$  for a polynomial number of slots. Their construction uses pairings and is proven secure under the matrix DDH assumption. By plugging the scheme into the compiler from [AYY22], it can be lifted to MI-PE for constant arity and without corruptions.

#### 1.2 Our Results

In this work, we introduce the notions of MC-ABE and MC-PE as a generalization of their multi-input siblings. We discuss our constructions for the two primitives below. For a comparison with known results, please see Table 1.

**MC-ABE for Non-Conjunctions.** Prior to our work, all known constructions of MI-ABE and MI-PE fall into one of two categories: they are either based on standard assumptions but their supported policies are only conjunctions of local policies [NPP22, FFMV23, ATY23], or they can handle more complex policy classes such as NC<sup>1</sup> or P but their security proof relies on nonstandard assumptions [AYY22, ARYY23]. We therefore raise the following question:

<sup>&</sup>lt;sup>1</sup>A policy f is said to be a *conjunction* of a policy class  $\mathcal{F}$  if there exist policies  $f_1, \ldots, f_n \in \mathcal{F}$  such that  $f(x_1, \ldots, x_n) = f_1(x_1) \wedge \cdots \wedge f_n(x_n)$ .

<sup>&</sup>lt;sup>2</sup>Unless stated otherwise, we use the term MCFE as a generalization of MIFE, so it allows multiple uses of labels. In contrast, a weaker notion of MCFE has been considered in the literature  $[CDG^{+}18]$  where each label can be used only once, thus it does not imply MIFE. We refer to this weaker version as MCFE without repetitions

Is it possible to build MI-ABE or MC-ABE from standard assumptions for policies that are not a conjunction of local policies?

It is well known that MI-ABE for LSS can be generically upgraded to MI-ABE for  $NC^1$  via a doubling of the attribute space, and that (polynomial-arity) MI-ABE for  $NC^1$  implies Witness Encryption (WE) for languages that can be verified in  $NC^1$  [BJK<sup>+</sup>18]. Considering the fact that the construction of WE with  $NC^1$  verification from standard assumptions is still an open problem, it is not surprising that MI-ABE for policies that are not a conjunction has also remained elusive so far, even for relatively simple policy classes such as LSS. As previous works [FFMV23, ATY23], we are not able to build MI-ABE or MC-ABE for a policy class that is powerful enough to trigger the entire chain of implications up to WE. Nonetheless, we identify various special cases that circumvent the known implications to WE, thereby giving an affirmative answer to the above question. Specifically, we present constructions of MC-ABE for the following situations:

- 1. Small Parameters. If the arity n and the attribute space  $\{0,1\}^k$  are small (more precisely, they satisfy  $kn = O(\log \lambda)$ ), then we can build an MC-ABE for all NC<sup>1</sup> policies.
- 2. Simple Policies. If  $k = n = \text{poly}(\lambda)$ , then we can build MC-ABEs for NC<sup>0</sup> policies and threshold policies with constant threshold, *i.e.* policies that accept any combination of at least  $\tau$  out of the total of kn attributes, where  $\tau = O(1)$  or  $\tau = kn - O(1)$ . Note that NC<sup>0</sup> policies can only depend on a constant number of inputs whereas threshold policies depend on *all* inputs; so they cannot be implemented in NC<sup>0</sup>.
- 3. Weaker Security. In the weaker MCFE model without repetitions, we can choose  $k = n = \text{poly}(\lambda)$  and build MC-ABE for the policy class NC<sup>1</sup> and no user corruptions, or for the policy class LSS with user corruptions.

We discuss the relation between our constructions and the (non-)implications to WE in more detail below.

**From MC-ABE to MC-PE.** In [AYY22], the authors present two generic compilers that lift an MI-ABE scheme to MI-PE for the same policy class. The first one can deal with any constant arity but works only in a weak security model where the adversary must not obtain valid decryption keys for *any* ciphertext, even if the ciphertext corresponds to a "non-challenge" encryption query  $(x, \mu^0, \mu^1)$  where  $\mu^0 = \mu^1$ . Note that the ability to decrypt such non-challenge ciphertexts does not render the security game trivial and admitting this kind of queries yields a stronger security model. Indeed, their second compiler allows the decryption of non-challenge ciphertexts, but works only for arity 2.

In this work, we present a new generic compiler that works for a constant number of inputs and achieves the stronger security model. Moreover, it can deal with labels and corruption of users, thus turning MC-ABE into MC-PE. Similar to [AYY22], our construction relies on lockable obfuscation whose security can be based on LWE.

### **1.3** Relation to Witness Encryption

A witness encryption (WE) scheme for an NP relation  $\mathcal{R}$  defined over a language  $\mathcal{L}$  allows a sender to efficiently encrypt a message  $\mu$  with respect to a problem instance x. A receiver holding a witness w can recover the message  $\mu$  if  $(x, w) \in \mathcal{R}$ . Security requires that ciphertexts for messages  $\mu^0$  and  $\mu^1$  are computationally indistinguishable if  $x \notin \mathcal{L}$ . The authors of [BJK<sup>+</sup>18] define a relaxation of classical WE that they call *non-trivially* eXponentially efficient WE (XWE), where the runtime of the encryption algorithm for witnesses of length n is  $\tilde{O}(2^{\gamma n})$  for some constant  $\gamma < 1$  called the *compression factor*.

Work	Arity	Attribute <sup>1</sup>	Collusion	Corruption	Labels	Policy Class	Assumptions
[AYY22]	2	private	1	×	×	$NC^1$	KOALA <sup>2</sup> , LWE
		private	1		×	$NC^1$	Evasive LWE <sup>3</sup>
[ARYY23]	const			×		Р	Evasive and Tensor LWE
[NPP22]	poly	public	1	1	$OT^4$	Conjunctions of LSS	SXDH
FFMV23	poly	- privete	private X Conj	×	×	Conjunctions	IWF
	const			of P			
[ATY23]	poly	public	1	1	×	$\begin{array}{c} {\rm Conjunctions} \\ {\rm of} \; NC^1 \end{array}$	Matrix DDH
$\begin{vmatrix} [ATY23] \\ + [AYY22] \end{vmatrix}$	const	private	1	×	×	$\begin{array}{c} {\rm Conjunctions} \\ {\rm of} \ NC^1 \end{array}$	Matrix DDH, LWE
G., 51	poly	public		1	ОТ	LSS	SXDH
Sec. 5.1				×		$NC^1$	
	$\log^5$					$NC^1$	
Sec. 5.2	poly public		1		$NC^0$ or const threshold	SXDH	
$\begin{vmatrix} \text{Sec. } 5.2 \\ + \text{Sec. } 6 \end{vmatrix}$	const <sup>6</sup>	private	1	<i>✓</i>		NC <sup>1</sup>	SXDH, LWE
[ATY23] + Sec. 6	const	private	1	1	×	$\begin{array}{c} {\rm Conjunctions} \\ {\rm of} \ NC^1 \end{array}$	Matrix DDH, LWE

Table 1: Comparison with existing works in MI-ABE and MI-PE

<sup>1</sup> Public attributes correspond to ABE and private attributes to PE.

 $^2$  KOALA is a nonstandard knowledge type assumption on pairings.

<sup>3</sup> Evasive LWE is a nonstandard knowledge type assumption on lattices.

 $^4$  OT refers to *one-time* labels, *i.e.* the weaker MCFE model without repetitions.

<sup>5</sup> More precisely, the scheme's arity n and attribute space  $\{0, 1\}^k$  are subject to the condition  $kn = O(\log \lambda)$ .

<sup>6</sup> The limitation from 5 is still in place and translates into  $k = O(\log \lambda)$ . Therefore, it does not include the next row which allows only conjunctions but  $k = poly(\lambda)$ .

In [BJK<sup>+</sup>18], the authors show that *n*-input ABE for a policy class  $\mathcal{F}$  implies WE for relations with length *n* witnesses whose verification algorithm is in  $\mathcal{F}$ . If  $n = \text{poly}(\lambda)$  and  $\mathcal{F} = \mathsf{P}$ , we obtain WE for all NP relations. But even for smaller arity or simpler policy classes there are nontrivial implications. Since there are NP relations that can be verified in NC<sup>1</sup> (*e.g.* 3-SAT), MI-ABE for NC<sup>1</sup> policies already implies WE for certain NP relations. Furthermore, it is shown that *n*-input ABE for  $n < \text{poly}(\lambda)$  implies XWE with a compression factor of  $\gamma = 1/(n+1)$ . Plugging the two-input ABE from [AYY22] or the O(1)-input ABE from [ARYY23] into the conversion to XWE, one obtains compression

factors of  $\gamma = 1/3$  and  $\gamma = 1/O(1)$ , respectively; although the latter result may be less interesting as their hardness assumption, evasive LWE, is already known to imply WE [Tsa22, VWW22]. When relying on standard assumptions, the best known compression factor is still  $\gamma = 1/2$  which corresponds to a classical single-input ABE scheme, and any improvement would be highly interesting. Unfortunately, all our constructions fail to improve the compression factor due to the following reasons. (1) We either need  $n = \text{poly}(\lambda)$ and k = 1 in which case we immediately get (polynomially efficient) WE, or  $n < \text{poly}(\lambda)$ and  $k = \text{poly}(\lambda)$  in which case we obtain XWE with compression factor 1/(n + 1). If both  $k, n < \text{poly}(\lambda)$ , it is unclear how the compression factor could be improved. (2) NC<sup>0</sup> or constant-threshold policies are not powerful enough to verify an NP language. (3) The weaker MC-ABE model without repetitions does not imply MI-ABE, thus fails to imply (X)WE.

The work [FFMV23] presents an interesting alternative pathway towards WE. If the MI-ABE is secure under corruptions, then a two-input scheme for conjunctions of some policy class  $\mathcal{F}$  implies WE for any relation whose verification algorithm lies in  $\mathcal{F}$ . Importantly, for the conversion of [FFMV23] to work, the first slot must have a wildcard while the second slot must not. This property is achieved by all our constructions. However, even in this case our construction for NC<sup>1</sup> fails to imply WE because for n = 2, the constraint  $kn = O(\log \lambda)$  translates into  $k = O(\log \lambda)$  which is not enough as witnesses must be of polynomial length.

# 2 Technical Overview

We first introduce our new primitives MC-ABE and MC-PE. Our syntax closely follows [ARYY23]. Specifically, the 0-th client (the "encryptor") runs an algorithm Enc which takes as input a label  $lab_0$ , an attribute  $x_0$  and a message  $\mu$  to create a ciphertext  $CT_{lab_0,x_0}$ . The other clients  $1, \ldots, n-1$  (the "attribute key generators") run an algorithm AKeyGen which takes only a label  $\mathsf{lab}_i$  and an attribute  $x_i$  to generate a decryption key  $\mathsf{DK}_{\mathsf{lab}_i,x_i}$ . Policy decryption keys  $\mathsf{DK}_f$  for a policy f are generated by a central authority which runs an algorithm PKeyGen.  $CT_{lab_0,x_0}$  can be decrypted using  $\{DK_{lab_i,x_i}\}_i$  and  $DK_f$  if  $\mathsf{lab}_0 = \cdots = \mathsf{lab}_{n-1}$  and  $f(x_0, \ldots, x_{n-1}) = 1$ . For MC-ABE security, we require the usual ciphertext indistinguishability against collusion attacks under corruptions. MC-PE security additionally considers left-or-right queries for attributes in both slot 0 ciphertexts and slot i attribute decryption keys for all  $i \in [n-1]$ . This leads to a subtle yet important difference in the security models. In MC-ABE, the encryption oracle of client 0 is the only left-or-right ("challenge") oracle. In this case, public-key security is stronger than secret-key security which is why we provide client 0 with a master public key MPK and all other clients  $i \in [n-1]$  with a secret key  $\mathsf{SK}_i$ . In MC-PE on the other hand, the oracles of all clients take left-or-right queries. Now considering a public encryption algorithm would actually make the primitive weaker due to inevitable leakage. This is a well-known phenomenon in the context of MIFE and MCFE in general. For this reason, we consider MC-PE in the secret-key setting where the encryptor takes a secret key  $\mathsf{SK}_0$  instead of a public key MPK. We summarize the syntax as follows:

$$\begin{array}{ll} \text{Client } 0: & \mathsf{Enc}(\mathsf{MPK}/\mathsf{SK}_0,\mathsf{lab},x_0,\mu) \to \mathsf{CT}_{x_0}\\ \text{Client } i \in [n-1]: & \mathsf{AKeyGen}(\mathsf{SK}_i,\mathsf{lab},x_i) \to \mathsf{DK}_{x_i}\\ \text{Authority:} & \mathsf{PKeyGen}(\mathsf{MSK},f) \to \mathsf{DK}_f \end{array}$$

For completeness, we mention that MC-ABE and MC-PE can also be considered in an n-message setting where not only the 0-th but all slots encrypt a message. In [AYY22], it was shown that a single-message MI-ABE scheme can be generically lifted to an n-message scheme. The conversion is extremely simple and basically runs n single-message schemes in parallel with rotated slots. The same technique generalizes to MC-ABE and MC-PE.

#### 2.1 Construction of MC-ABE

**Ingredients to our Constructions.** Let  $\mathbb{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, g_1, g_2, g_t, e, p)$  be a pairing group. For  $i \in \{1, 2, t\}$  and  $a \in \mathbb{Z}_p$ , we write  $[\![a]\!]_i = g_i^a$  and use additive notations for the group operations.

An inner-product functional encryption (IPFE) scheme based on  $\mathbb{G}$  enables the generation of ciphertexts  $i\mathsf{CT}(\llbracket \mathbf{x} \rrbracket_1)$  associated with vectors  $\mathbf{x} \in \mathbb{Z}_p^N$  encoded in  $\mathbb{G}_1$  and decryption keys  $i\mathsf{DK}(\llbracket \mathbf{y} \rrbracket_2)$  for vectors  $\mathbf{y} \in \mathbb{Z}_p^N$  encoded in  $\mathbb{G}_2$  such that the decryption of  $i\mathsf{CT}(\llbracket \mathbf{x} \rrbracket_1)$ with  $i\mathsf{DK}(\llbracket \mathbf{y} \rrbracket_2)$  reveals only the inner product  $\llbracket \mathbf{x}^\top \mathbf{y} \rrbracket_t$  of  $\mathbf{x}$  and  $\mathbf{y}$  encoded in  $\mathbb{G}_t$  and hides all other information about  $\mathbf{x}$  whereas  $\llbracket \mathbf{y} \rrbracket_2$  is usually public. When we use several IPFE schemes in parallel, we add an index to indicate the respective instance, *e.g.* for the *i*-th IPFE instance, we write  $i\mathsf{CT}_i(\llbracket \mathbf{x} \rrbracket_1)$  and  $i\mathsf{DK}_i(\llbracket \mathbf{y} \rrbracket_2)$ .

In the same vein, *identity-based encryption* (IBE) allows creating ciphertexts  $idCT(i, \mu)$  associated with an identity *i* for a message  $\mu$ , and decryption keys idDK(i') associated with an identity *i'*. Decryption is possible if i = i'.

Given a vector  $\mathbf{x} \in \{0,1\}^k$ , we write  $\Pi_{\mathbf{x}} = \{j \in [k] : \mathbf{x}[j] = 1\}$ . A linear secret sharing (LSS) scheme allows to decompose a secret scalar  $s \in \mathbb{Z}_p$  into a vector of shares  $\mathsf{Share}(s, f) \to \mathbf{s} \in \mathbb{Z}_p^k$  with respect to some policy  $f : \{0,1\}^k \to \{0,1\}$  such that s can be reconstructed from a subset of the shares  $\{\mathbf{s}[j]\}_{j\in\Pi_{\mathbf{x}}}$  for  $\mathbf{x} \in \{0,1\}^k$  if and only if  $f(\mathbf{x}) = 1$ . For this reconstruction, there exists an efficient algorithm  $\mathsf{FindCoef}(\Pi_{\mathbf{x}}, f)$  that outputs coefficients  $\omega_1, \ldots, \omega_k$  such that  $\omega_j = 0$  for all  $j \notin \Pi_{\mathbf{x}}$  and  $\sum_{i \in [k]} \omega_i \mathbf{s}[j] = s$ .

**Key-Policy ABE for LSS Policies.** Our starting point is a technique that combines IPFE with secret sharing schemes. The same approach has recently been used to build ciphertext-policy ABEs with interesting new features [AWY20, AG21, LLL22, AG23]. Very roughly, these works view a secret sharing as a weak form of one-time, non-collusion resistant ABE, which is then lifted to full ABE using IPFE. To encrypt a message  $\mu$  from a polynomial-size space<sup>3</sup> under a policy f, they generate secret shares  $\mathsf{Share}(\mu, f) \to \mathbf{s} \in \mathbb{Z}_p^k$  and encode them in the ciphertexts of k independent IPFE instances. To generate a key for an attribute vector  $\mathbf{x} \in \{0, 1\}^k$ , one picks a uniformly random scalar r and generates IPFE secret keys of r for those IPFE instances that correspond to indices in  $\Pi_{\mathbf{x}}$ :

$$\begin{array}{l} \mathsf{CP}\mathsf{-}\mathsf{ABE}\mathsf{.}\mathsf{CT}_{f} : \quad \{\mathsf{i}\mathsf{CT}_{j}(\llbracket \mathbf{s}[j] \rrbracket_{2})\}_{j \in [k]} \\ \mathsf{CP}\mathsf{-}\mathsf{ABE}\mathsf{.}\mathsf{DK}_{\mathbf{x}} : \quad \llbracket r \rrbracket_{\mathbf{t}}, \{\mathsf{i}\mathsf{DK}_{j}(\llbracket r \rrbracket_{1})\}_{j \in \Pi_{\mathbf{x}}} \end{array} \} \quad \llbracket r \rrbracket_{\mathbf{t}}, \{\llbracket r \cdot \mathbf{s}[j] \rrbracket_{\mathbf{t}}\}_{j \in \Pi_{\mathbf{x}}} \end{array}$$

IPFE decryption yields target group encodings of  $r\mathbf{s}[j]$  for all  $j \in \Pi_{\mathbf{x}}$ . If  $f(\mathbf{x}) = 1$ , one can run FindCoef $(\Pi_{\mathbf{x}}, f) \to {\{\omega_j\}_j}$  and recover the product  $r \cdot \mu$  encoded in  $\mathbb{G}_t$ :

$$\sum_{j\in\Pi_{\mathbf{x}}}\omega_{j}\llbracket r\cdot\mathbf{s}[j]\rrbracket_{\mathbf{t}} = \llbracket r\cdot\sum_{j\in\Pi_{\mathbf{x}}}\omega_{j}\cdot\mathbf{s}[j]\rrbracket_{\mathbf{t}} = \llbracket r\cdot\mu\rrbracket_{\mathbf{t}}$$

Then one can find  $\mu$  by solving the discrete logarithm of  $[\![r \cdot \mu]\!]_t$  in basis  $[\![r]\!]_t$ . Under an appropriate hardness assumption, the presence of r prevents adversaries from meaningfully "combining" information obtained from decryptions with different ABE decryption keys.

To turn this into a key-policy scheme, the obvious idea is to flip ciphertexts and decryption keys. However, there is one subtlety: when generating a secret sharing for a policy f during the key generation, the message  $\mu$  is not known. So one cannot generate secret shares of  $\mu$ . Therefore, we generate the secret sharing for a random scalar s which is used to mask  $\mu$ . Then one uses another IPFE instance to enable decryption:

$$\begin{array}{ll} \mathsf{KP}\mathsf{-}\mathsf{ABE}.\mathsf{CT}_{\mathbf{x}} : & \mathsf{iCT}_0(\llbracket r, \mu \rrbracket_1), \{\mathsf{iCT}_j(\llbracket r \rrbracket_1)\}_{j \in \Pi_{\mathbf{x}}} \\ \mathsf{KP}\mathsf{-}\mathsf{ABE}.\mathsf{DK}_f : & \mathsf{iDK}_0(\llbracket s, 1 \rrbracket_2), \{\mathsf{iDK}_j(\llbracket \mathbf{s}[j] \rrbracket_2)\}_{j \in [k]} \end{array} \right\} \quad \begin{array}{l} \llbracket r \cdot s + \mu \rrbracket_t, \\ \{\llbracket r \cdot \mathbf{s}[j] \rrbracket_t\}_{j \in \Pi} \end{array}$$

<sup>&</sup>lt;sup>3</sup>The restriction to a polynomial-size message space is only for notational convenience throughout the technical overview. For superpolynomial size, one can simply view the construction as a KEM with messages in  $\mathbb{G}_t$  that can be used as a one-time pad.

Similar to above, if  $f(\mathbf{x}) = 1$ , one can run FindCoef $(\Pi_{\mathbf{x}}, f) \to {\{\omega_j\}_j}$  and recover the message  $\mu$  encoded in  $\mathbb{G}_t$ :

$$\llbracket r \cdot s + \mu \rrbracket_{\mathbf{t}} - \sum_{j \in \Pi_{\mathbf{x}}} \omega_j \llbracket r \cdot \mathbf{s}[j] \rrbracket_{\mathbf{t}} = \llbracket r \cdot s + \mu \rrbracket_{\mathbf{t}} - \llbracket r \cdot \sum_{j \in \Pi_{\mathbf{x}}} \omega_j \mathbf{s}[j] \rrbracket_{\mathbf{t}} = \llbracket \mu \rrbracket_{\mathbf{t}}$$

MC-ABE for LSS Without Repetitions. We next discuss how the generation of the ciphertext  $CT_x$  can be distributed so as to turn the above key-policy ABE into an MC-ABE. A natural approach is to follow [CDG<sup>+</sup>18, NPP22] who construct (Decentralized) MCFE for inner products. Even though not explicitly stated as such, they essentially use an independent IPFE instance for each client, and the common randomness r in the ciphertexts facing a secret sharing  $(\mathbf{s}[j])_j$  in the decryption keys is provided by a random oracle. Translating this idea into our context, each client  $i \in [0; n-1]$  holds the master secret keys of k independent IPFE instances, where k is the dimension of the attribute vectors<sup>4</sup>. The 0-th client additionally holds  $iMSK_0$  as it takes the message input. To obtain the common random scalar  $[\![r]\!]_1$  encoded in  $\mathbb{G}_1$ , we use a hash function H:  $\{0,1\}^* \to \mathbb{G}_1$ . To generate a decryption key for an attribute vector  $\mathbf{x}_i \in \{0,1\}^k$ with respect to a label lab, the corresponding client  $i \in [n-1]$  computes  $[r]_1 \leftarrow H(lab)$  and issues  $\{\mathsf{iCT}_{i,j}(\llbracket r \rrbracket_1)\}_{j\in\Pi_{\mathbf{x}_i}}$ . Similarly, to encrypt a message  $\mu$  with respect to  $\mathbf{x}_0 \in \{0,1\}^k$ , the 0-th client computes  $\{\mathsf{iCT}_{0,j}(\llbracket r \rrbracket_1)\}_{j\in\Pi_{\mathbf{x}_0}}$  and additionally provides  $\mathsf{iCT}_0(\llbracket r, \mu \rrbracket_1)$ . Decryption keys  $\mathsf{DK}_f$  for policies f are still generated by a central authority, so the policy key generation algorithm does not need to be modified. This leads us to the following MC-ABE for LSS:

$$\begin{split} & \mathsf{MC}\text{-}\mathsf{ABE}.\mathsf{CT}_{\mathbf{x}_{0}}: \ \mathsf{i}\mathsf{CT}_{0}([\![r,\mu]\!]_{1}), \big\{\mathsf{i}\mathsf{CT}_{0,j}([\![r]\!]_{1})\big\}_{j\in\Pi_{\mathbf{x}_{0}}} \\ & \mathsf{MC}\text{-}\mathsf{ABE}.\mathsf{DK}_{\mathbf{x}_{i}}: \ \big\{\mathsf{i}\mathsf{CT}_{i,j}([\![r]\!]_{1})\big\}_{j\in\Pi_{\mathbf{x}_{i}}} \\ & \mathsf{MC}\text{-}\mathsf{ABE}.\mathsf{DK}_{f}: \ \mathsf{i}\mathsf{DK}_{0}([\![s,1]\!]_{2}), \big\{\mathsf{i}\mathsf{DK}_{i,j}([\![\mathbf{s}[i,j]]\!]_{2})\big\}_{i\in[0;n-1]}^{j\in[k]} \\ \end{split} \right\} \begin{cases} [\![rs+\mu]\!]_{\mathsf{t}}, \\ \{[\![rs[i,j]]\!]_{\mathsf{t}}\}_{i\in[0;n-1]}^{j\in\Pi_{\mathbf{x}_{i}}} \\ \{[\![rs[i,j]]\!]_{\mathsf{t}}\}_{i\in[0;n-1]}^{j\in\Pi_{\mathbf{x}_{i}}} \end{cases} \end{cases}$$

where  $[\![r]\!]_1 \leftarrow \mathsf{H}(\mathsf{lab})$  and  $\mathbf{s}[i, j]$  denotes the entry of the share vector corresponding to the *j*-th coordinate of  $\mathbf{x}_i$  for  $i \in [0; n-1]$  and  $j \in [k]$ . The security notion that we can achieve for this scheme suffers from the same limitations as [CDG<sup>+</sup>18, NPP22]; most importantly, we cannot prove security under repetitions. Moreover, not being able to prove security under repetitions implies that the encryption algorithm must take a secret key, as otherwise the adversary could create multiple ciphertexts under the same label by herself. The reason for these restrictions is the fact that our only source of randomness is the random oracle whose only input is the label. Hence, to achieve security in a stronger model, our first step is to remove the random oracle from the construction.

Removing the Random Oracle and Enabling Public Encryption. In our first attempt above, the random oracle provides common randomness across independently generated ciphertexts and keys. Clearly, this is not possible anymore without a random oracle. Therefore, it seems inevitable to have one client (say, the 0-th) generate all the IPFE ciphertexts  $\{iCT_{i,j}([\![r]\!]_1)\}_{i,j}$ . However, when generating  $CT_{\mathbf{x}_0}$ , the vectors  $\mathbf{x}_1, \ldots, \mathbf{x}_{n-1}$  are unknown, so it is unclear which  $iCT_{i,j}([\![r]\!]_1)$  for i > 0 should be included in  $CT_{\mathbf{x}_0}$ .

As a solution, we let client 0 generate all ciphertexts  $\{i\mathsf{CT}_{i,j}(\llbracket r \rrbracket_1)\}_{i\in[n-1]}^{j\in[k]}$ , but instead of providing them "in the clear", we hide them with an additional layer of identity-based encryption. Specifically, the 0-th client encrypts each  $\mathsf{iCT}_{i,j}(\llbracket r \rrbracket_1)$  with respect to the identity (lab, j) using the public key of client i. Correspondingly, client  $i \in [n-1]$  provides the identity-based decryption keys  $\mathsf{idDK}_i(\mathsf{lab}, j)$  for each  $j \in \Pi_{\mathbf{x}_i}$  needed to recover the

<sup>&</sup>lt;sup>4</sup>Ciphertexts and decryption keys corresponding to the *j*-th IPFE scheme of client *i* are denoted by  $iCT_{i,j}$  and  $iDK_{i,j}$ .

IPFE ciphertexts generated by client 0. This idea yields the following MC-ABE for LSS in the standard model:

$$\begin{array}{ll}
\mathsf{CT}_{\mathbf{x}_{0}}: & \begin{bmatrix} \mathsf{i}\mathsf{CT}_{0}(\llbracket r, \mu \rrbracket_{1}), \{\mathsf{i}\mathsf{CT}_{0,j}(\llbracket r \rrbracket_{1})\}_{j \in \Pi_{\mathbf{x}_{0}}} \\ \{\mathsf{i}\mathsf{d}\mathsf{CT}_{i}((\mathsf{lab}, j), \mathsf{i}\mathsf{CT}_{i,j}(\llbracket r \rrbracket_{1}))\}_{i \in [n-1]}^{j \in [k]} \end{bmatrix} \\
\mathsf{DK}_{\mathbf{x}_{i}}: & \{\mathsf{i}\mathsf{d}\mathsf{DK}_{i}(\mathsf{lab}, j)\}_{j \in \Pi_{\mathbf{x}_{i}}} \\
\mathsf{DK}_{f}: & \mathsf{i}\mathsf{DK}_{0}(\llbracket s, 1 \rrbracket_{2}), \{\mathsf{i}\mathsf{DK}_{i,j}(\llbracket \mathbf{s}[i, j] \rrbracket_{2})\}_{i \in [0; n-1]}^{j \in [k]} \end{bmatrix} \\
\end{array} \right\} \begin{bmatrix} \llbracket r \cdot \mathbf{s} + \mu \rrbracket_{t}, \\ \{\llbracket r \cdot \mathbf{s}[i, j] \rrbracket_{t}\}_{i \in [0; n]}^{j \in \Pi_{\mathbf{x}_{i}}} \\ \{\llbracket r \cdot \mathbf{s}[i, j] \rrbracket_{t}\}_{i \in [0; n]}^{j \in \Pi_{\mathbf{x}_{i}}} \end{bmatrix} \\$$

Here,  $r \stackrel{\$}{\leftarrow} \mathbb{Z}_p$  is a fresh random scalar for each ciphertext. Due to this fact, the scheme remains secure even under several encryption queries for the same label and, in particular, enables a public encryption algorithm. Indeed, if each encryption samples a fresh  $r \stackrel{\$}{\leftarrow} \mathbb{Z}_p$ , then each message  $\llbracket \mu \rrbracket_t$  is hidden by a fresh looking mask  $\llbracket r \cdot s \rrbracket_t$ . So the ability to create ciphertexts by herself does not help the adversary to recover information from a challenge ciphertext anymore.

On the negative side, the scheme in (1) is still not secure under repetitions for slots  $i \in [n-1]$ . For example, consider an adversary that submits queries for decryption keys  $\mathsf{DK}_{\mathbf{x}_i}$  and  $\mathsf{DK}_{\mathbf{x}'_i}$  for two attribute vectors  $\mathbf{x}_i, \mathbf{x}'_i \in \{0,1\}^k$ . Then  $\mathsf{DK}_{\mathbf{y}_i} \coloneqq \mathsf{DK}_{\mathbf{x}_i} \cup \mathsf{DK}_{\mathbf{x}'_i}$  is a decryption key for the vector  $\mathbf{y}_i \in \{0,1\}^k$  having a 1 in all coordinates  $j \in [k]$  where  $1 \in \{\mathbf{x}[j], \mathbf{x}'[j]\}$ . Thus,  $\mathsf{DK}_{\mathbf{y}_i}$  may be used to decrypt ciphertexts that cannot be decrypted by neither  $\mathsf{DK}_{\mathbf{x}_i}$  nor  $\mathsf{DK}_{\mathbf{x}'_i}$ .

MC-ABE for LSS With Repetitions. To achieve security under repetitions for slots  $i \in [n-1]$ , we must make sure that multiple decryption keys for the same label-slot pair (lab, i) cannot be combined in a meaningful way as it is possible for the scheme in (1). In other words, all components of a decryption key  $\mathsf{DK}_{\mathbf{x}_i}$  should "depend on" the entire vector  $\mathbf{x}_i$  instead of only a single coordinate  $\mathbf{x}_i[j]$ . To this end, we now let  $\mathsf{DK}_{\mathbf{x}_i} = \mathsf{idDK}_i(\mathsf{lab}, \mathbf{x}_i)$  as opposed to  $\{\mathsf{idDK}_i(\mathsf{lab}, j)\}_{j \in \Pi_{\mathbf{x}_i}}$ . Then security of the MC-ABE under repetitions directly corresponds to the collusion resistance of the employed IBE.

On the other hand, correctness is no longer straightforward. This is because a successful decryption using the new keys requires the 0-th client to provide encryptions of the IPFE ciphertexts  $\{iCT_{i,j}([\![r]\!]_1)\}_{i,j}$  with respect to identities that depend on attribute vectors  $\mathbf{x}_1, \ldots, \mathbf{x}_{n-1}$ . These vectors are not given as input and, thus, are unknown at encryption time. Moreover, decryption with a key  $\mathsf{DK}_f$  is supposed to work with any combination of  $\mathbf{x}_1, \ldots, \mathbf{x}_{n-1}$  satisfying  $f(\mathbf{x}_0, \ldots, \mathbf{x}_{n-1}) = 1$ . Therefore, the problem is not only that these attribute vectors are unknown, but in general there can be many possible choices that should allow decrypting. In particular, when f is the constant function that always outputs 1, then decryption must succeed for any choice of  $\mathbf{x}_1, \ldots, \mathbf{x}_{n-1}$ . This observation ultimately forces the encryptor to provide encryptions of the IPFE ciphertexts  $\{iCT_{i,j}([\![r]\!]_1)\}_j$  with respect to every identity (lab,  $\mathbf{x}_i$ ) such that  $\mathbf{x}_i \in \{0, 1\}^k$ , for  $i \in [n-1]$ . More precisely, a ciphertext  $\mathsf{CT}_{\mathbf{x}_0}$  for a message  $\mu$  consists of the following components:

$$\mathsf{CT}_{\mathbf{x}_{0}}: \left\{ \begin{cases} \mathsf{iCT}_{0}(\llbracket r_{\mathbf{x}}, \mu \rrbracket_{1}), \{\mathsf{iCT}_{0,j}(\llbracket r_{\mathbf{x}} \rrbracket_{1})\}_{j \in \Pi_{\mathbf{x}_{0}}}, \\ \{\mathsf{idCT}_{i}((\mathsf{lab}, \mathbf{x}_{i}), \mathsf{iCT}_{i,j}(\llbracket r_{\mathbf{x}} \rrbracket_{1}))\}_{i \in [n-1]}^{j \in \Pi_{\mathbf{x}_{i}}} \right\}_{\mathbf{x}=(\mathbf{x}_{1}, \dots, \mathbf{x}_{n-1}) \in \{0, 1\}^{(n-1)k}}$$
(2)

where  $r_{\mathbf{x}} \stackrel{\text{\scale}}{=} \mathbb{Z}_p$  for each  $\mathbf{x} \in \{0, 1\}^{(n-1)k}$ . It is clear that these ciphertexts have exponential size if k or n are chosen too large. However, it remains polynomial if one chooses e.g.  $k \cdot n = O(\log \lambda)$  which gives  $|\{0, 1\}^{(n-1)k}| = \operatorname{poly}(\lambda)$ .

**Upgrading the Policy Class to** NC<sup>1</sup>. Let f be a policy specified by an NC<sup>1</sup> circuit over the variables  $(\mathbf{x}_0, \ldots, \mathbf{x}_{n-1}) \in \{0, 1\}^{[0;n-1] \times [k]}$ . We can view f as a Boolean formula consisting of (fan-in 1)  $\neg$  gates and (fan-in 2)  $\land$  and  $\lor$  gates. Using De Morgan laws, we can push the  $\neg$  gates to the leaves such that all internal nodes consist only of  $\land$  and  $\lor$ gates, while leaves are labeled by either attributes or their negations. In this way, we obtain a monotone formula  $\overline{f}: \{0, 1\}^{[0;n-1] \times [k] \times \{0,1\}} \to \{0,1\}$  that is "equivalent" to f in the following sense. For each  $\mathbf{x} \in \{0,1\}^{[k]}$ , we define the *extended* vector  $\overline{\mathbf{x}} \in \{0,1\}^{[k] \times \{0,1\}}$ component-wise via  $\overline{\mathbf{x}}[(j,1)] = \mathbf{x}[j]$  and  $\overline{\mathbf{x}}[(j,0)] = 1 - \mathbf{x}[j]$  for each  $j \in [k]^5$ . Then we have  $f(\mathbf{x}_0, \ldots, \mathbf{x}_{n-1}) = \overline{f}(\overline{\mathbf{x}}_0, \ldots, \overline{\mathbf{x}}_{n-1})$  for each  $(\mathbf{x}_0, \ldots, \mathbf{x}_{n-1})$ . Lewko and Waters [LW11] presented an LSS for all monotone access structures which implies that  $\overline{f}$  can be captured by an LSS.

Given an MC-ABE  $\overline{\mathsf{aFE}}$  for LSS policies  $\overline{f}: \{0,1\}^{[0;n-1]\times[k]\times\{0,1\}} \to \{0,1\}$ , we can build an MC-ABE  $\mathsf{aFE}$  for  $\mathsf{NC}^1$  policies  $f: \{0,1\}^{[0;n-1]\times[k]} \to \{0,1\}$  by simply replacing the inputs  $\mathbf{x}_0, \ldots, \mathbf{x}_{n-1}$  and f with  $\overline{\mathbf{x}}_0, \ldots, \overline{\mathbf{x}}_{n-1}$  and  $\overline{f}$ . In general,  $\mathsf{aFE}$  is only secure if the adversary is not allowed to corrupt users. To see this, we first note that there exist vectors  $\overline{\mathbf{x}} \in \{0,1\}^{[k]\times\{0,1\}}$  that are not an extension of a vector  $\mathbf{x} \in \{0,1\}^{[k]}$ . More precisely,  $\overline{\mathbf{x}}$  is an extension of some  $\mathbf{x}$  if and only if  $\overline{\mathbf{x}}[j,0] = 1 - \overline{\mathbf{x}}[j,1]$  for all  $j \in [k]$ . Let us call such vectors  $\overline{\mathbf{x}}$  valid. By construction, an  $\mathsf{aFE}$  decryption key for a vector  $\mathbf{x}_i$  is an  $\overline{\mathsf{aFE}}$  decryption key for the extended vector  $\overline{\mathbf{x}}_i$ . Therefore, to reduce the security of  $\mathsf{aFE}$ to the security of  $\overline{\mathsf{aFE}}$ , we must argue that the adversary cannot obtain  $\overline{\mathsf{aFE}}$  decryption keys for vectors that are not valid. Without corruptions, this is easy to see. However, if the adversary can obtain a client's secret key, then this is no longer the case, as the adversary could generate decryption keys for invalid vectors by herself. Thus, using this conversion generically, we can convert the schemes in (1) and (2) into MC-ABEs for  $\mathsf{NC}^1$ without corruptions.

Moreover, we can even achieve security with corruptions when performing a *concrete* security analysis for the scheme in (2). For this, we recall that the secret key  $\mathsf{SK}_i$  of some client  $i \in [n-1]$  consists of an IBE master secret key that is used to generate decryption keys for identities  $(\mathsf{lab}, \overline{\mathbf{x}}_i)$ . Even though this key could be maliciously used to generate decryption keys for invalid vectors  $\overline{\mathbf{x}}_i$ , this does not help to win the security game as these identities do not occur in the challenge ciphertext.

**Other Policy Classes.** We recall from (2) that our scheme becomes inefficient when we choose  $k, n = \text{poly}(\lambda)$  since then there is an exponential number of  $\mathbf{x} = (\mathbf{x}_1, \dots, \mathbf{x}_{n-1}) \in \{0, 1\}^{[n-1]\times[k]}$ . Nevertheless, there exist nontrivial subclasses of NC<sup>1</sup> which do not require to consider all  $\mathbf{x}$  during the encryption procedure.

- $\mathsf{NC}^0$  Policies. The output of an  $\mathsf{NC}^0$  policy f depends only on a set L of size  $\tau = O(1)$  out of the total of kn inputs. As the remaining  $kn \tau$  inputs can be chosen arbitrarily without changing the output, it suffices to consider vectors  $\mathbf{x}$  that are 0 outside L. There are  $\binom{kn}{\tau} \leq (kn)^{\tau} = \operatorname{poly}(\lambda)$  possible sets L and for each choice we only need to consider  $2^{\tau} = O(1)$  vectors  $\mathbf{x}$ .
- Threshold Policies with Constant Threshold. Threshold policies with a threshold  $\tau \leq O(1)$  are not in NC<sup>0</sup> as they depend on all kn inputs. However, they have the property that each authorized set also has an authorized subset of size  $\tau$ . This allows an argument similar to above where we only deal with subsets L of size  $\tau$ . Symmetrically, we can also handle policies with a threshold  $kn \tau$  where we consider  $\binom{kn}{kn-\tau} \leq (kn)^{\tau} = \text{poly}(\lambda)$  sets L of size  $kn \tau$ .

While the idea is simple, the concrete implementation requires some care because it must

<sup>&</sup>lt;sup>5</sup>We can think of  $\overline{\mathbf{x}}$  as a vector of length 2k whose coordinates are indexed by the set  $[k] \times \{0, 1\}$  for convenience.

be guaranteed that the choices of L in the 0-th client remain compatible with the IBE keys provided by the other clients. For details, please see Section 5.2.

**Security.** To get a grasp of the security proof, it is instructive to first consider the case where the IPFE is simulation secure. This means the only values that the adversary learns are

- encodings  $[\![s]\!]_2$  of random scalars  $s \stackrel{\$}{\leftarrow} \mathbb{Z}_p$  sampled during the key generation and their corresponding share vectors  $[\![s]\!]_2$ , and
- target group encodings of the form  $[\![r \cdot s + \mu]\!]_t$  (IPFE instance 0) and  $[\![r \cdot \mathbf{s}[i, j]]\!]_t$ (IPFE instance (i, j)) for random scalars  $r \stackrel{\text{\tiny{e}}}{\leftarrow} \mathbb{Z}_p$  sampled during the encryption of the challenge message.

Importantly, nothing about r is leaked in  $\mathbb{G}_1$ . So we can rely on the DDH to obtain a fresh looking mask  $[\![r \cdot s]\!]_t$  with a fresh share vector  $[\![r \cdot s]\!]_2$  for each combination of r and s. Then we can exploit the one-time security provided by the LSS scheme to replace the individual secret sharings with random values: by the admissibility of the adversary, there does not exist any pair (r, s) such that the adversary has sufficient information about a subset of shares that allows her to recover the mask  $[\![r \cdot s]\!]_t$ . Instead, they look uniformly random and, thus, perfectly hide the challenge message.

Unfortunately, simulation security for many ciphertexts is known to be impossible in the standard model [BSW11]. Therefore, we can only rely on an IPFE with indistinguishabilitybased security. This makes the proof slightly more complex since we cannot directly conclude anymore that the adversary learns the scalars r only as part of the inner products encoded in  $\mathbb{G}_t$ . To circumvent this problem, we use a primitive called *slotted IPFE* [LL20a] which is a mix between public-key and private-key IPFE that provides standard security on the public part and additionally hides the function vectors in the private part. Using this primitive, we can move the scalar r from the message vectors encoded in  $\mathbb{G}_1$  into a hidden coordinate of the function vectors in  $\mathbb{G}_2$ . Subsequently, we can rely on the DDH in  $\mathbb{G}_2$  and proceed with the proof as in the case of simulation security.

Finally, we want to mention an important detail that occurs during the security proof. The adversary's admissibility condition only covers the case when she obtains at least one key for each slot (via either corruption or attribute key generation queries). Therefore, we must protect against so-called incomplete queries, where the adversary does not submit a query for every slot, but still has sufficient information to decrypt. In the context of IPFE (without access control) this can be done using a primitive called all-or-nothing encoding [CDSG<sup>+</sup>20]. In the context of attribute-based MIFE for attribute-weighted sums, [ATY23] uses a ciphertext-policy ABE for arithmetic branching programs which was recently proposed by Lin and Luo [LL20b]. In our case, we can avoid the usage of a complex primitive like ABE because we can model the completeness condition as part of our policies. This is feasible since our construction can check a global condition before releasing any information. Previous works considered only conjunctions of local checks in each slot which is not powerful enough to verify completeness.

### 2.2 MC-PE from MC-ABE and Lockable Obfuscation

**Lockable Obfuscation.** We employ a primitive called *lockable obfuscation* (LO) [GKW17, WZ17]. Roughly speaking, LO allows to obfuscate a circuit C with respect to a message  $\mu$  and a lock value  $\sigma$ . Correctness asks that an evaluation of the obfuscated circuit on some input x yields  $\mu$  if  $C(x) = \sigma$  and  $\perp$  otherwise. Simulation security requires that if  $\sigma$  looks random to the adversary, then the obfuscated circuit is computationally indistinguishable from a garbage program that does not carry any information about  $\mu$  or C.

The Compilers of [AYY22]. The authors of [AYY22] present two compilers from MI-ABE to MI-PE which nest several obfuscated circuits in a sophisticated manner. Very roughly, the obfuscated circuit  $\tilde{C}_0$  for the zeroth slot takes as input another obfuscated circuit  $\tilde{C}_1$  for the first slot, which in turn takes an obfuscated circuit  $\tilde{C}_2$  for the second slot and so on until one arrives at the last slot n-1.  $\tilde{C}_0$  is generated with respect to an attribute  $x_0$  and a message  $\mu$  whereas the other  $\tilde{C}_i$ 's only depend on an attribute  $x_i$ . The crucial part of the construction is to establish "communication" between consecutive circuits without violating attribute privacy. The idea is to build a recursive evaluation chain where the innermost circuit checks the condition  $f(x_0, \ldots, x_{n-1}) = 1$  using the MI-ABE; and a successful evaluation of an obfuscated circuit  $\tilde{C}_i$ , for  $i \in [n-1]$ , unlocks the lock and reveals a secret which is needed for a successful evaluation of  $\tilde{C}_{i-1}$ .

In their first compiler, these secret values are *global* secrets. This leads to a straightforward construction as all clients know these common secrets when they obfuscate their circuits. However, the supported security model is weak. This is because once the adversary submits any combination of oracle queries that enables a valid decryption process, these global secrets are revealed and security collapses even if all involved oracle queries have the same left and right input. To achieve security in a stronger model, their second compiler avoids these global secrets. However, this makes the construction more complex, and they are able to deal with only two slots. Our new construction can be viewed as a generalization of this arity-2 compiler to any constant arity. We therefore recall the arity-2 construction as a warm-up.

Construction in the Two-Input Setting. We start from an MI-ABE (aSetup, aEnc, aAKeyGen, aPKeyGen, aDec). For notational convenience, we use the shorthand notations  $\mathsf{aCT}_{\ell}(x_0,\mu)$ ,  $\mathsf{aDK}_{\ell}(i,x_i)$  and  $\mathsf{aDK}_{\ell}(f)$  to denote executions of  $\mathsf{aEnc}_{\ell}(\mathsf{aMPK}_{\ell},x_0,\mu)$ ,  $aAKeyGen(aSK_{\ell,i}, x_i)$  and  $aPKeyGen(aMSK_{\ell}, f)$ , for two independently generated aFE instances  $(\mathsf{aMPK}_{\ell}, \mathsf{aMSK}_{\ell}, \{\mathsf{aSK}_{\ell,i}\}_i) \leftarrow \mathsf{aSetup}(1^{\lambda})$  indexed by  $\ell \in \{0, 1\}$ . The MI-PE encryptor (client 0) possesses  $(\mathsf{aMPK}_0, \mathsf{aSK}_1)$  and the attribute key generator (client 1) possesses  $(aMPK_1, aSK_0)$ . The master secret key contains the ABE master secret keys  $(aMSK_0, aMSK_0)$ .  $aMSK_1$ ). To encrypt a message  $\mu$  with respect to an attribute  $x_0$ , client 0 samples a random lock value  $\sigma_0$  and computes  $\mathsf{aCT}_0(x_0, \sigma_0)$  and  $\mathsf{aDK}_1(1, x_0)$ . The final ciphertext is an obfuscation  $C_0$  of a circuit  $C_0[\mathsf{aCT}_0(x_0, \sigma_0), \mathsf{aDK}_1(1, x_0)]$  with respect to the message  $\mu$  and lock value  $\sigma_0$ . The notation  $C[\alpha]$  indicates that the value  $\alpha$  is hardwired in the description of the circuit C. Similarly, to produce a decryption key with respect to an attribute  $x_1$ , client 1 samples a lock value  $\sigma_1$ , generates  $\mathsf{aCT}_1(x_1, \sigma_1)$  and  $\mathsf{aDK}_0(1, x_1)$  and outputs an obfuscation  $\tilde{C}_1$  of a circuit  $C_1[\mathsf{aCT}_1(x_1,\sigma_1)]$  with respect to the message  $\mathsf{aDK}_0(1,x_1)$ and lock value  $\sigma_1$ . An MI-PE decryption key consists of a set of MI-ABE decryption keys { $\mathsf{aDK}_0(f), \mathsf{aDK}_1(f)$ }.

The pivotal point that makes the whole scheme work is the definition of the circuits. Specifically, decryption evaluates the obfuscated outer circuit  $\tilde{C}_0$  on input the obfuscated inner circuit  $\tilde{C}_1$  and the MI-ABE keys  $\{\mathsf{aDK}_0(f), \mathsf{aDK}_1(f)\}$ . Suppose that  $f(x_0, x_1) = 1$ . For a successful decryption, we must unlock  $\tilde{C}_0$ . The lock value  $\sigma_0$  is already hardwired in the circuit  $C_0[\mathsf{aCT}_0(x_0, \sigma_0), \mathsf{aDK}_1(1, x_0)]$ , however it is hidden in the ciphertext  $\mathsf{aCT}_0(x_0, \sigma_0)$ . To decrypt this ciphertext, we need the decryption key  $\mathsf{aDK}_0(1, x_1)$  embedded in  $\tilde{C}_1$ . For this reason  $C_0[\mathsf{aCT}_0(x_0, \sigma_0), \mathsf{aDK}_1(1, x_0)]$  starts by evaluating  $\tilde{C}_1$  on input  $(\mathsf{aDK}_1(1, x_0), \mathsf{aDK}_1(f))$ . From its inputs, the inner circuit  $C_1[\mathsf{aCT}_1(x_1, \sigma_1)]$  obtains everything it needs to decrypt its hardwired ciphertext  $\mathsf{aCT}_1(x_a, \sigma_1)$  and to recover the correct lock value  $\sigma_1$  which unlocks  $\tilde{C}_1$  and reveals  $\mathsf{aDK}_0(1, x_1)$ . At this point,  $C_0[\mathsf{aCT}_0(x_0, \sigma_0), \mathsf{aDK}_1(1, x_0)]$  can perform a similar computation by decrypting the ciphertext  $\mathsf{aCT}_0(x_0, \sigma_0)$  and recovering  $\sigma_0$ . This eventually unlocks  $\tilde{C}_0$  and outputs  $\mu$ . Importantly, this construction does not use global secrets, hence its security is not compromised after one successful decryption. **Generalization to Constant-Arity MC-ABE.** Our new compiler generalizes this framework to more than two slots and the more general MC-PE model. We will use independent MC-ABE instances for each slot to check if decryption is permitted, and each MC-PE client holds the key of one slot from each MC-ABE instance. Specifically, we let client  $i \in [0; n-1]$  control

- the *i*-th slot of the MC-ABE instances  $\ell \in [0; i-1]$ ,
- the 0-th slot of the MC-ABE instance  $\ell = i$ , and
- the (i + 1)-th slot of the MC-ABE instances  $\ell \in [i + 1; n 1]$ .

In particular, we note that each MC-PE client is the encryptor in exactly one of the MC-ABE schemes.

To encrypt a message  $\mu$  with respect to a label |ab| and an attribute  $x_0$ , client 0 samples a random lock value  $\sigma_0$  and creates  $aCT_0(|ab, x_0, \sigma_0)$  and  $aDK_\ell(|ab, 1, x_0)$  for all  $\ell \in [n-1]$ . Then, it issues an obfuscation of a circuit  $C_0[aCT_0(|ab, x_0, \sigma_0), \{aDK_\ell(|ab, 1, x_0)\}_{\ell \in [n-1]}]$ generated with respect to the message  $\mu$  and the lock value  $\sigma_0$ . Similarly, to generate a key for a label |ab| and an attribute  $x_i$ , client  $i \in [n-1]$  samples a lock value  $\sigma_i$ and creates  $aCT_i(|ab, x_i, \sigma_i)$ ,  $aDK_\ell(|ab, i, x_i)$  for  $\ell \in [0; i-1]$ , and  $aDK_\ell(|ab, i+1, x_i)$  for  $i \in [i+1; n-1]$ . Then it outputs an obfuscation of a circuit  $C_i[aCT_i, \{aDK_\ell(|ab, i+1, x_i)\}_{\ell \in [i+1; n-1]}]$  generated with respect to the message  $\{aDK_\ell(|ab, i, x_i)\}_{\ell \in [0; n-1]}$  and lock value  $\sigma_i$ . Decryption keys for a policy f are a set of MC-ABE keys  $\{aDK_\ell(f)\}_{\ell \in [0; n-1]}$ .

As in the two-input case, the crucial point is to establish communication between the obfuscated circuits in a secure way. However, the nested evaluations become more complex now. We first observe the following properties satisfied by all obfuscated circuits  $\tilde{C}_i$  for  $i \in [0; n-1]$ :

- 1. Decryption keys  $\mathsf{aDK}_{\ell}(\mathsf{lab}, i+1, x_i)$  for  $\ell > i$  are hardwired in the description of the circuit. This means they can be accessed during the evaluation of  $\widetilde{C}_i$  and passed as input to the evaluation of  $\widetilde{C}_i$  for j > i.
- 2. Decryption keys  $\mathsf{aDK}_{\ell}(\mathsf{lab}, i, x_i)$  for  $\ell < i$  are stored as the message of  $\widetilde{C}_i$  which is revealed in case of a successful evaluation. This means they can be recovered and used during the evaluation of  $\widetilde{C}_j$  for j < i.

Suppose that  $f(x_0, \ldots, x_{n-1}) = 1$ . Decryption evaluates  $\widetilde{C}_0$  on input the obfuscated circuits  $\{\widetilde{C}_i\}_{i \in [n-1]}$  and the MC-ABE keys  $\{\mathsf{aDK}_\ell(f)\}_{\ell \in [0;n-1]}$ . The lock value of  $\widetilde{C}_0$  is hidden in its hardwired ciphertext  $\mathsf{aCT}_0(\mathsf{lab}, x_0, \sigma_0)$ . To decrypt this ciphertext, we need the keys  $\{\mathsf{aDK}_0(\mathsf{lab}, i, x_i)\}_{i \in [n-1]}$  stored in the messages of  $\{\widetilde{C}_i\}_{i \in [n-1]}$ , so we need to evaluate them first. Specifically, via a chain of recursive calls where each  $\widetilde{C}_i$  invokes the evaluation of  $\widetilde{C}_{i+1}$ , we arrive at the evaluation of  $\widetilde{C}_{n-1}$ . From property 1, it follows that  $\widetilde{C}_{n-1}$  receives as input all the keys  $\{\mathsf{aDK}_{n-1}(\mathsf{lab}, i, x_i)\}_{i \in [n-1]}$  to decrypt its hardwired ciphertext  $\mathsf{aCT}_{n-1}(\mathsf{lab}, x_{n-1}, \sigma_{n-1})$  and to recover the lock value  $\sigma_{n-1}$ . In this way,  $\widetilde{C}_{n-1}$  can be unlocked and its message revealed. In the next step, it follows from property 2 that now the evaluation of  $\widetilde{C}_{n-2}$  has everything it needs to perform a similar computation to recover  $\sigma_{n-2}$  and unlock  $\widetilde{C}_{n-2}$ , and so on.

While at first glance it may seem that this decryption procedure is efficient for any (polynomial) number of slots, there is a subtle problem: each obfuscation increases the size of the circuit by a polynomial factor. As we nest the evaluation of the circuits, this leads to an exponential blow-up in the number of slots. Therefore, the decryption algorithm is only efficient for n = O(1), *i.e.* constant arity.

Security. The security proof is a simple sequence of hybrids over all slots from n-1 to 0. In each hybrid, if  $f(x_0, \ldots, x_{n-1}) = 0$ , then we can rely on the security of the *i*-th MC-ABE instance to replace the ciphertext  $\mathsf{aCT}_i(x_i, \sigma_i)$  hardwired in  $\widetilde{C}_i$  with a ciphertext of the zero string  $\mathsf{aCT}_i(x_i, 0)$ . Then the lock value  $\sigma_i$  appears random to the adversary and the obfuscated circuit  $\widetilde{C}_i$  can be replaced with a simulated obfuscation that carries no information about  $x_i$ . In the last step, we replace  $\widetilde{C}_0$  with a simulation that erases all information about  $x_0$  and  $\mu$ . We stress the importance of the fact that the global authorization is checked in each obfuscated circuit independently. Clearly, if a client  $i \in [0; n-1]$  is compromised, then the adversary can pick arbitrary attributes  $x_i$  and generate MC-ABE decryption keys  $\mathsf{aDK}_\ell(\mathsf{lab}, i+1, x_i)$  for  $\ell > i$  and  $\mathsf{aDK}_\ell(\mathsf{lab}, i, x_i)$  for  $\ell < i$ . But this does not help as long as the other clients  $j \in [0; n-1] \setminus \{i\}$  do not provide obfuscated circuits  $\widetilde{C}_j$  for attributes  $x_j$  such that  $f(x_0, \ldots, x_{n-1}) = 1$  for at least one attribute  $x_i$ . This is because the condition  $f(x_0, \ldots, x_{n-1}) = 1$  is checked in each obfuscated circuit  $\widetilde{C}_j$  using its own independent MC-ABE instance.

**Comparison with [FFMV23].** The authors of [FFMV23] build single-key MI-ABE for conjunctions of P. Their nesting technique bears similarities with ours which, in particular, leads to the same limitation that n = O(1). The key difference between the constructions is that they start from a *single-input PE* whereas we start from a *multi-client ABE*. This has a significant impact on the supported policy classes as well as the achieved security level.

For the former, we recall that single-input PE exists for powerful policy classes [GVW15], which is why their construction can evaluate arbitrary local predicates in each slot, but globally they are restricted to conjunctions. On the other hand, MC-ABE from standard assumptions is only known for simple policy classes (also constructed in this work). Nonetheless, these policies are not a conjunction, so the schemes' supported policy classes are incomparable.

To see the impact on security, we recall from the above sketch of the security proof that it is crucial to check the *global* authorization in each obfuscated circuit. Clearly, this is impossible with a single-input PE scheme. (Otherwise, we already had a multi-input PE scheme and there was no need for a compiler). For this reason, the security argument in [FFMV23] relies on the fact that there must exist a (fixed) non-corrupted slot *i* such that the local predicate is not satisfied for any attribute  $x_i$  that is input to a slot-*i* oracle query. In this case, the chain of nested evaluations breaks at slot *i* and the message embedded in  $\tilde{C}_0$  remains hidden. For two decryption keys DK<sub>f</sub> and DK'<sub>f</sub>, the evaluation chains could break at different slots  $i \neq i'$ . So while these keys might not be individually authorized for decryption, together they could be used to establish a complete decryption chain, which is the reason that the scheme in [FFMV23] is not secure under collusions. On the other hand, our MC-ABE scheme allows to check the global predicate in each step of the nested evaluation. Hence, mixing two keys does not help if neither of them is individually authorized, and we obtain security even under collusions.

# **3** Preliminaries

### 3.1 Notational Conventions

Let  $\lambda \in \mathbb{N}$  be the security parameter. Except in the definitions, we will suppress  $\lambda$  in subscripts for brevity. A nonnegative function  $\varepsilon \colon \mathbb{N} \to \mathbb{R}$  is negligible if  $\varepsilon(\lambda) = O(\lambda^{-n})$  for all  $n \in \mathbb{N}$ . An algorithm is said to be *efficient* if it runs in probabilistic polynomial time (PPT) in the security parameter.

To avoid confusion, we always write vectors  $\mathbf{v}$  and matrices  $\mathbf{A}$  in boldface and use uppercase letters for the latter. Scalars *s* are written in italics. Unless otherwise stated, all vectors  $\mathbf{v}$  are viewed as column vectors. The corresponding row vector is denoted by  $\mathbf{v}^{\top}$ .

Security Experiments and Distributions. Let  $\mathbf{Exp}$  be an interactive *experiment* that interacts with an algorithm  $\mathcal{A}$  (called the *adversary*), depends on the security parameter  $\lambda$  and has binary outcome. We also refer to such objects as *games* or *hybrids*. We let " $\mathbf{Exp}_{\mathcal{A}}(1^{\lambda}) \to 1$ " denote the event that the outcome of running  $\mathbf{Exp}$  with  $\mathcal{A}$  on security parameter  $\lambda$  is 1. For two experiments  $\mathbf{Exp}^0$  and  $\mathbf{Exp}^1$ , we define the distinguishing advantage of  $\mathcal{A}$  against the tuple ( $\mathbf{Exp}^0, \mathbf{Exp}^1$ ) as

$$\mathbf{Adv}_{\mathbf{Exp}^{0},\mathbf{Exp}^{1},\mathcal{A}}(\lambda) \coloneqq \left| \Pr\left[ \mathbf{Exp}_{\mathcal{A}}^{1}(1^{\lambda}) \to 1 \right] - \Pr\left[ \mathbf{Exp}_{\mathcal{A}}^{0}(1^{\lambda}) \to 1 \right] \right|$$

We write  $\mathbf{Exp}^0 \approx_c \mathbf{Exp}^1$  if the experiments are computationally indistinguishable, i.e. their distinguishing advantage is negligible for all efficient adversaries  $\mathcal{A}$ . We write  $\mathbf{Exp}^0 \approx_s \mathbf{Exp}^1$  if the experiments are statistically indistinguishable, i.e. their distinguishing advantage is negligible for all (even unbounded) adversaries. We write  $\mathbf{Exp}^0 \equiv \mathbf{Exp}^1$  if the experiments are *identically distributed*, *i.e.* their distinguishing advantage is 0 for all (even unbounded) adversaries. By default, the term *indistinguishable* refers to computational indistinguishability.

More general, the same notations can be used for sequences of distributions. Let  $D^0 = \{D^0_\lambda\}_{\lambda \in \mathbb{N}}$  and  $D^1 = \{D^1_\lambda\}_{\lambda \in \mathbb{N}}$  be two sequences of distributions. For  $b \in \{0, 1\}$ , we define  $\operatorname{Exp}^b_{\mathcal{A}}(1^{\lambda})$  as follows: sample  $x \stackrel{\$}{\leftarrow} D^b_\lambda$ , run  $\mathcal{A}(1^{\lambda}, x)$  and use the output of  $\mathcal{A}$  as the outcome of the experiment. Then we write  $D^0 \approx_c D^1$  (resp.  $D^0 \approx_s D^1$ ,  $D^0 \equiv D^1$ ) if  $\operatorname{Exp}^0_{\mathcal{A}} \approx_c \operatorname{Exp}^1_{\mathcal{A}}$  (resp.  $\operatorname{Exp}^0_{\mathcal{A}} \approx_s \operatorname{Exp}^1_{\mathcal{A}}$ ).

**Sets and Indexing.** We denote by  $\mathbb{Z}$  and  $\mathbb{N}$  the sets of integers and natural numbers (positive integers). For integers m and n, we write [m;n] to denote the set  $\{z \in \mathbb{Z} : m \leq z \leq n\}$  and let  $[n] \coloneqq [1;n]$ . For a prime number p,  $\mathbb{Z}_p$  denotes the finite field of integers modulo p. For a finite set S, we let  $2^S$  denote the power set of S.

To index a vector or the columns of a matrix, we write  $\mathbf{v}[i]$  and  $\mathbf{A}[j]$ . In contrast, objects of some collection that is not regarded as a vector or matrix are indexed using subscripts (or superscripts in some cases). For instance,  $\mathbf{v}_i$  represents a vector, not a component of some vector. If *i* runs through some index set [n], it means that there are *n* vectors  $\mathbf{v}_1, \ldots, \mathbf{v}_n$ . If the *n* objects are scalars (or not explicitly vectors), we will write  $v_1, \ldots, v_n$  instead.

For convenience, objects might be indexed by arbitrary sets, not just integers. For finite sets  $\mathfrak{s}, A$ , we write  $A^{\mathfrak{s}} := \{(\mathbf{v}[i])_{i \in \mathfrak{s}} : \mathbf{v}[i] \in A\}$  for the set of vectors whose entries are in A and indexed by  $\mathfrak{s}, e.g. \mathbb{Z}_p^{[n]}$  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 a vector  $\mathbf{v} \in \mathbb{Z}_p^{\mathfrak{s}_2}$ , we denote by  $\mathbf{u} = \mathbf{v}|_{\mathfrak{s}_1}$  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 all  $i \in \mathfrak{s}_1$ . Conversely, for any vector  $\mathbf{u} \in \mathbb{Z}_p^{\mathfrak{s}_1}$ , we write  $\mathbf{v} = \mathbf{u}|_{\mathfrak{s}_2}$  for its zero-extension into  $\mathbb{Z}_p^{\mathfrak{s}_2}$ , *i.e.*  $\mathbf{v} \in \mathbb{Z}_p^{\mathfrak{s}_2}$  and  $\mathbf{v}[i] = \mathbf{u}[i]$  if  $i \in \mathfrak{s}_1$ , and  $\mathbf{v}[i] = 0$  if  $i \in \mathfrak{s}_2 \setminus \mathfrak{s}_1$ .

#### 3.2 Pairing Groups and Hardness Assumptions

Pairing Groups. Our constructions use a sequence of pairing groups

$$\mathbb{G} = \{\mathbb{G}_{\lambda} = (\mathbb{G}_{\lambda,1}, \mathbb{G}_{\lambda,2}, \mathbb{G}_{\lambda,t}, g_{\lambda,1}, g_{\lambda,2}, g_{\lambda,t}, e_{\lambda}, p_{\lambda})\}_{\lambda \in \mathbb{N}} ,$$

where  $\mathbb{G}_{\lambda,1}$  (resp.  $\mathbb{G}_{\lambda,2}$ ,  $\mathbb{G}_{\lambda,t}$ ) is a cyclic group of order  $p_{\lambda}$  generated by  $g_{\lambda,1}$  (resp.  $g_{\lambda,2}$ ,  $g_{\lambda,t}$ ), and  $e_{\lambda} : \mathbb{G}_{\lambda,1} \times \mathbb{G}_{\lambda,2} \to \mathbb{G}_{\lambda,t}$  is the pairing operation satisfying  $e_{\lambda}(g_{\lambda,1}^{a}, g_{\lambda,2}^{b}) = g_{\lambda,t}^{ab}$  for all integers a, b. The group operations and the pairing map are required to be efficiently computable.

Following the implicit notation in [EHK<sup>+</sup>13], we write  $[a]_i$  to denote  $g^a_{\lambda,i}$  for  $i \in \{1, 2, t\}$ . This notation extends component-wise to matrices and vectors having entries in  $\mathbb{Z}_p$ . Equipped with these notations, group operations are written additively and the pairing operation multiplicatively, *e.g.*  $[\![\mathbf{A}]\!]_1 - \mathbf{B}[\![\mathbf{C}]\!]_1 \mathbf{D} = [\![\mathbf{A} - \mathbf{B}\mathbf{C}\mathbf{D}]\!]_1$  and  $[\![\mathbf{A}]\!]_1[\![\mathbf{B}]\!]_2 = [\![\mathbf{A}\mathbf{B}]\!]_t$ .

**Computational Assumptions.** We state the assumptions needed for our constructions. Let  $\{\mathbb{G}_{\lambda} = (\mathbb{G}_{\lambda,1}, \mathbb{G}_{\lambda,2}, \mathbb{G}_{\lambda,t}, g_{\lambda,1}, g_{\lambda,2}, g_{\lambda,t}, e_{\lambda}, p_{\lambda})\}_{\lambda \in \mathbb{N}}$  be a sequence of pairing groups.

**Definition 1** (Decisional Diffie-Hellman Assumption (DDH)). Let  $i \in \{1, 2, t\}$ . The DDH assumption holds in  $\{\mathbb{G}_{\lambda,i}\}_{\lambda \in \mathbb{N}}$  if  $\{[[a, b, ab]]_i\}_{\lambda \in \mathbb{N}} \approx_c \{[[a, b, ab + c]]_i\}_{\lambda \in \mathbb{N}}$  for  $a, b, c \notin \mathbb{Z}_{p_{\lambda}}$ .

**Definition 2** (Symmetric eXternal Diffie-Hellman Assumption (SXDH)). The SXDH assumption holds in  $\{\mathbb{G}_{\lambda}\}_{\lambda\in\mathbb{N}}$  if the DDH assumption holds in both  $\{\mathbb{G}_{\lambda,1}\}_{\lambda\in\mathbb{N}}$  and  $\{\mathbb{G}_{\lambda,2}\}_{\lambda\in\mathbb{N}}$ .

#### 3.3 Monotone Access Structures and Linear Secret Sharing Schemes

Let  $\mathcal{X} = \{0,1\}^{\mathfrak{s}}$  be the attribute universe with index set  $\mathfrak{s}$ . An access structure on  $\mathcal{X}$  is a collection  $\mathcal{S} \subseteq 2^{\mathfrak{s}} \setminus \emptyset$  of nonempty subsets of  $\mathfrak{s}$ . We call the sets in  $\mathcal{S}$  authorized, and those in  $2^{\mathfrak{s}} \setminus \mathcal{S}$  unauthorized. Each access structure  $\mathcal{S}$  corresponds to an access policy  $f: \mathcal{X} \to \{0,1\}$  defined via

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } \{i \in \mathfrak{s} : \mathbf{x}[i] = 1\} \in \mathcal{S} \\ 0 & \text{if } \{i \in \mathfrak{s} : \mathbf{x}[i] = 1\} \notin \mathcal{S} \end{cases}.$$

An access structure  $S \subseteq 2^{\mathfrak{s}}$  is said to be *monotone* if the following condition is satisfied for all  $S_1, S_2 \subseteq \mathfrak{s}$ : if  $S_1 \in S$  and  $S_1 \subseteq S_2$ , then  $S_2 \in S$ . A policy is said to be monotone if its corresponding access structure is monotone.

We next recall the definition of a linear secret sharing scheme.

**Definition 3** (Linear Secret Sharing (LSS) Scheme [BL90, Bei96]). Let  $\ell, n \in \mathbb{N}$  and p be a prime number. We denote  $\mathbf{e}_1 = (1, 0, \dots, 0)^\top \in \mathbb{Z}_p^n$  the first unit-vector in  $\mathbb{Z}_p^n$ . A *linear secret sharing (LSS) scheme* over  $\mathbb{Z}_p$  for an access structure  $S \subseteq 2^{\mathfrak{s}}$  on an attribute universe  $\mathcal{X} = \{0, 1\}^{\mathfrak{s}}$  is specified by a share generating matrix  $\mathbf{M} \in \mathbb{Z}_p^{n \times \ell}$  and a function  $\rho: [\ell] \to \mathfrak{s}$  mapping the columns of  $\mathbf{M}$  to indices in  $\mathfrak{s}$ , which satisfy the following condition:

$$S \in \mathcal{S} \iff \mathbf{e}_1 \in \operatorname{span}\{\mathbf{M}[j] : j \in [\ell], \rho(j) \in S\}$$
 (3)

For convenience, we often do not distinguish between an access structure S, its corresponding policy f and a pair  $(\mathbf{M}, \rho)$  satisfying (3). In particular, we may write  $f = (\mathbf{M}, \rho)$ . In order to share a value  $s \in \mathbb{Z}_p$  using an LSS scheme over  $\mathbb{Z}_p$ , one samples  $\mathbf{u} \stackrel{s}{\leftarrow} \mathbb{Z}_p^{n-1}$  and computes the share vector

$$\mathbf{s} = (s, \mathbf{u}[1], \dots, \mathbf{u}[n-1]) \cdot \mathbf{M} \in \mathbb{Z}_p^{\ell}$$
.

Then a set  $\{\mathbf{s}[j]\}_{j\in J}$  for some  $J \subseteq [\ell]$  can be used to reconstruct s if and only if  $\{\rho(j)\}_{j\in J}$  is authorized with respect to the access structure corresponding to  $f = (\mathbf{M}, \rho)$ . Indeed, in this case there exist coefficients  $\omega_1, \ldots, \omega_\ell \in \mathbb{Z}_p$  such that  $\omega_j = 0$  for all  $j \in [\ell] \setminus J$  and  $\sum_{j \in [\ell]} \omega_j \mathbf{M}[j] = \mathbf{e}_1$ . These coefficients can be used to compute

$$\sum_{j \in J} \omega_j \mathbf{s}[j] = \sum_{j \in [\ell]} \omega_j \mathbf{s}[j] = (s, \mathbf{u}[1], \dots, \mathbf{u}[n-1]) \cdot \sum_{j \in [\ell]} \omega_j \mathbf{M}[j] = s$$

Lewko and Waters [LW11] presented an LSS scheme for all monotone access structures.

#### 3.4 Function-Hiding Slotted Inner-Product Functional Encryption

We recall the definition of slotted IPFE from [LL20a]. Similar to [LL20a, AWY20, LLL22], this primitive will allow us to employ techniques akin to dual system encryption [Wat09, LW10]. To adhere to the formalism used in this work, we present the syntax in a pairing-based setting.

**Definition 4** (Slotted IPFE). Let  $\mathbb{G} = \{\mathbb{G}_{\lambda} = (\mathbb{G}_{\lambda,1}, \mathbb{G}_{\lambda,2}, \mathbb{G}_{\lambda,t}, g_{\lambda,1}, g_{\lambda,2}, g_{\lambda,t}, e_{\lambda}, p_{\lambda})\}_{\lambda \in \mathbb{N}}$  be a sequence of pairing groups. A *slotted IPFE scheme based on*  $\mathbb{G}$  consists of five efficient algorithms:

- $\begin{aligned} \mathsf{Setup}(1^\lambda,\mathfrak{s}_{\mathsf{pub}},\mathfrak{s}_{\mathsf{pri}}) &\to (\mathsf{MPK},\mathsf{MSK}) \text{: On input the security parameter and two disjoint} \\ & \text{index sets, the public slot } \mathfrak{s}_{\mathsf{pub}} \text{ and the private slot } \mathfrak{s}_{\mathsf{pri}}, \text{ this algorithm outputs a pair} \\ & \text{of a master public and a master secret key (MPK, \mathsf{MSK}). We denote the whole index} \\ & \text{set by } \mathfrak{s} \coloneqq \mathfrak{s}_{\mathsf{pub}} \cup \mathfrak{s}_{\mathsf{pri}}. \end{aligned}$
- $\mathsf{Enc}(\mathsf{MSK}, [\![\mathbf{x}]\!]_1) \to \mathsf{CT}$ : On input a master secret key MSK and an encoding of a vector  $\mathbf{x} \in \mathbb{Z}_{p_{\lambda}}^{\mathfrak{s}}$  in  $\mathbb{G}_{\lambda,1}$ , this algorithm outputs a ciphertext CT for  $\mathbf{x}$ .
- $\text{KeyGen}(\text{MSK}, \llbracket \mathbf{y} \rrbracket_2) \to \text{SK}$ : On input a master secret key MSK and an encoding of a vector  $\mathbf{y} \in \mathbb{Z}_{n_{\lambda}}^{\mathfrak{s}}$  in  $\mathbb{G}_{\lambda,2}$ , this algorithm outputs a decryption key DK for  $\mathbf{y}$ .
- $\mathsf{Dec}(\mathsf{DK},\mathsf{CT}) \to \llbracket d \rrbracket_t$ : On input a decryption key  $\mathsf{DK}$  and a ciphertext  $\mathsf{CT}$ , this algorithm outputs an element  $\llbracket d \rrbracket_t \in \mathbb{G}_{\lambda,t}$ .
- SlotEnc(MPK,  $[\![\mathbf{x}_{pub}]\!]_1$ )  $\rightarrow$  CT: On input a master public key MPK and an encoding of a message vector  $\mathbf{x}_{pub} \in \mathbb{Z}_{p_{\lambda}}^{\mathfrak{s}_{pub}}$  in  $\mathbb{G}_{\lambda,1}$ , this algorithm outputs a ciphertext for the vector  $\mathbf{x} = \mathbf{x}_{pub}|^{\mathfrak{s}} \in \mathbb{Z}_{p_{\lambda}}^{\mathfrak{s}}$ .

**Correctness.** A slotted IPFE scheme satisfies *decryption correctness* if for all  $\lambda \in \mathbb{N}$ , all disjoint index sets  $\mathfrak{s}_{pub}, \mathfrak{s}_{pri}$  and all vectors  $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_{p_{\lambda}}^{s}$ , it holds that

$$\Pr\left[\mathsf{Dec}(\mathsf{DK},\mathsf{CT}) = \llbracket\langle \mathbf{x}, \mathbf{y} \rangle \rrbracket_t \; \left| \begin{array}{c} (\mathsf{MPK},\mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda,\mathfrak{s}_{\mathsf{pub}},\mathfrak{s}_{\mathsf{pri}}) \\ \mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{MSK},\llbracket \mathbf{x} \rrbracket_1) \\ \mathsf{DK} \leftarrow \mathsf{KeyGen}(\mathsf{MSK},\llbracket \mathbf{y} \rrbracket_2) \end{array} \right] = 1$$

Furthermore, we say that a slotted IPFE scheme satisfies *slot-mode correctness* if for all  $\lambda \in \mathbb{N}$ , all disjoint index sets  $\mathfrak{s}_{pub}, \mathfrak{s}_{pri}$  and  $\mathbf{x}_{pub} \in \mathbb{Z}_p^{\mathfrak{s}_{pub}}$ , the following distributions  $\mathcal{D}_0, \mathcal{D}_1$  are identical:

$$\begin{split} \mathcal{D}_0 &= \left\{ (\mathsf{MPK},\mathsf{MSK},\mathsf{CT}) \; \left| \begin{array}{c} (\mathsf{MPK},\mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda,\mathfrak{s}_{\mathsf{pub}},\mathfrak{s}_{\mathsf{pri}}) \\ \mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{MSK},[\![\mathbf{x}_{\mathsf{pub}}]\!]_1) \end{array} \right\} \;\;, \\ \mathcal{D}_1 &= \left\{ (\mathsf{MPK},\mathsf{MSK},\mathsf{CT}) \; \left| \begin{array}{c} (\mathsf{MPK},\mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda,\mathfrak{s}_{\mathsf{pub}},\mathfrak{s}_{\mathsf{pri}}) \\ \mathsf{CT} \leftarrow \mathsf{SlotEnc}(\mathsf{MPK},[\![\mathbf{x}_{\mathsf{pub}}]\!]_1) \end{array} \right\} \;\;, \end{split} \right. \end{split}$$

where the probability is taken over the random coins of the algorithms.

Security. We define adaptive function-hiding IND-CPA security.

**Definition 5** (Function-Hiding Security). For a slotted IPFE scheme iFE and a PPT adversary  $\mathcal{A}$ , we define the security experiment  $\mathbf{Exp}_{\mathsf{iFE},\mathcal{A}}^{\mathsf{sl-ipfe-b}}(1^{\lambda})$  as shown in Figure 1. The oracles  $\mathcal{O}\mathsf{KeyGen}$  and  $\mathcal{O}\mathsf{Enc}$  can be called in any order and any (polynomial) number of times. The adversary  $\mathcal{A}$  is admissible with respect to  $\mathcal{Q}_{\mathsf{enc}}$  and  $\mathcal{Q}_{\mathsf{key}}$ , denoted by  $\mathsf{adm}(\mathcal{A}) = 1$ , if all  $(\llbracket \mathbf{x}_0 \rrbracket_1, \llbracket \mathbf{x}_1 \rrbracket_1) \in \mathcal{Q}_{\mathsf{enc}}$  and  $(\llbracket \mathbf{y}_0 \rrbracket_2, \llbracket \mathbf{y}_1 \rrbracket_2) \in \mathcal{Q}_{\mathsf{key}}$  satisfy  $\langle \mathbf{x}_0, \mathbf{y}_0 \rangle = \langle \mathbf{x}_1, \mathbf{y}_1 \rangle$  and  $\mathbf{y}_0|_{\mathfrak{s}_{\mathsf{pub}}} = \mathbf{y}_1|_{\mathfrak{s}_{\mathsf{pub}}}$ . Otherwise, we say that  $\mathcal{A}$  is not admissible and write  $\mathsf{adm}(\mathcal{A}) = 0$ . We call iFE function-hiding if  $\mathbf{Exp}_{\mathsf{iFE},\mathcal{A}}^{\mathsf{sl-ipfe-1}}(1^{\lambda}) \approx_c \mathbf{Exp}_{\mathsf{iFE},\mathcal{A}}^{\mathsf{sl-ipfe-1}}(1^{\lambda})$ .

$ \begin{array}{l} & \frac{Initialize(1^{\lambda}, \mathfrak{s}_{pub}, \mathfrak{s}_{pri}):}{\mathcal{Q}_{enc}, \mathcal{Q}_{key} \leftarrow \varnothing} \\ & (MPK, MSK) \leftarrow Setup(1^{\lambda}, \mathfrak{s}_{pub}, \mathfrak{s}_{pri}) \end{array} $	$ \begin{array}{l} \displaystyle \frac{\mathcal{O}KeyGen([\![\mathbf{y}_0]\!]_2, [\![\mathbf{y}_1]\!]_2):}{\mathcal{Q}_{key} \leftarrow \mathcal{Q}_{key} \cup \{([\![\mathbf{y}_0]\!]_2, [\![\mathbf{y}_1]\!]_2)\} \\ \mathrm{Return} \ DK \leftarrow KeyGen(MSK, [\![\mathbf{y}_b]\!]_2) \end{array} $
Return MPK	
	Finalize $(b')$ :
$\mathcal{O}Enc(\llbracket \mathbf{x}_0 \rrbracket_2, \llbracket \mathbf{x}_1 \rrbracket_2)$ :	If $\operatorname{adm}(\mathcal{A}) = 1$ , return $\beta \leftarrow (b' \stackrel{?}{=} b)$
$\overline{\mathcal{Q}_{enc}} \leftarrow \overline{\mathcal{Q}_{enc}} \cup \{(\llbracket \mathbf{x}_0 \rrbracket_1, \llbracket \mathbf{x}_1 \rrbracket_1)\}$ Return CT \leftarrow Enc(MSK, $\llbracket \mathbf{x}_h \rrbracket_1)$	Else, return a random bit $\beta \stackrel{\hspace{0.1em} \leftarrow}{\leftarrow} \{0,1\}$



There exists a slotted IPFE scheme based on  $\mathbb{G}$  which can be proven (adaptively) function-hiding under the SXDH<sup>6</sup> assumption in  $\mathbb{G}$ . The construction is based on a sequence of works [ALS16, Wee17, LV16, Lin17] and has been described explicitly in [LL20a].

#### 3.5 Identity-Based Encryption

We recall the definition of identity-based encryption (IBE).

**Definition 6** (Identity-Based Encryption). Let  $\mathcal{M} = {\mathcal{M}_{\lambda}}_{\lambda \in \mathbb{N}}$  and  $\mathcal{I} = {\mathcal{I}_{\lambda}}_{\lambda \in \mathbb{N}}$  be sequences of message and identity spaces, respectively. An *identity-based encryption scheme* for  $\mathcal{M}$  and  $\mathcal{I}$  consists of four efficient algorithms:

- $\mathsf{Setup}(1^{\lambda}) \to (\mathsf{MPK}, \mathsf{MSK})$ : On input the security parameter, this algorithm outputs a pair of a master public key MPK and a master secret key MSK.
- $\mathsf{Enc}(\mathsf{MPK}, i, \mu) \to \mathsf{CT}$ : On input a master public key MSK, an identity  $i \in \mathcal{I}_{\lambda}$  and a message  $\mu \in \mathcal{M}_{\lambda}$ , this algorithm outputs a ciphertext CT for  $\mu$  created with respect to i.
- $\mathsf{KeyGen}(\mathsf{MSK}, i') \to \mathsf{DK}$ : On input a master secret key  $\mathsf{MSK}$  and an identity  $i' \in \mathcal{I}_{\lambda}$ , this algorithm outputs a decryption key  $\mathsf{DK}$  for i'.
- $\mathsf{Dec}(\mathsf{DK},\mathsf{CT}) \to \mu' \lor \bot$ : On input a decryption key  $\mathsf{DK}$  and a ciphertext  $\mathsf{CT}$ , this algorithm outputs an element  $\mu' \in \mathcal{M}_{\lambda}$  or  $\bot$ .

**Correctness.** An IBE scheme is said to be *correct* if for all  $\lambda \in \mathbb{N}$ , all identities  $i \in \mathcal{I}_{\lambda}$  and all messages  $\mu \in \mathcal{M}_{\lambda}$ , it holds that

$$\Pr\left[ \mathsf{Dec}(\mathsf{DK},\mathsf{CT}) = \mu \; \left| \begin{array}{c} (\mathsf{MPK},\mathsf{MSK}) \leftarrow \mathsf{Setup}(1^{\lambda}) \\ \mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{MPK},i,\mu) \\ \mathsf{DK} \leftarrow \mathsf{KeyGen}(\mathsf{MSK},i) \end{array} \right] = 1 \; ,$$

where the probability is taken over the random coins of the algorithms.

Security. We define adaptive IND-CPA security.

**Definition 7** (Security). For an IBE scheme IBE and a PPT adversary  $\mathcal{A}$ , we define the security experiment  $\mathbf{Exp}_{\mathsf{IBE},\mathcal{A}}^{\mathsf{ibe-b}}(1^{\lambda})$  as shown in Figure 2. The oracle  $\mathcal{O}\mathsf{KeyGen}$  can be called any (polynomial) number of times whereas the oracle  $\mathcal{O}\mathsf{Enc}$  can be called only once. We call IBE secure if  $\mathbf{Exp}_{\mathsf{IBE},\mathcal{A}}^{\mathsf{ibe-0}}(1^{\lambda}) \approx_c \mathbf{Exp}_{\mathsf{IBE},\mathcal{A}}^{\mathsf{ibe-1}}(1^{\lambda})$ .

There exist various IBE schemes in the group-based setting, e.g. [Wat09, CLL+13, JR17].

<sup>&</sup>lt;sup>6</sup>More precisely, the security proof only relies on  $\mathsf{MDDH}_k$ , for any k > 1, in both  $\mathbb{G}_1$  and  $\mathbb{G}_2$ . This assumption is implied by SXDH on  $\mathbb{G}$ .

Initialize $(1^{\lambda})$ :	$\underbrace{\mathcal{O}Enc(i,\mu^0,\mu^1):}$
$\mathcal{Q} \leftarrow arnothing; i_{enc} \leftarrow \bot$	$i_{enc} \leftarrow i$
$(MPK,MSK) \leftarrow Setup(1^{\lambda})$	Return $CT \leftarrow Enc(MPK, i, \mu^b)$
Return MPK	
	Finalize $(b')$ :
$\mathcal{O}KeyGen(i')$ :	If $i_{enc} \notin \mathcal{Q}$ , return $\beta \leftarrow (b' \stackrel{?}{=} b)$
$\overline{\mathcal{Q} \leftarrow \mathcal{Q} \cup \{i'\}}$	Else, return a random bit $\beta \stackrel{\hspace{0.1em}\text{\tiny\$}}{\leftarrow} \{0,1\}$
$\text{Return} \ DK \leftarrow KeyGen(MSK, i')$	

Figure 2: Security game  $\mathbf{Exp}_{\mathsf{IBE},\mathcal{A}}^{\mathsf{ibe},b}(1^{\lambda})$  for Definition 7

### 3.6 Lockable Obfuscation

We recall the definition of a lockable obfuscator [GKW17, WZ17]. Given polynomials  $n = n(\lambda), m = m(\lambda)$  and  $d = d(\lambda)$ , we denote by  $C_{n,m,d}(\lambda)$  the class of depth  $d(\lambda)$  circuits with  $n(\lambda)$  bits input and  $m(\lambda)$  bits output.

**Definition 8** (Lockable Obfuscation). Let  $\mathcal{M} = {\mathcal{M}_{\lambda}}_{\lambda \in \mathbb{N}}$  be a sequence of message spaces and  ${\mathcal{C}_{n,m,d}(\lambda)}_{\lambda \in \mathbb{N}}$  a sequence of circuit classes. A lockable obfuscator for  $\mathcal{M}$  and  $\mathcal{C}$  is a tuple of two efficient algorithms:

 $\mathsf{Obf}(1^{\lambda}, C, \mu, \sigma) \to (\widetilde{C})$ : On input  $1^{\lambda}$ , a circuit  $C \in \mathcal{C}_{n,m,d}(\lambda)$ , a message  $\mu \in \mathcal{M}_{\lambda}$  and a "lock value"  $\sigma \in \{0, 1\}^{m(\lambda)}$ , this algorithm outputs an obfuscated circuit  $\widetilde{C}$ .

 $\mathsf{Eval}(\widetilde{C}, x) \to \mu' \lor \bot$ : On input an obfuscated circuit  $\widetilde{C}$  and an input  $x \in \{0, 1\}^{n(\lambda)}$ , this algorithm outputs a value  $\mu' \in \mathcal{M}_{\lambda}$  or  $\bot$ .

**Correctness.** A lockable obfuscator satisfies *(perfect) correctness* if for all  $\lambda \in \mathbb{N}$ , all circuits  $C \in \mathcal{C}_{n,m,d}(\lambda)$ , all messages  $\mu \in \mathcal{M}_{\lambda}$  and all inputs  $x \in \{0,1\}^{n(\lambda)}$ , the following two implications are satisfied:

- 1. if  $C(x) = \sigma$ , then  $\mathsf{Eval}(\mathsf{Obf}(1^{\lambda}, C, \mu, \sigma), x) = \mu$
- 2. if  $C(x) \neq \sigma$ , then  $\mathsf{Eval}(\mathsf{Obf}(1^{\lambda}, C, \mu, \sigma), x) = \bot$

**Security.** We define security against multiple challenges. In [AYY22], this definition was observed to be equivalent to the original single-challenge version from [GKW17].

**Definition 9** (Security against Multiple Queries). For a lockable obfuscation scheme LObf = (Obf, Eval) and an efficient algorithm Sim, we define the following oracles:

 $\mathcal{O}\mathsf{Obf}^0(C,\mu)$ : sample  $\sigma \xleftarrow{\hspace{0.1em}\$} \{0,1\}^{m(\lambda)}$  and return  $\widetilde{C} \leftarrow \mathsf{Obf}(1^\lambda, C, \mu, \sigma)$ 

 $\mathcal{O}\mathsf{Obf}^1(C,\mu)$ : return  $\mathsf{Sim}(1^{\lambda},1^{|C|},1^{|\mu|})$ 

We call LObf *secure* if there exists a PPT simulator Sim such that for all PPT adversaries  $\mathcal{A}$ , there exists a negligible function negl(·) such that

$$\mathbf{Adv}_{\mathsf{LObf},\mathcal{A}}^{\mathsf{lock}}(\lambda) \coloneqq \left| \Pr\left[ \mathcal{A}^{\mathcal{O}\mathsf{Obf}^1} \to 1 \right] - \Pr\left[ \mathcal{A}^{\mathcal{O}\mathsf{Obf}^0} \to 1 \right] \right| \le \operatorname{negl}(\lambda) \ .$$

Perfectly correct lockable obfuscators for general circuits are known to exist under the LWE assumption [GKW17, GKVW20].

# 4 Multi-Client Attribute-Based and Predicate Encryption

We define multi-client attribute-based encryption (MC-ABE) and multi-client predicate encryption (MC-PE). Since the only difference between these notions lies in the security game, we unify the syntax of the algorithms.

**Definition 10** (Public-Key Syntax). Let  $n = n(\lambda)$  be a polynomial. Furthermore, let  $\mathcal{M} = \{\mathcal{M}_{\lambda}\}_{\lambda \in \mathbb{N}}$  be a sequence of message spaces,  $\mathcal{X} = \{\mathcal{X}_{\lambda}\}_{\lambda \in \mathbb{N}}$  a sequence of attribute universes,  $\mathcal{L} = \{\mathcal{L}_{\lambda}\}_{\lambda \in \mathbb{N}}$  a sequence of label spaces and  $\mathcal{F} = \{\mathcal{F}_{\lambda}\}_{\lambda \in \mathbb{N}}$  a sequence of policy classes, where each policy  $f_{\lambda} \in \mathcal{F}_{\lambda}$  maps from  $\mathcal{X}_{\lambda}^{n}$  to  $\{0, 1\}$ . An *MC-ABE* (resp. *MC-PE*) scheme for  $\mathcal{M}, \mathcal{X}, \mathcal{F}$  and  $\mathcal{L}$  consists of five efficient algorithms:

- $\mathsf{Setup}(1^{\lambda}) \to (\mathsf{MPK}, \mathsf{MSK}, \{\mathsf{SK}_i\}_{i \in [n-1]})$ : On input the security parameter  $1^{\lambda}$ , this algorithm outputs a pair of master public key MPK and master secret key MSK as well as a set of secret keys  $\{\mathsf{SK}_i\}_{i \in [n-1]}$ .
- $\mathsf{Enc}(\mathsf{MPK},\mathsf{lab},x_0,\mu) \to \mathsf{CT}_{\mathsf{lab}}$ : On input the master public key MPK, a label  $\mathsf{lab} \in \mathcal{L}_{\lambda}$ , an attribute  $x_0 \in \mathcal{X}_{\lambda}$  and a message  $\mu \in \mathcal{M}_{\lambda}$ , this algorithm outputs a ciphertext  $\mathsf{CT}_{\mathsf{lab}}$ .

In case of an MC-ABE scheme, we assume that  $CT_{lab}$  implicitly includes  $x_0$ .

 $\begin{aligned} \mathsf{AKeyGen}(\mathsf{SK}_i,\mathsf{lab},x_i) \to \mathsf{DK}_{\mathsf{lab},i}: \text{ On input a secret key } \mathsf{SK}_i \text{ for some } i \in [n-1], \text{ a label} \\ \mathsf{lab} \in \mathcal{L}_\lambda \text{ and an attribute } x_i \in \mathcal{X}_\lambda, \text{ this algorithm outputs a decryption key } \mathsf{DK}_{\mathsf{lab},i}. \end{aligned}$ 

In case of an MC-ABE scheme, we assume that  $\mathsf{DK}_{\mathsf{lab},i}$  implicitly includes  $x_i$ .

 $\mathsf{PKeyGen}(\mathsf{MSK}, f) \to \mathsf{DK}_f$ : On input the master secret key  $\mathsf{MSK}$  and a policy  $f \in \mathcal{F}_{\lambda}$ , this algorithm outputs a decryption key  $\mathsf{DK}_f$ .

We assume that  $\mathsf{DK}_f$  implicitly includes a description of f.

 $\mathsf{Dec}(\mathsf{DK}_f, \{\mathsf{DK}_{\mathsf{lab},i}\}_{i\in[n-1]}, \mathsf{CT}_{\mathsf{lab}}) \to \mu' \lor \bot$ : On input a decryption key  $\mathsf{DK}_f$  for a policy  $f \in \mathcal{F}_{\lambda}$ , a set of attribute decryption keys  $\{\mathsf{DK}_{\mathsf{lab},i}\}_{i\in[n-1]}$  generated with respect to some label  $\mathsf{lab} \in \mathcal{L}_{\lambda}$  and a ciphertext  $\mathsf{CT}_{\mathsf{lab}}$  created with respect to the same label  $\mathsf{lab}$ , this algorithm outputs an element  $\mu' \in \mathcal{M}_{\lambda}$  or  $\bot$ .

Below, we discuss security in the public-key and secret-key setting. In the secret-key setting, we slightly change the syntax, as we find it more intuitive to let the encryption algorithm take a secret key  $SK_0$  instead of a master public key MPK if this key is not given to the adversary.

Remark 1 (Comparison With the Syntax of [AYY22, FFMV23, ARYY23]). Recent papers [AYY22, FFMV23, ARYY23] consider a varying syntax for MI-ABE and MI-PE. Our definition follows [ARYY23] as their syntax is the most natural one for our MC-ABE constructions. In fact, when removing the MCFE-related labels from our definition and replacing all keys with a unique master secret key, we almost obtain their definition except for one minor modification: their scheme defines n - 1 potentially different algorithms {KeyGen<sub>i</sub>}<sub>i\in[n-1]</sub> for the attribute key generation whereas we consider only one algorithm AKeyGen. In our constructions, the attribute key generation works in exactly the same way for all clients. Therefore, this modification does not have any impact on the semantics.

**Correctness.** An MC-ABE (resp. MC-PE) is *correct* if for every  $\lambda, n \in \mathbb{N}$ , label  $\mathsf{lab} \in \mathcal{L}_{\lambda}$ , message  $\mu \in \mathcal{M}_{\lambda}$ , policy  $f \in \mathcal{F}_{\lambda}$  and attributes  $x_0, \ldots, x_{n-1} \in \mathcal{X}_{\lambda}$  such that

$\begin{array}{l} \underbrace{Initialize(1^{\lambda}):}{\mathcal{C},\mathcal{Q}_{enc},\mathcal{Q}_{akey}},\mathcal{Q}_{pkey} \leftarrow \varnothing\\ (MPK,MSK,\{SK_i\}_{i \in [n-1]}) \leftarrow Setup(1^{\lambda})\\ \mathrm{Return} \ MPK \end{array}$	$\begin{array}{l} & \mathcal{O}AKeyGen(i,lab,x_i):\\ & \overline{\mathcal{Q}_{akey}} \leftarrow \mathcal{Q}_{akey} \cup \{(i,lab,x_i)\}\\ & \text{Return }DK_i \leftarrow AKeyGen(SK_i,lab,x_i) \end{array}$
$\frac{\mathcal{O}Corrupt(i) \text{ for } i \in [n-1]:}{\mathcal{C} \leftarrow \mathcal{C} \cup \{i\}; \text{ return } SK_i}$	$ \begin{array}{l} \displaystyle \frac{\mathcal{O}PKeyGen(f):}{\mathcal{Q}_{pkey} \leftarrow \mathcal{Q}_{pkey} \cup \{f\} \\ \mathrm{Return} \ DK_f \leftarrow PKeyGen(MSK, f) \end{array} $
$ \begin{array}{l} \displaystyle \frac{\mathcal{O}Enc(lab, x_0, \mu^0, \mu^1):}{\mathcal{Q}_{enc} \leftarrow \mathcal{Q}_{enc} \cup \{(lab, x_0, \mu^0, \mu^1)\}} \\ \mathrm{Return} \ CT \leftarrow Enc(MPK, lab, x_0, \mu^b) \end{array} $	$\frac{Finalize(b'):}{\mathrm{If} \; adm(\mathcal{A}) = 1,  \mathrm{return} \; \beta \leftarrow (b' \stackrel{?}{=} b)}{\mathrm{Else, return a random bit} \; \beta \stackrel{\$}{\leftarrow} \{0, 1\}}$

Figure 3: Security game  $\mathbf{Exp}_{\mathsf{aFE},\mathcal{A}}^{\mathsf{mc-abe-}b}(1^{\lambda})$  for Definition 11

 $f(x_0,\ldots,x_{n-1})=1$ , it holds that

$$\Pr \begin{bmatrix} \mu' = \mu & (\mathsf{MPK}, \mathsf{MSK}, \{\mathsf{SK}_i\}_{i \in [n-1]}) \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{CT}_{\mathsf{lab}} \leftarrow \mathsf{Enc}(\mathsf{MPK}, \mathsf{lab}, x_0, \mu) \\ \forall i \in [n-1] \colon \mathsf{DK}_{\mathsf{lab}, i} \leftarrow \mathsf{AKeyGen}(\mathsf{SK}_i, \mathsf{lab}, x_i) \\ \mathsf{DK}_f \leftarrow \mathsf{PKeyGen}(\mathsf{MSK}, f) \\ \mu' \coloneqq \mathsf{Dec}(\mathsf{DK}_f, \{\mathsf{DK}_{\mathsf{lab}, i}\}_{i \in [n-1]}, \mathsf{CT}_{\mathsf{lab}}) \end{bmatrix} = 1$$

**Security.** We define security for MC-ABE in the public-key setting as well as security for MC-PE in the secret-key setting.

**Definition 11** (Public-Key Security for MC-ABE). Let  $xxx \in \{sel, adap\}$  and  $yyy \in \{norep, rep\}$ . For an MC-ABE scheme aFE and a PPT adversary  $\mathcal{A}$ , we define the experiment  $\mathbf{Exp}_{\mathsf{aFE},\mathcal{A}}^{\mathsf{mc-abe},b}(1^{\lambda})$  as shown in Figure 3. The oracles  $\mathcal{O}\mathsf{Corrupt}$ ,  $\mathcal{O}\mathsf{Enc}$ ,  $\mathcal{O}\mathsf{A}\mathsf{Key}\mathsf{Gen}$  and  $\mathcal{O}\mathsf{P}\mathsf{Key}\mathsf{Gen}$  can be called in any order and any polynomial number of times, except for  $\mathcal{O}\mathsf{Enc}$  which can be called only once. Let  $(\mathsf{lab}, x_0, \mu^0, \mu^1)$  denote the single query to  $\mathcal{O}\mathsf{Enc}$ . The adversary  $\mathcal{A}$  is admissible, denoted by  $\mathsf{adm}(\mathcal{A}) = 1$ , if it satisfies the following conditions:

- 1. For all  $f \in \mathcal{Q}_{pkey}$  and  $x_1, \ldots, x_{n-1} \in \mathcal{X}_{\lambda}$  such that  $(i, \mathsf{lab}, x_i) \in \mathcal{Q}_{\mathsf{akey}}$  for all  $i \in [n-1] \setminus \mathcal{C}$ , it holds  $f(x_0, \ldots, x_{n-1}) = 0$ .
- 2. If xxx = sel, then the adversary submits the queries to  $\mathcal{O}Corrupt$ ,  $\mathcal{O}Enc$  and  $\mathcal{O}AKeyGen$  upfront in one shot.
- If yyy = norep, then for each i ∈ [n − 1] and lab ∈ L the adversary submits at most one query of the form OAKeyGen(i, lab, \*), i.e. we have |{x<sub>i</sub> ∈ {0,1}<sup>k</sup> : (i, lab, x<sub>i</sub>) ∈ Q<sub>akey</sub>}| ≤ 1.

Otherwise, we say that  $\mathcal{A}$  is not admissible and write  $\mathsf{adm}(\mathcal{A}) = 0$ . We call  $\mathsf{aFE} \mathsf{xxx-yyy-}$ secure if  $\mathbf{Exp}_{\mathsf{aFE},\mathcal{A}}^{\mathsf{mc-abe-0}}(1^{\lambda}) \approx_c \mathbf{Exp}_{\mathsf{aFE},\mathcal{A}}^{\mathsf{mc-abe-1}}(1^{\lambda})$ .

**Definition 12** (Secret-Key Security for MC-PE). Let  $xxx \in \{sel, adap\}$  and  $yyy \in \{norep, rep\}$ . For an MC-PE scheme pFE and a PPT adversary  $\mathcal{A}$ , we define the experiment  $\mathbf{Exp}_{\mathsf{pFE},\mathcal{A}}^{\mathsf{mc-pe-b}}(1^{\lambda})$  as shown in Figure 4. The oracles  $\mathcal{O}$ Corrupt,  $\mathcal{O}$ Enc,  $\mathcal{O}$ AKeyGen and  $\mathcal{O}$ PKeyGen can be called in any order and any polynomial number of times. Let  $\mathsf{lab} \in \mathcal{L}$ .

$\begin{split} & \frac{Initialize(1^{\lambda}):}{\mathcal{C},\mathcal{Q}_{enc},\mathcal{Q}_{akey}}, \mathcal{Q}_{pkey} \leftarrow \varnothing \\ & (\{SK_i\}_{i \in [0;n-1]},MSK) \leftarrow Setup(1^{\lambda}) \end{split}$	$ \begin{array}{l} \displaystyle \frac{\mathcal{O}AKeyGen(i,lab,x_i^0,x_i^1):}{\mathcal{Q}_{akey} \leftarrow \mathcal{Q}_{akey} \cup \{(i,lab,x_i^0,x_i^1)\} \\ \mathrm{Return} \ DK_i \leftarrow AKeyGen(SK_i,lab,x_i^b) \end{array} $
$ \frac{\mathcal{O}Corrupt(i) \text{ for } i \in [0; n-1]]:}{\mathcal{C} \leftarrow \mathcal{C} \cup \{i\}; \text{ return } SK_i} $ $ \mathcal{O}Enc(lab, x_0^0, x_0^1, \mu^0, \mu^1): $	$\begin{array}{l} \displaystyle \frac{\mathcal{O}PKeyGen(f):}{\mathcal{Q}_pkey \leftarrow \mathcal{Q}_pkey \cup \{f\}} \\ \mathrm{Return} \ DK_f \leftarrow PKeyGen(MSK, f) \end{array}$
$ \frac{\overline{\mathcal{Q}_{enc}} \leftarrow \mathcal{Q}_{enc} \cup \{(lab, x_0^0, x_0^1, \mu^0, \mu^1)\}}{\operatorname{Return} CT \leftarrow Enc(SK_1, lab, x_0^b, \mu^b)} $	$\frac{\text{Finalize}(b'):}{\text{If } adm(\mathcal{A}) = 1, \text{ return } \beta \leftarrow (b' \stackrel{?}{=} b)} \\ \text{Else, return a random bit } \beta \stackrel{\$}{\leftarrow} \{0, 1\}$

Figure 4: Security game  $\mathbf{Exp}_{\mathsf{pFE},\mathcal{A}}^{\mathsf{mc-pe}-b}(1^{\lambda})$  for Definition 12

We define  $Q'_{0,\mathsf{lab}} = \{(x_0^0, x_0^1, \mu^0, \mu^1) : (\mathsf{lab}, x_0^0, x_0^1, \mu^0, \mu^1) \in \mathcal{Q}_{\mathsf{enc}}\}$  and  $Q'_{i,\mathsf{lab}} = \{(x_i^0, x_i^1) : (i, \mathsf{lab}, x_i^0, x_i^1) \in \mathcal{Q}_{\mathsf{akey}}\}$  as well as

$$\begin{aligned} \mathcal{Q}_{0,\mathsf{lab}} &= \begin{cases} \mathcal{Q}'_{0,\mathsf{lab}} & \text{if } 0 \in [0; n-1] \setminus \mathcal{C} \\ \mathcal{Q}'_{0,\mathsf{lab}} \cup \{(x_0, x_0, \mu, \mu) : x_0 \in \mathcal{X}, \mu \in \mathcal{M}\} & \text{if } 0 \in \mathcal{C} \end{cases} \\ \mathcal{Q}_{i,\mathsf{lab}} &= \begin{cases} \mathcal{Q}'_{i,\mathsf{lab}} & \text{if } i \in [0; n-1] \setminus \mathcal{C} \\ \mathcal{Q}'_{i,\mathsf{lab}} \cup \{(x_i, x_i) : x_i \in \mathcal{X}\} & \text{if } i \in \mathcal{C} \end{cases} \end{aligned}$$

for all  $i \in [n-1]$ . The adversary  $\mathcal{A}$  is *admissible*, denoted by  $\mathsf{adm}(\mathcal{A}) = 1$ , if it satisfies the following conditions:

- 1. For all lab  $\in \mathcal{L}$ ,  $(x_0^0, x_0^1, \mu^0, \mu^1) \in \mathcal{Q}_{0,\mathsf{lab}}$ ,  $(x_1^0, x_1^1) \in \mathcal{Q}_{1,\mathsf{lab}}, (x_2^0, x_2^1) \in \mathcal{Q}_{2,\mathsf{lab}}, \dots, (x_{n-1}^0, x_{n-1}^1) \in \mathcal{Q}_{n-1,\mathsf{lab}}$  and policies  $f \in \mathcal{Q}_{\mathsf{pkey}}$ , it holds  $f(x_0^0, \dots, x_{n-1}^0) = f(x_0^1, \dots, x_{n-1}^1) = 0$  or  $(x_0^0, \dots, x_{n-1}^0, \mu^0) = (x_0^1, \dots, x_{n-1}^1, \mu^1).$
- 2. If xxx = sel, then the adversary cannot call  $\mathcal{O}Corrupt$ ,  $\mathcal{O}Enc$  and  $\mathcal{O}AKeyGen$  anymore after submitting the first query to  $\mathcal{O}PKeyGen$ .
- 3. If yyy = norep, then for each  $i \in [n-1]$  and  $\mathsf{lab} \in \mathcal{L}$  the adversary submits at most one query of the form  $\mathcal{O}\mathsf{AKeyGen}(i,\mathsf{lab},\star,\star)$ , *i.e.* we have  $|\{x_i \in \{0,1\}^k : (i,\mathsf{lab},x_i) \in \mathcal{Q}_{\mathsf{akey}}\}| \leq 1$ .

Otherwise, we say that  $\mathcal{A}$  is not admissible and write  $\mathsf{adm}(\mathcal{A}) = 0$ . We call pFE xxx-yyysecure if  $\mathbf{Exp}_{\mathsf{pFE},\mathcal{A}}^{\mathsf{mc-pe-0}}(1^{\lambda}) \approx_c \mathbf{Exp}_{\mathsf{pFE},\mathcal{A}}^{\mathsf{mc-pe-1}}(1^{\lambda})$ .

# 5 Construction of MC-ABE

In this section, we present our constructions for MC-ABE. For this, we need the following two definitions.

**Complete Policies.** Let  $k, n \in \mathbb{N}$ . We define the function  $c: \{0, 1\}^{[n-1]\times\{0\}} \to \{0, 1\}$  by  $c((x_{i,0})_{i\in[n-1]}) = \bigwedge_{i\in[n-1]} x_{i,0}$ . Given a policy  $f: \{0, 1\}^{[0;n-1]\times[k]} \to \{0, 1\}$ , we write  $(c \wedge f)$  for the *complete* policy

$$(c \wedge f) \colon \{0,1\}^{[0;n-1] \times [0;k]} \to \{0,1\}$$
  
$$(x_{i,j})_{i,j} \mapsto c((x_{i,0})_{i \in [n-1]}) \wedge f((x_{i,j})_{(i,j) \in [0;n-1] \times [k]}) .$$

We note that c is monotone (so having certificates of  $\{\neg x_{i,0}\}_{i\in[n-1]}$  is never relevant) and that  $(c \land f)$  does not depend on the attribute  $x_{0,0}$  at all. Taking these facts into account would allow for a slight improvement in the efficiency of Constructions 2 and 5 below, as some of the generated iFE instances are never used. We disregard this fact for the sake of a clearer write-up.

**Transformation to Monotone Policies.** For  $(x_{i,j})_{i,j} \in \{0,1\}^{[0;n-1]\times[0;k]}$ , we define  $(\overline{x}_{i,j}^{\beta})_{i,j}^{\beta} \in \{0,1\}^{[0;n-1]\times[0;k]\times\{0,1\}}$  by  $\overline{x}_{i,j}^{1} = x_{i,j}$  and  $\overline{x}_{i,j}^{0} = 1 - x_{i,j}$ . Given a policy  $f: \{0,1\}^{[0;n-1]\times[0;k]} \to \{0,1\}$  computable by an NC<sup>1</sup> circuit, we construct the corresponding monotone policy  $\overline{f}: \{0,1\}^{[0;n-1]\times[0;k]\times\{0,1\}} \to \{0,1\}$  as follows: first, we view f as a Boolean formula consisting of (fan-in 1)  $\neg$  gates and (fan-in 2)  $\land$  and  $\lor$  gates. Then, using De Morgan laws, we push the  $\neg$  gates to the leaves such that all internal nodes consist only of  $\land$  and  $\lor$  gates, while leaves are labeled by either attributes or their negations. Finally, for each  $(i,j) \in [0;n-1] \times [0;k]$  we identify the attribute  $x_{i,j} \in \{0,1\}$  with the attribute  $\overline{x}_{i,j}^{1}$  and the negation of  $x_{i,j}$  with  $\overline{x}_{i,j}^{0}$ . The resulting formula  $\overline{f}$  is monotone and equivalent to f in the sense that it satisfies  $\overline{f}((\overline{x}_{i,j}^{\beta})_{i,j}^{\beta}) = f((x_{i,j})_{i,j})$  for all inputs  $(x_{i,j})_{i,j} \in \{0,1\}^{[0;n-1]\times[0;k]}$ .

We combine the transformation to monotone policies with complete policies. Given an NC<sup>1</sup> policy  $f: \{0,1\}^{[0;n-1]\times[k]} \to \{0,1\}$ , we denote by  $\overline{(c \wedge f)}: \{0,1\}^{[0;n-1]\times[0;k]\times\{0,1\}} \to \{0,1\}$  the monotone policy obtained by applying the above transformation to the complete policy  $(c \wedge f): \{0,1\}^{[0;n-1]\times[0;k]} \to \{0,1\}$ .

### 5.1 MC-ABE Without Repetitions

We start with the description of our MC-ABE for LSS without repetitions. As explained in Section 2, this construction can be lifted to MC-ABE for NC<sup>1</sup> policies when not considering corruptions.

**Construction 2** (MC-ABE for LSS Without Repetitions). Let  $k, n = \text{poly}(\lambda)$ . For a vector  $\mathbf{x} \in \{0,1\}^k$ , we denote  $\Pi_{\mathbf{x}} = \{j \in [k] : \mathbf{x}[j] = 1\}$ . Our construction uses the following ingredients:

- A slotted IPFE scheme iFE = (iSetup, iEnc, iKeyGen, iDec) based on a pairing group  $\mathbb{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, g_1, g_2, g_t, e, p).$
- An identity-based encryption scheme  $\mathsf{idFE} = (\mathsf{idSetup}, \mathsf{idEnc}, \mathsf{idKeyGen}, \mathsf{idDec})$  with identity space  $\mathcal{I} = \mathcal{L} \times [0; k]$  for  $\mathcal{L} = \{0, 1\}^{\operatorname{poly}(\lambda)}$  and message space being the ciphertext space of iFE.

The MC-ABE scheme aFE for *n* clients and LSS policies with label space  $\mathcal{L}$ , message space  $\mathcal{M} = \mathbb{G}_t$  and attribute universe  $\mathcal{X} = \{0, 1\}^k$  works as follows:

 $\mathsf{Setup}(1^{\lambda})$  takes as input the security parameter  $1^{\lambda}$  and generates

$$\begin{split} (\mathsf{iMPK}_0, \mathsf{iMSK}_0) &\leftarrow \mathsf{iSetup}(1^{\lambda}, \{1, 2\}, \{3\}) \\ \big\{ (\mathsf{iMPK}_{i,j}, \mathsf{iMSK}_{i,j}) &\leftarrow \mathsf{iSetup}(1^{\lambda}, \{1\}, \{2\}) \big\}_{(i,j) \in ([0;n-1] \times [0;k])} \\ \big\{ (\mathsf{idMPK}_i, \mathsf{idMSK}_i) &\leftarrow \mathsf{idSetup}(1^{\lambda}) \big\}_{i \in [n-1]} . \end{split}$$

Then it outputs (MPK, MSK,  $\{SK_i\}_{i \in [n-1]}$ ) as follows:

$$\begin{split} \mathsf{MPK} &= \left(\mathsf{iMPK}_{0}, \{\mathsf{iMPK}_{i,j}\}_{(i,j)\in([0;n-1]\times[0;k])}, \{\mathsf{idMPK}_{i}\}_{i\in[n-1]}\right) \\ \mathsf{MSK} &= \left(\mathsf{iMSK}_{0}, \{\mathsf{iMSK}_{i,j}\}_{(i,j)\in([0;n-1]\times[0;k])}\right) \\ \{\mathsf{SK}_{i} &= \mathsf{idMSK}_{i}\}_{i\in[n-1]} \end{split}$$

We implicitly parse these keys in the algorithms below.

Enc(MPK, lab,  $\mathbf{x}_0, \llbracket \mu \rrbracket_t$ ) takes as input MPK, a label lab  $\in \mathcal{L}$ , an attribute  $\mathbf{x}_0 \in \{0, 1\}^k$ and a message  $\llbracket \mu \rrbracket_t \in \mathbb{G}_t$ . The algorithm samples random elements  $\llbracket r \rrbracket_1, \llbracket \sigma \rrbracket_1 \stackrel{\scriptscriptstyle{\otimes}}{\leftarrow} \mathbb{G}_1$ , computes  $\llbracket d \rrbracket_t = \llbracket \sigma + \mu \rrbracket_t$  and generates

$$\begin{split} &\mathsf{iCT}_0 \leftarrow \mathsf{iSlotEnc}(\mathsf{iMPK}_0, \llbracket (r, \sigma) \rrbracket_1) \\ \big\{ \mathsf{iCT}_{i,j} \leftarrow \mathsf{iSlotEnc}(\mathsf{iMPK}_{i,j}, \llbracket r \rrbracket_1) \big\}_{(i,j) \in \{0\} \times \Pi_{\mathbf{x}_0} \cup [n-1] \times [0;k]} \\ \big\{ \mathsf{idCT}_{i,j} \leftarrow \mathsf{idEnc}(\mathsf{idMPK}_i, (\mathsf{lab}, j), \mathsf{iCT}_{i,j}) \big\}_{(i,j) \in [n-1] \times [0;k]} \end{split}.$$

Then it outputs  $\mathsf{CT}_{\mathsf{lab}} = (\llbracket d \rrbracket_{\mathsf{t}}, \mathsf{iCT}_0, \{\mathsf{iCT}_{0,j}\}_{j \in \Pi_{\mathbf{x}_0}}, \{\mathsf{idCT}_{i,j}\}_{(i,j) \in [n-1] \times [0;k]}).$ 

 $\mathsf{AKeyGen}(\mathsf{SK}_i,\mathsf{lab},\mathbf{x}_i)$  takes as input  $\mathsf{SK}_i$  for some  $i \in [n-1]$ , a label  $\mathsf{lab} \in \mathcal{L}$  and an attribute  $\mathbf{x}_i \in \{0,1\}^k$ . Then the algorithm outputs the decryption key  $\mathsf{DK}_{\mathsf{lab},i} = \{\mathsf{id}\mathsf{DK}_{i,j}\}_{j\in\{0\}\cup\Pi_{\mathbf{x}_i}}$  computed as follows:

 ${idDK_{i,j} \leftarrow idKeyGen(idMSK_i, (lab, j))}_{j \in {0} \cup \Pi_{\mathbf{x}_i}}$ .

PKeyGen(MSK, f) takes as input MSK and a monotone policy f. Let  $(c \wedge f) = (\mathbf{M} \in \mathbb{Z}_p^{m \times \ell}, \rho \colon [\ell] \to ([0; n-1] \times [0; k]))$ . The algorithm samples  $s \stackrel{\$}{\leftarrow} \mathbb{Z}_p$  and  $\mathbf{u} \stackrel{\$}{\leftarrow} \mathbb{Z}_p^{m-1}$ , computes  $\mathbf{s} = (s, \mathbf{u}^\top) \cdot \mathbf{M}$  and generates

$$\begin{split} & \mathsf{i}\mathsf{D}\mathsf{K}_0 \leftarrow \mathsf{i}\mathsf{KeyGen}(\mathsf{i}\mathsf{M}\mathsf{S}\mathsf{K}_0, [\![(s, 1, 0)]\!]_2) \\ & \left\{\mathsf{i}\mathsf{D}\mathsf{K}_j \leftarrow \mathsf{i}\mathsf{KeyGen}(\mathsf{i}\mathsf{M}\mathsf{S}\mathsf{K}_{\rho(j)}, [\![(\mathbf{s}[j], 0)]\!]_2)\right\}_{j \in [\ell]} \ . \end{split}$$

Finally, it outputs  $\mathsf{DK}_f = {\mathsf{i}\mathsf{DK}_j}_{j \in [0;\ell]}$ .

 $\begin{aligned} \mathsf{Dec}(\mathsf{DK}_f, \{\mathsf{DK}_{\mathsf{lab},i}\}_{i\in[n-1]}, \mathsf{CT}_{\mathsf{lab}}) \text{ takes as input a decryption key } \mathsf{DK}_f \text{ for a policy } f, \text{ a set} \\ & \text{ of decryption keys } \{\mathsf{DK}_{\mathsf{lab},i}\}_{i\in[n-1]} \text{ generated with respect to attributes } \mathbf{x}_1, \ldots, \mathbf{x}_{n-1} \in \\ & \{0,1\}^k \text{ and a label } \mathsf{lab} \in \mathcal{L}, \text{ and a ciphertext } \mathsf{CT}_{\mathsf{lab}} \text{ created with respect to an attribute } \mathbf{x}_0 \in \{0,1\}^k \text{ and the same label } \mathsf{lab}. \text{ We parse } \mathsf{DK}_f, \{\mathsf{DK}_{\mathsf{lab},i}\}_{i\in[n-1]} \text{ and } \mathsf{CT}_{\mathsf{lab}} \\ & \text{ as in the algorithms above. Let } (c \wedge f) = (\mathbf{M} \in \mathbb{Z}_p^{m \times \ell}, \rho \colon [\ell] \to ([0; n-1] \times [0; k])). \\ & \text{ If } X = \bigcup_{i \in [0; n-1]} (\{i\} \times (\{0\} \cup \Pi_{\mathbf{x}_i})) \cap \rho([\ell]) \text{ does not satisfy the policy } (c \wedge f), \text{ then the algorithm outputs } \bot. \text{ Otherwise, it decrypts} \end{aligned}$ 

$$\left\{\mathsf{iCT}_{i,j} \leftarrow \mathsf{idDec}(\mathsf{idDK}_{i,j},\mathsf{idCT}_{i,j})\right\}_{(i,j)\in X\cap([n-1]\times[0;k])},$$

and finds coefficients  $\omega_1, \ldots, \omega_\ell \in \mathbb{Z}_p$  such that  $\sum_{j \in [\ell]} \omega_j \mathbf{M}[j] = \mathbf{e}_1$  and  $\omega_j = 0$  for all  $j \notin \rho^{-1}(X)$ . Finally, the algorithm computes

$$\begin{split} \llbracket d_0 \rrbracket_{\mathsf{t}} &\leftarrow \mathsf{i}\mathsf{Dec}(\mathsf{i}\mathsf{D}\mathsf{K}_0,\mathsf{i}\mathsf{C}\mathsf{T}_0) \\ \left\{ \llbracket d_j \rrbracket_{\mathsf{t}} &\leftarrow \mathsf{i}\mathsf{Dec}(\mathsf{i}\mathsf{D}\mathsf{K}_j,\mathsf{i}\mathsf{C}\mathsf{T}_{\rho(j)}) \right\}_{j \in \rho^{-1}(X)} \end{split}$$

and outputs  $[\![\mu']\!]_{t} = [\![d]\!]_{t} - [\![d_{0}]\!]_{t} + \sum_{j \in \rho^{-1}(X)} \omega_{j} [\![d_{j}]\!]_{t}.$ 

**Correctness.** By the correctness of idFE and iFE, we have  $d_0 = rs + \sigma$  and, for all  $j \in \rho^{-1}(X)$ ,  $d_j = r\mathbf{s}[j]$ . Note that  $\bigcup_{i \in [n-1]} \{(i,0)\} \subseteq X$ , so X always satisfies c. If additionally  $f(\mathbf{x}_0, \ldots, \mathbf{x}_{n-1}) = 1$ , then X is authorized with respect to  $c \wedge f$ . In this case, we conclude from the correctness of the LSS scheme that

$$\mu' = d - d_0 + \sum_{j \in \rho^{-1}(X)} \omega_j d_j = (\sigma + \mu) - (rs + \sigma) + r \cdot \underbrace{\sum_{j \in \rho^{-1}(X)} \omega_j \mathbf{s}[j]}_{= s} = \mu$$

**Security.** We prove selective security without repetitions.

**Proposition 3.** If the DDH assumption holds in  $\mathbb{G}_2$ , iFE is slot-mode correct and functionhiding and idFE is secure, then Construction 2 is sel-norep-secure.

When instantiating iFE with the slotted ABE from [LL20a] and idFE with the IBE from  $[\text{CLL}^+13, \text{JR}17]^7$ , we obtain the following theorem.

Theorem 4. Assuming SXDH on pairings, there exist

- a sel-norep-secure MC-ABE for LSS, and
- a sel-norep-secure MC-ABE for NC<sup>1</sup> without corruptions.

*Proof* (Proposition 3). We consider a sequence  $G_0^0, \ldots, G_6^0, G_6^1, \ldots, G_0^1$  of hybrid games, where  $G_0^b = \mathbf{Exp}_{\mathsf{aFE},\mathcal{A}}^{\mathsf{mc-abe-}b}(1^{\lambda})$ . We argue  $G_6^0 \equiv G_6^1$  and  $G_{\nu-1}^b \approx_c G_{\nu}^b$  for  $\nu \in [6]$  and  $b \in \{0,1\}$ . Then the claim follows via a hybrid argument. Modifications between consecutive games are highlighted using boxes.

**Game**  $G_0^b$ : This is  $\operatorname{Exp}_{\mathsf{aFE},\mathcal{A}}^{\mathsf{mc-abe}-b}(1^{\lambda})$ . In particular, for the reply to the (single) encryption query  $\mathcal{O}\mathsf{Enc}(\mathsf{lab}, \mathbf{x}_0, \llbracket \mu^0 \rrbracket_t, \llbracket \mu^1 \rrbracket_t)$ , the challenger samples random elements  $\llbracket r \rrbracket_1, \llbracket \sigma \rrbracket_1 \stackrel{\$}{\leftarrow} \mathbb{G}_1$ , computes  $\llbracket d \rrbracket_t \coloneqq \llbracket \sigma + \mu^b \rrbracket_t$ , generates

$$\begin{split} &\mathsf{iCT}_0 \leftarrow \mathsf{iSlotEnc}(\mathsf{iMPK}_0, \llbracket (r, \sigma) \rrbracket_1) \\ & \left\{ \mathsf{iCT}_{i,j} \leftarrow \mathsf{iSlotEnc}(\mathsf{iMPK}_{i,j}, \llbracket r \rrbracket_1) \right\}_{(i,j) \in \{0\} \times \Pi_{\mathbf{x}_0} \cup [n-1] \times [0;k]} \\ & \left\{ \mathsf{idCT}_{i,j} \leftarrow \mathsf{idEnc}(\mathsf{idMPK}_i, (\mathsf{lab}, j), \mathsf{iCT}_{i,j}) \right\}_{(i,j) \in [n-1] \times [0;k]} , \end{split}$$

and sends  $\mathsf{CT}_{\mathsf{lab}} = (\llbracket d \rrbracket_{\mathsf{t}}, \mathsf{iCT}_0, \{\mathsf{iCT}_{0,j}\}_{j \in \Pi_{\mathbf{x}_0}}, \{\mathsf{idCT}_{i,j}\}_{(i,j) \in [n-1] \times [0;k]})$  to  $\mathcal{A}$ .

Game  $G_1^b$ : This game is the same as  $G_0^b$ , except that the challenger computes

$$\begin{split} &\mathsf{iCT}_0 \leftarrow \mathsf{iEnc}(\mathsf{iMSK}_0, [\![(r, \sigma, 0)]\!]_1) \\ &\left\{\mathsf{iCT}_{i,j} \leftarrow \mathsf{iEnc}(\mathsf{iMSK}_{i,j}, [\![(r, 0)]\!]_1)\right\}_{(i,j) \in \{0\} \times \Pi_{\mathbf{x}_0} \cup [n-1] \times [0;k]} \end{split}$$

We have  $\mathsf{G}_0^b \equiv \mathsf{G}_1^b$  which follows from the slot-mode correctness of iFE.

**Game**  $G_2^b$ : We define the set  $J^{\geq 1}$  containing the indices of all ciphertexts  $\mathsf{idCT}_{i,j}$  that the adversary is able to decrypt:

 $J^{\geq 1} = \left(\mathcal{C} \times [0;k]\right) \cup \{(i,j) : \exists (i,\mathsf{lab},\mathbf{x}_i) \in \mathcal{Q}_{\mathsf{akey}} \text{ s.t. } (\mathbf{x}_i[j] = 1 \lor j = 0)\} \ .$ 

This game is the same as  $G_1^b$ , except that the challenger now computes

$$\mathsf{iCT}_{i,j} \leftarrow \begin{cases} \mathsf{iEnc}(\mathsf{iMSK}_{i,j}, \llbracket (r,0) \rrbracket_1) & \text{ if } (i,j) \in J^{\geq 1} \\ \mathsf{iEnc}(\mathsf{iMSK}_{i,j}, \llbracket (0,0) \rrbracket_1) & \text{ if } (i,j) \notin J^{\geq 1} \end{cases}.$$

We have  $\mathsf{G}_1^b \approx_c \mathsf{G}_2^b$  from the security of idFE.

**Game**  $G_3^b$ : This game is the same as  $G_2^b$ , except that we hardwire the encryption query into the decryption keys. Initially, the challenger samples  $r, \sigma \in \mathbb{Z}_p$  and answers to the

<sup>&</sup>lt;sup>7</sup>The seminal adaptively secure group-based (H)IBE is [Wat09] but it relies on both DDH and D-Lin.

selective encryption query by sending  $CT_{lab} = (\llbracket d \rrbracket_t = \llbracket \sigma + \mu^b \rrbracket_t, iCT_0, \{iCT_{0,j}\}_{j \in \Pi_{\mathbf{x}_0}}, \{idCT_{i,j}\}_{(i,j) \in [n-1] \times [0;k]})$  computed as follows:

$$\begin{split} & \mathsf{iCT}_0 \leftarrow \mathsf{iEnc}(\mathsf{iMSK}_0, \llbracket (0,0,1) \rrbracket_1) \\ & \mathsf{iCT}_{i,j} \leftarrow \begin{cases} \mathsf{iEnc}(\mathsf{iMSK}_{i,j}, \llbracket (0,1) \rrbracket_1) & \mathrm{if} \ (i,j) \in J \\ & \mathsf{iEnc}(\mathsf{iMSK}_{i,j}, \llbracket (0,0) \rrbracket_1) & \mathrm{if} \ (i,j) \notin J \end{cases} \\ & \mathsf{idCT}_{i,j} \leftarrow \mathsf{idEnc}(\mathsf{idMPK}_i, (\mathsf{lab}, j), \mathsf{iCT}_{i,j}) \ , \end{split}$$

where  $J = J^{\geq 1} \cup \{(0, j) : \mathbf{x}_0[j] = 1\}$ . Upon receiving a query  $\mathcal{O}\mathsf{PKeyGen}(f)$  with  $(c \wedge f) = (\mathbf{M} \in \mathbb{Z}_p^{m \times \ell}, \rho : [\ell] \to ([0; n - 1] \times [0; k]))$ , it samples  $s \stackrel{\$}{\leftarrow} \mathbb{Z}_p, \mathbf{u} \stackrel{\$}{\leftarrow} \mathbb{Z}_p^{m-1}$  and computes  $\mathbf{s} = (s, \mathbf{u}^{\top}) \cdot \mathbf{M}$ . Then it sends  $\mathsf{DK}_f = \{\mathsf{i}\mathsf{DK}_\kappa\}_{\kappa \in [0; \ell]}$  to  $\mathcal{A}$  generated as follows:

$$\begin{split} &\mathsf{i}\mathsf{D}\mathsf{K}_{0} \leftarrow \mathsf{i}\mathsf{KeyGen}(\mathsf{i}\mathsf{M}\mathsf{S}\mathsf{K}_{0}, \llbracket(s, 1, \llbracket r \cdot s + \sigma)\rrbracket_{2}) \\ &\mathsf{i}\mathsf{D}\mathsf{K}_{\kappa} \leftarrow \begin{cases} \mathsf{i}\mathsf{KeyGen}(\mathsf{i}\mathsf{M}\mathsf{S}\mathsf{K}_{\rho(\kappa)}, \llbracket(\mathbf{s}[\kappa], \llbracket r \cdot \mathbf{s}[\kappa]])\rrbracket_{2}) & \text{if } \rho(\kappa) \in J \\ \mathsf{i}\mathsf{KeyGen}(\mathsf{i}\mathsf{M}\mathsf{S}\mathsf{K}_{\rho(\kappa)}, \llbracket(\mathbf{s}[\kappa], 0)\rrbracket_{2}) & \text{if } \rho(\kappa) \notin J \end{cases} \end{split}$$

As the public parts of the iFE keys and the inner products between vectors embedded in the iFE keys and ciphertexts do not change, it follows  $G_2^b \approx_c G_3^b$  from the functionhiding security of iFE.

**Game**  $G_4^b$ : This game is the same as  $G_3^b$  except that the challenger embeds a fresh secret sharing in each decryption key. Specifically, upon receiving a query  $\mathcal{O}\mathsf{PKeyGen}(f)$ with  $(c \wedge f) = (\mathbf{M}, \rho)$ , the challenger samples  $s, t \notin \mathbb{Z}_p$  and  $\mathbf{u}, \mathbf{v} \notin \mathbb{Z}_p^{m-1}$ , computes  $\mathbf{s} = (s, \mathbf{u}^\top) \cdot \mathbf{M}$  and  $\mathbf{t} = (t, \mathbf{v}^\top) \cdot \mathbf{M}$  and sends  $\mathsf{DK}_f = \{\mathsf{i}\mathsf{DK}_\kappa\}_{\kappa\in[0;\ell]}$  to  $\mathcal{A}$  generated as follows:

$$\begin{split} & \mathsf{i}\mathsf{D}\mathsf{K}_{0} \leftarrow \mathsf{i}\mathsf{KeyGen}(\mathsf{i}\mathsf{M}\mathsf{S}\mathsf{K}_{0}, [\![(s, 1, t + \sigma)]\!]_{2}) \\ & \mathsf{i}\mathsf{D}\mathsf{K}_{\kappa} \leftarrow \begin{cases} \mathsf{i}\mathsf{KeyGen}(\mathsf{i}\mathsf{M}\mathsf{S}\mathsf{K}_{\rho(\kappa)}, [\![(\mathbf{s}[\kappa], \mathbf{t}[\kappa]])]\!]_{2}) & \text{if } \rho(\kappa) \in J \\ & \mathsf{i}\mathsf{KeyGen}(\mathsf{i}\mathsf{M}\mathsf{S}\mathsf{K}_{\rho(\kappa)}, [\![(\mathbf{s}[\kappa], 0)]\!]_{2}) & \text{if } \rho(\kappa) \notin J \end{cases}. \end{split}$$

It is not hard to see that  $G_3^b \approx_c G_4^b$  under the DDH assumption in  $\mathbb{G}_2$ . (Note that we can exploit the random self-reducibility of the DDH problem here, so a single DDH instance suffices).

Let  $f \in \mathcal{Q}_{\mathsf{pkey}}$  and  $(c \wedge f) = (\mathbf{M} \in \mathbb{Z}_p^{m \times \ell}, \rho : [\ell] \to ([0; n-1] \times [0; k]))$ . We let  $M_f = \mathsf{span}\{\mathbf{M}[\kappa] : \kappa \in [\ell], \rho(\kappa) \in J\}$  be the vector space spanned by the columns of  $\mathbf{M}$  associated with attributes in J. Furthermore, let  $M_f^{\perp}$  denote the orthogonal complement of  $M_f$  and  $A_f^{\perp}$  the affine space of  $M_f^{\perp}$  containing all vectors whose first coordinate is 1. We argue that  $A_f^{\perp}$  is nonempty. To do so, we distinguish two cases.

- 1. If there exists  $i \in [n-1]$  such that  $i \notin C$  and  $\mathcal{Q}_{\mathsf{akey}}$  does not contain a tuple of the form  $(i, \mathsf{lab}, \star)$ , then  $J \cap ([n-1] \times \{0\})$  does not satisfy c (and thus  $(c \wedge f)$ ). Hence, we have  $\mathbf{e}_1 \notin M_f$  which implies that  $A_f^{\perp}$  is nonempty.
- 2. If there exists a tuple  $(i, \mathsf{lab}, \mathbf{x}_i) \in \mathcal{Q}_{\mathsf{akey}}$  for all  $i \in [n-1] \setminus \mathcal{C}$ , then it follows from the admissibility of  $\mathcal{A}$  that  $J \cap ([n-1] \times k)$  cannot satisfy f (and thus  $(c \wedge f)$ ), which implies that  $\mathbf{e}_1 \notin M_f$  and  $A_f^{\perp}$  is nonempty.
- **Game**  $G_5^b$ : This game is the same as  $G_4^b$ , except that we modify how the challenger generates the share vector **t** for the reply to a query OPKeyGen(f). Let  $(c \land f) =$

 $(\mathbf{M}, \rho)$ . In  $\mathsf{G}_4^b$ , the challenger samples  $t \stackrel{\$}{\leftarrow} \mathbb{Z}_p, \mathbf{v} \stackrel{\$}{\leftarrow} \mathbb{Z}_p^{m-1}$  and computes  $\mathbf{t} = (t, \mathbf{v}^\top) \cdot \mathbf{M}$ . In the current game, the challenger picks an additional vector  $(1, \overline{\mathbf{v}})^\top \in A_f^\perp$  and sets

$$\mathbf{t} = (t, \mathbf{v}^{\top}) \cdot \mathbf{M} + (0, t \overline{\mathbf{v}}^{\top}) \cdot \mathbf{M} \quad .$$

The distributions  $\{\mathbf{v} : \mathbf{v} \stackrel{s}{\leftarrow} \mathbb{Z}_p^{m-1}\}$  and  $\{\mathbf{v} + t\overline{\mathbf{v}} : \mathbf{v} \stackrel{s}{\leftarrow} \mathbb{Z}_p^{m-1}\}$  are identical for any  $\overline{\mathbf{v}} \in \mathbb{Z}_p^{m-1}$ . The first distribution corresponds to  $\mathsf{G}_4^b$  and the second one to  $\mathsf{G}_5^b$ (followed by the same post-processing), so we have  $\mathsf{G}_4^b \equiv \mathsf{G}_5^b$ .

**Game**  $G_6^b$ : This game is the same as  $G_5^b$  except that we change the behavior of the key generation oracle upon receiving a query  $\mathcal{O}\mathsf{PKeyGen}(f)$ . Let  $(c \land f) = (\mathbf{M}, \rho)$ . As in the previous hybrid, the challenger samples  $s, t \stackrel{\text{\tiny{\$}}}{=} \mathbb{Z}_p$  and  $\mathbf{u}, \mathbf{v} \stackrel{\text{\tiny{\$}}}{=} \mathbb{Z}_p^{m-1}$  and computes  $\mathbf{s} = (s, \mathbf{u}^{\top}) \cdot \mathbf{M}$ . In the current hybrid, it then generates  $\mathsf{DK}_f = \{\mathsf{i}\mathsf{DK}_\kappa\}_{\kappa\in[0,\ell]}$  as follows:

$$\begin{split} & \mathsf{i}\mathsf{D}\mathsf{K}_0 \leftarrow \mathsf{i}\mathsf{KeyGen}(\mathsf{i}\mathsf{M}\mathsf{S}\mathsf{K}_0, [\![(s, 1, t + \sigma)]\!]_2) \\ & \mathsf{i}\mathsf{D}\mathsf{K}_\kappa \leftarrow \begin{cases} \mathsf{i}\mathsf{KeyGen}(\mathsf{i}\mathsf{M}\mathsf{S}\mathsf{K}_{\rho(\kappa)}, [\![(\mathbf{s}[\kappa], (0, \mathbf{v}^\top) \cdot \mathbf{M}[\kappa]])]\!]_2) & \text{if } \rho(\kappa) \in J \\ & \mathsf{i}\mathsf{KeyGen}(\mathsf{i}\mathsf{M}\mathsf{S}\mathsf{K}_{\rho(\kappa)}, [\![(\mathbf{s}[\kappa], 0)]\!]_2) & \text{if } \rho(\kappa) \notin J \end{cases} \end{split}$$

whereas in  $G_5^b$ , the challenger additionally picks  $(1, \overline{\mathbf{v}})^\top \in A_f^\perp$  and computes

$$\mathsf{iDK}_{\kappa} \leftarrow \mathsf{iKeyGen}(\mathsf{iMSK}_{\rho(\kappa)}, [\![(\mathbf{s}[\kappa], (t, (\mathbf{v} + t\overline{\mathbf{v}})^{\top}) \cdot \mathbf{M}[\kappa])]\!]_2)$$

for all  $\kappa \in [\ell]$  such that  $\rho(\kappa) \in J$ . We can observe that

$$(t, (\mathbf{v} + t\bar{\mathbf{v}})^{\top}) \cdot \mathbf{M}[\kappa] = (0, \mathbf{v}^{\top}) \cdot \mathbf{M}[\kappa] + t(1, \bar{\mathbf{v}}^{\top}) \cdot \mathbf{M}[\kappa] = (0, \mathbf{v}^{\top}) \cdot \mathbf{M}[\kappa] \ ,$$

where the last equality follows from the fact that  $(1, \bar{\mathbf{v}}^{\top}) \in M_f^{\perp}$  and  $\mathbf{M}[\kappa] \in M_f$ . Thus, we have  $\mathsf{G}_5^b \equiv \mathsf{G}_6^b$ . We notice that in  $\mathsf{G}_6^b$  the random element  $\llbracket t \rrbracket_2 \stackrel{\text{\tiny{\$}}}{=} \mathbb{G}_2$  appears only in iDK<sub>0</sub>, so it perfectly masks  $\llbracket \sigma \rrbracket_2$ . From this, it follows in turn that  $\llbracket \sigma \rrbracket_t$ serves as a perfect mask for the challenge message  $\llbracket \mu^b \rrbracket_t$  in  $\llbracket d \rrbracket_t = \llbracket \sigma + \mu^b \rrbracket_t$ , and we conclude that  $\mathsf{G}_6^0 \equiv \mathsf{G}_6^1$ .

#### 5.2 MC-ABE With Repetitions

In this section, we present our construction of MC-ABE in the stronger security model with repetitions.

**Our Policy Classes.** We consider various policy classes  $\mathcal{F}$  containing policies of the form  $f: \{0,1\}^{[0;n-1]\times[k]} \to \{0,1\}$ .

• Small Parameters. Let k, n such that  $kn = O(\log \lambda)$ , *i.e.* the total length of the input is logarithmic in  $\lambda$ . We let  $\mathcal{F}^{\mathsf{log-att}}$  denote the class of all policies with input  $\{0,1\}^{[0;n-1]\times[k]}$  computable by an  $\mathsf{NC}^1$  circuit and, for  $\mathbf{x}_0, \ldots, \mathbf{x}_n \in \{0,1\}^k$ , we define the sets

$$\Pi_{\mathbf{x}_{0}}^{\mathsf{log-att}'} = \left\{ (\mathbf{y}_{0}', \dots, \mathbf{y}_{n-1}') : \mathbf{y}_{0}' = \mathbf{x}_{0} \land \mathbf{y}_{1}', \dots, \mathbf{y}_{n-1}' \in \{0, 1\}^{k} \right\}$$

and  $\Omega_{\mathbf{x}_i}^{\mathsf{log-att}'} = {\mathbf{x}_i}$  for  $i \in [n-1]$ .

•  $\mathsf{NC}^0$  Policies. Let  $k, n = \mathrm{poly}(\lambda)$  and d = O(1) be some fixed upper bound on the depth of the considered circuits. Then each policy depends on at most  $\tau = 2^d = O(1)$  out of the kn input bits. We denote by  $\mathcal{F}^{\mathsf{const-dep}}$  the set of all  $\mathsf{NC}^0$  policies with depth d and input  $\{0, 1\}^{[0;n-1]\times[k]}$ . For  $\mathbf{x}_0, \ldots, \mathbf{x}_n \in \{0, 1\}^k$ , we define the sets

$$\begin{split} \Pi_{\mathbf{x}_0}^{\mathsf{const-dep}'} &= \left\{ (\mathbf{y}_0', \dots, \mathbf{y}_{n-1}') : \frac{\mathbf{y}_0' = \mathbf{x}_0 \wedge \mathbf{y}_1', \dots, \mathbf{y}_{n-1}' \in \{0, 1, \bot\}^k}{\text{s.t. } \sum_{i \in [0; n-1]} \delta(\mathbf{y}_i') = \tau} \right\} \\ \Omega_{\mathbf{x}_i}^{\mathsf{const-dep}'} &= \left\{ \mathbf{z}_i' \in \{0, 1, \bot\}^k : (\forall j \in [k]. \ \mathbf{z}_i'[j] \in \{\mathbf{x}_i[j], \bot\}) \wedge (\delta(\mathbf{z}_i') \leq \tau) \right\} \end{split}$$

where  $\delta(\mathbf{y}) = |\{j \in [k] : \mathbf{y}[j] \in \{0, 1\}\}|$  denotes the number of coordinates being not equal to  $\perp$ .

• Threshold Policies. Let  $k, n = \text{poly}(\lambda)$ . Instead of a constant input locality, we may also consider policies, where every authorized set has a constant-size subset that is also authorized. This property is satisfied by *e.g.* threshold policies with a constant threshold  $\tau = O(1)$ . We denote by  $f^{t-\text{thr}}$  the threshold policy which allows the reconstruction of the secret from arbitrary t (out of the total of kn) shares. Then we define the policy class  $\mathcal{F}^{\leq \text{const-thr}} = \{f^{t-\text{thr}} : t \in [\tau]\}$  and, for  $\mathbf{x}_0, \ldots, \mathbf{x}_n \in \{0, 1\}^k$ , we set

$$\Pi_{\mathbf{x}_{0}}^{\leq \text{const-thr}'} = \left\{ (\mathbf{y}_{0}', \dots, \mathbf{y}_{n-1}') : \frac{\mathbf{y}_{0}' = \mathbf{x}_{0} \wedge \mathbf{y}_{1}', \dots, \mathbf{y}_{n-1}' \in \{1, \bot\}^{k} \\ \text{s.t.} \quad \sum_{i \in [0; n-1]} \delta(\mathbf{y}_{i}') = \tau \right\}$$
$$\Omega_{\mathbf{x}_{i}}^{\leq \text{const-thr}'} = \left\{ \mathbf{z}_{i}' \in \{1, \bot\}^{k} : (\forall j \in [k]. \ \mathbf{z}_{i}'[j] \in S_{\mathbf{x}_{i}[j]}) \wedge (\delta(\mathbf{z}_{i}') \leq \tau) \right\} ,$$

where  $S_0 = \{\bot\}$  and  $S_1 = \{1, \bot\}$ . Note that threshold policies are in particular monotone policies, which is why we can pick  $\mathbf{y}'_i, \mathbf{z}'_i \in \{1, \bot\}^k$  as opposed to  $\mathbf{y}'_i, \mathbf{z}'_i \in \{0, 1, \bot\}^k$ . This will improve the efficiency of the scheme as it reduces the size of the sets  $\prod_{\mathbf{x}_0}^{\leq \text{const-thr}'}$  and  $\Omega_{\mathbf{x}_i}^{\leq \text{const-thr}'}$ .

Conversely, we define the policy class  $\mathcal{F}^{\geq \text{const-thr}} = \{f^{t-\text{thr}} : t \in [kn - \tau; kn]\}$  and the sets

$$\Pi_{\mathbf{x}_0}^{\geq \text{const-thr}'} = \left\{ (\mathbf{y}_0', \dots, \mathbf{y}_{n-1}') : \begin{array}{l} \mathbf{y}_0' = \mathbf{x}_0 \land \mathbf{y}_1', \dots, \mathbf{y}_{n-1}' \in \{1, \bot\}^k \\ \text{s.t.} \quad \sum_{i \in [0; n-1]} \delta(\mathbf{y}_i') \ge kn - \tau \end{array} \right\}$$

and  $\Omega_{\mathbf{x}_i}^{\geq \text{const-thr}'} = {\mathbf{z}'_i}$  where  $\mathbf{z}'_i \in {\{1, \bot\}}^k$  is defined coordinate-wise as  $\mathbf{z}'_i[j] = s_{\mathbf{x}_i[j]}$  for all  $j \in [k]$  where  $s_0 = \bot$  and  $s_1 = 1$ .

As in Construction 2, we must protect against incomplete queries by considering the complete policy  $(c \wedge f)$  instead of f. For this, we define

$$\begin{split} \Pi_{\mathbf{x}_{0}}^{\mathsf{type}} = \left\{ (\mathbf{y}_{0}, \dots, \mathbf{y}_{n-1}) : \begin{array}{l} \mathbf{y}_{0} = (\bot, \mathbf{y}_{0}') \land \mathbf{y}_{1} = (1, \mathbf{y}_{1}') \land \dots \land \mathbf{y}_{n-1} = (1, \mathbf{y}_{n-1}') \\ \text{s.t.} \ (\mathbf{y}_{0}', \dots, \mathbf{y}_{n-1}') \in \Pi_{\mathbf{x}_{0}}^{\mathsf{type}'} \\ \Omega_{\mathbf{x}_{i}}^{\mathsf{type}} = \left\{ \mathbf{z}_{i} = (1, \mathbf{z}_{i}') : \mathbf{z}_{1}' \in \Omega_{\mathbf{x}_{i}}^{\mathsf{type}'} \right\} \ , \end{split}$$

where type  $\in$  {log-att, const-dep,  $\leq$  const-thr,  $\geq$  const-thr} and  $i \in [n-1]$ .

**Construction 5** (MC-ABE with Repetitions). Let type  $\in \{\text{log-att, const-loc}, \leq \text{const-thr}, \geq \text{const-thr}\}$ . If type = log-att, pick k and n such that  $kn = O(\log \lambda)$ . Otherwise, let  $k = n = \text{poly}(\lambda)$ . Our construction uses the following ingredients:

• A slotted IPFE scheme iFE = (iSetup, iEnc, iKeyGen, iDec) based on a pairing group  $\mathbb{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, g_1, g_2, g_t, e, p)$ .

• An identity-based encryption scheme  $\mathsf{idFE} = (\mathsf{idSetup}, \mathsf{idEnc}, \mathsf{idKeyGen}, \mathsf{idDec})$  with identity space  $\mathcal{I} = \mathcal{L} \times \{0, 1\}^{[0;k]}$  for  $\mathcal{L} = \{0, 1\}^{\mathrm{poly}(\lambda)}$  and message space being the ciphertext space of  $\mathsf{iFE}$ 

The MC-ABE scheme aFE for *n* clients and the policy class  $\mathcal{F}^{\mathsf{type}}$  with message space  $\mathcal{M} = \mathbb{G}_{\mathsf{t}}$ , label space  $\mathcal{L}$  and attribute universe  $\mathcal{X} = \{0, 1\}^k$  works as follows:

 $\mathsf{Setup}(1^{\lambda})$  takes as input the security parameter  $1^{\lambda}$  and generates

$$\begin{split} &(\mathsf{iMPK}_0,\mathsf{iMSK}_0) \leftarrow \mathsf{iSetup}(1^{\lambda},\{1,2\},\{3\})\\ &\left\{(\mathsf{iMPK}_{i,j}^{\beta},\mathsf{iMSK}_{i,j}^{\beta}) \leftarrow \mathsf{iSetup}(1^{\lambda},\{1\},\{2\})\right\}_{(i,j)\in[0;n-1]\times[0;k]}^{\beta\in\{0,1\}}\\ &\left\{(\mathsf{idMPK}_i,\mathsf{idMSK}_i) \leftarrow \mathsf{idSetup}(1^{\lambda})\right\}_{i\in[n-1]} \,. \end{split}$$

Then it outputs (MPK, MSK,  $\{SK_i\}_{i \in [n-1]}$ ) as follows:

$$\begin{split} \mathsf{MPK} &= \left(\mathsf{iMPK}_{0}, \{\mathsf{iMPK}_{i,j}^{\beta}\}_{(i,j)\in[0;n-1]\times[0;k]}^{\beta\in\{0,1\}}, \{\mathsf{idMPK}_{i}\}_{i\in[n-1]}\right) \\ \mathsf{MSK} &= \left(\mathsf{iMSK}_{0}, \{\mathsf{iMSK}_{i,j}^{\beta}\}_{(i,j)\in[0;n-1]\times[0;k]}^{\beta\in\{0,1\}}\right) \\ \{\mathsf{SK}_{i} &= \mathsf{idMSK}_{i}\}_{i\in[n-1]} . \end{split}$$

We implicitly parse these keys in the algorithms below.

Enc(MPK, lab,  $\mathbf{x}_0$ ,  $\llbracket \mu \rrbracket_t$ ) takes MPK, a label lab  $\in \mathcal{L}$ , an attribute  $\mathbf{x}_0 \in \{0, 1\}^k$  and a message  $\llbracket \mu \rrbracket_t \in \mathbb{G}_t$  as input. We define  $q_0 := |\Pi_{\mathbf{x}_0}^{\text{type}}|$  and parse  $\Pi_{\mathbf{x}_0}^{\text{type}} = \{\mathbf{y}^{\nu}\}_{\nu \in [q_0]}$  where  $\mathbf{y}^{\nu} = (\mathbf{y}_1^{\nu}, \dots, \mathbf{y}_n^{\nu})$ . For  $i \in [0; n-1]$ , let  $Y_i^{\nu} = \{i\} \times \{j \in [0; k] : \mathbf{y}_i^{\nu}[j] \neq \bot\}$ . For convenience, we also set  $Y^{\nu} = \bigcup_{i \in [0; n-1]} Y_i^{\nu}$  and  $Y_{\geq 1}^{\nu} = Y^{\nu} \setminus Y_0^{\nu}$ . The algorithm samples random elements  $\llbracket r^1 \rrbracket_1, \dots, \llbracket r^{q_0} \rrbracket_1, \llbracket \sigma \rrbracket_1 \notin \mathbb{G}_1$ , computes  $\llbracket d \rrbracket_t = \llbracket \sigma + \mu \rrbracket_t$  and generates

$$\begin{split} & \left\{ \mathsf{iCT}_{i,j}^{\nu} \leftarrow \mathsf{iSlotEnc}(\mathsf{iMPK}_0, \llbracket (r^{\nu}, \sigma) \rrbracket_1) \right\}_{(i,j) \in Y^{\nu}}^{\nu \in [q_0]} \\ & \left\{ \mathsf{iCT}_{i,j}^{\nu} \leftarrow \mathsf{iSlotEnc}(\mathsf{iMPK}_{i,j}^{\mathbf{y}_i^{\nu}[j]}, \llbracket r^{\nu} \rrbracket_1) \right\}_{(i,j) \in Y^{\nu}}^{\nu \in [q_0]} \\ & \left\{ \mathsf{idCT}_{i,j}^{\nu} \leftarrow \mathsf{idEnc}(\mathsf{idMPK}_i, (\mathsf{lab}, \mathbf{y}_i^{\nu}), \mathsf{iCT}_{i,j}^{\nu}) \right\}_{(i,j) \in Y_{\geq 1}^{\nu}}^{\nu \in [q_0]} \\ \end{split}$$

. .

Finally, it outputs the ciphertext

$$\mathsf{CT}_{\mathsf{lab}} = \left( \llbracket d \rrbracket_{\mathsf{t}}, \{\mathsf{i}\mathsf{CT}_0^\nu\}^{\nu \in [q_0]}, \{\mathsf{i}\mathsf{CT}_{0,j}^\nu\}_{(0,j) \in Y_0^\nu}^{\nu \in [q_0]}, \{\mathsf{i}\mathsf{d}\mathsf{CT}_{i,j}^\nu\}_{(i,j) \in Y_{\geq 1}^\nu}^{\nu \in [q_0]} \right) \; .$$

AKeyGen(SK<sub>i</sub>, lab,  $\mathbf{x}_i$ ) takes as input SK<sub>i</sub> for some  $i \in [n-1]$ , a label lab  $\in \mathcal{L}$  and an attribute  $\mathbf{x}_i \in \{0,1\}^k$ . We define  $q_i \coloneqq |\Omega_{\mathbf{x}_i}^{\text{type}}|$  and parse  $\Omega_{\mathbf{x}_i}^{\text{type}} = \{\mathbf{z}_i^{\nu}\}_{\nu \in [q_i]}$ . The algorithm outputs  $\mathsf{DK}_{\mathsf{lab},i} = \{\mathsf{idDK}_i^{\nu}\}_{\nu \in [q_i]}$  computed as follows:

 $idDK_i^{\nu} \leftarrow idKeyGen(idMSK_i, (lab, \mathbf{z}_i^{\nu}))$ .

PKeyGen(MSK, f) takes as input MSK and a policy  $f \in \mathcal{F}^{\text{type}}$ . Let  $(c \wedge f) = (\mathbf{M} \in \mathbb{Z}_p^{m \times \ell}, \rho = (\rho_1, \rho_2, \rho_3) \colon [\ell] \to [0; n-1] \times [0; k] \times \{0, 1\})$ . The algorithm samples  $s \notin \mathbb{Z}_p$  and  $\mathbf{u} \notin \mathbb{Z}_p^{m-1}$ , computes  $\mathbf{s} = (s, \mathbf{u}^{\top}) \cdot \mathbf{M}$  and generates

$$\begin{split} &\mathsf{i}\mathsf{D}\mathsf{K}_{0} \leftarrow \mathsf{i}\mathsf{K}\mathsf{e}\mathsf{y}\mathsf{Gen}(\mathsf{i}\mathsf{M}\mathsf{S}\mathsf{K}_{0}, \llbracket(s,1,0)\rrbracket_{2}) \\ & \left\{\mathsf{i}\mathsf{D}\mathsf{K}_{\kappa} \leftarrow \mathsf{i}\mathsf{K}\mathsf{e}\mathsf{y}\mathsf{Gen}(\mathsf{i}\mathsf{M}\mathsf{S}\mathsf{K}_{\rho_{1}(\kappa),\rho_{2}(\kappa)}^{\rho_{3}(\kappa)}, \llbracket(\mathbf{s}[\kappa],0)\rrbracket_{2})\right\}_{\kappa \in [\ell]} \end{split}$$

Finally, it outputs  $\mathsf{DK}_f = {\mathsf{iDK}_{\kappa}}_{\kappa \in [0;\ell]}$ .

- $\begin{aligned} \mathsf{Dec}(\mathsf{DK}_f,\{\mathsf{DK}_{\mathsf{lab},i}\}_{i\in[n-1]},\mathsf{CT}_{\mathsf{lab}}) \text{ takes as input a decryption key } \mathsf{DK}_f \text{ for a policy } f \in \mathcal{F}^{\mathsf{type}}, \text{ a set of decryption keys } \{\mathsf{DK}_{\mathsf{lab},i}\}_{i\in[n-1]} \text{ created with respect to attributes } \\ \mathbf{x}_1,\ldots,\mathbf{x}_{n-1}\in\{0,1\}^k \text{ and a label } \mathsf{lab}\in\mathcal{L}, \text{ and a ciphertext } \mathsf{CT}_{\mathsf{lab}} \text{ created with respect to attributes } \\ \mathbf{x}_1,\ldots,\mathbf{x}_{n-1}\in\{0,1\}^k \text{ and a label } \mathsf{lab}\in\mathcal{L}, \text{ and a ciphertext } \mathsf{CT}_{\mathsf{lab}} \text{ created with respect to a tribute } \\ \text{to an attribute } \mathbf{x}_0\in\{0,1\}^k \text{ and the same label } \mathsf{lab}. \text{ Parse } \mathsf{DK}_f, \{\mathsf{DK}_{\mathsf{lab},i}\}_{i\in[n-1]}, \\ \mathsf{CT}_{\mathsf{lab}}, \Pi^{\mathsf{type}}_{\mathbf{x}_0}, \{\Omega^{\mathsf{type}}_{\mathbf{x}_i}\}_{i\in[n-1]} \text{ and } \{Y^\nu\}_{\nu\in[q_0]} \text{ as in the algorithms above. Let } \overline{(c \wedge f)} = \\ (\mathbf{M}\in\mathbb{Z}_p^{m\times\ell},\rho\colon[\ell]\to[0;n-1]\times[0;k]\times\{0,1\}) \text{ The algorithm picks indices } \nu_0\in[q_0], \nu_1\in[q_1],\ldots,\nu_{n-1}\in[q_{n-1}] \text{ such that} \end{aligned}$ 
  - 1.  $\mathbf{y}_{i}^{\nu_{0}} = \mathbf{z}_{i}^{\nu_{i}}$  for all  $i \in [n-1]$ , and
  - 2.  $X = \{(i, j, \mathbf{y}_i^{\nu_0}[j]) : (i, j) \in Y^{\nu_0}\} \cap \rho([\ell]) \text{ satisfies the policy } \overline{(c \wedge f)}.$

If no such indices exist, then the algorithm outputs  $\perp$ . Otherwise, it decrypts

 $\left\{\mathsf{iCT}_{i,j} \leftarrow \mathsf{idDec}(\mathsf{idDK}_i^{\nu_i},\mathsf{idCT}_{i,j}^{\nu_0})\right\}_{(i,j)\in Y_{>1}^{\nu_0}} \ ,$ 

and finds coefficients  $\omega_1, \ldots, \omega_\ell \in \mathbb{Z}_p$  such that  $\sum_{\kappa \in [\ell]} \omega_\kappa \mathbf{M}[\kappa] = \mathbf{e}_1$  and  $\omega_\kappa = 0$  for all  $\kappa \notin \rho^{-1}(X)$ . Finally, the algorithm computes

$$\begin{split} & [\![d_0]\!]_{\mathsf{t}} \leftarrow \mathsf{iDec}(\mathsf{iDK}_0,\mathsf{iCT}_0) \\ & \big\{ [\![d_\kappa]\!]_{\mathsf{t}} \leftarrow \mathsf{iDec}(\mathsf{iDK}_\kappa,\mathsf{iCT}_{\rho_1(\kappa),\rho_2(\kappa)}) \big\}_{\kappa \in \rho^{-1}(X)} \ , \end{split}$$

and outputs  $\llbracket \mu' \rrbracket_{\mathsf{t}} = \llbracket d \rrbracket_{\mathsf{t}} - \llbracket d_0 \rrbracket_{\mathsf{t}} + \sum_{\kappa \in \rho^{-1}(X)} \omega_{\kappa} \llbracket d_{\kappa} \rrbracket_{\mathsf{t}}.$ 

**Correctness.** Let type  $\in \{\text{log-att, const-dep}, \leq \text{const-thr}, \geq \text{const-thr}\}$ . We consider attributes  $\mathbf{x}_0, \ldots, \mathbf{x}_{n-1} \in \{0, 1\}^k$  and  $f \in \mathcal{F}^{\text{type}}$  such that  $f(\mathbf{x}_0, \ldots, \mathbf{x}_{n-1}) = 1$ . Let  $\text{lab} \in \mathcal{L}$  and  $\mathsf{DK}_f$ ,  $\{\mathsf{DK}_{\mathsf{lab},i}\}_{i\in[n-1]}$  and  $\mathsf{CT}_{\mathsf{lab}}$  be computed as in Construction 5. We first argue that there always exists a possible choice of indices  $\nu_0, \ldots, \nu_{n-1}$  which satisfies conditions 1 and 2. For convenience, we parse  $\Pi^{\text{type}}_{\mathbf{x}_0}$  and  $\{\Omega^{\text{type}}_{\mathbf{x}_i}\}_{i\in[n-1]}$  as in Construction 5.

- type = log-att. In this case, we have  $\nu_i = 1$  for all  $i \in [n-1]$  since  $q_i = 1$ , and thus  $\mathbf{z}_i^{\nu_i} = \mathbf{x}_i$ . For condition 1, we furthermore observe that there exists an index  $\nu_0 \in [q_0]$  such that  $\mathbf{y}_i^{\nu_0} = \mathbf{x}_i$  for all  $i \in [n-1]$ . This is because by definition,  $\Pi_{\mathbf{x}_0}^{\log-\text{att}'}$  contains the tuple  $(\mathbf{x}_0, \ldots, \mathbf{x}_{n-1})$  for all possible choices of  $\mathbf{x}_1, \ldots, \mathbf{x}_{n-1}$ . For condition 2, we first note that the choice of  $\nu_0$  is unique, so X is uniquely determined. Then it is not hard to see that this set X satisfies the policy  $(c \wedge f)$  since c only checks that there is one decryption key  $\mathsf{DK}_{\mathsf{lab},i}$  for each i and  $f(\mathbf{x}_0, \ldots, \mathbf{x}_{n-1}) = 1$ by assumption.
- type = const-dep. We first observe that a circuit f of depth d = O(1) depends on at most  $\tau = 2^d = O(1)$  inputs. For this reason, there exists a subset  $L \subseteq [n] \times [k]$ of size  $\tau$  such that the output of f only depends on inputs in L. (If the locality of f is smaller than  $\tau$ , then the choice of L is not unique). For condition 1, let  $i \in$ [n-1]. By construction of  $\Omega_{\mathbf{x}_i}^{\text{const-dep}}$ , there exists an index  $\nu_i$  such that  $\mathbf{z}_i^{\nu_i}[j] = \bot$ for  $(i,j) \in (\{i\} \times [k]) \setminus L$  and  $\mathbf{z}_i^{\nu_i}[j] = \mathbf{x}_i[j]$  for  $(i,j) \in L$ . Furthermore, for each choice of L and  $\mathbf{x}_1, \ldots, \mathbf{x}_{n-1}$ , there exists an index  $\nu_0 \in [q_0]$  such that for all  $(i,j) \in [n-1] \times [k]$ , it holds  $\mathbf{y}_i^{\nu_0}[j] = \mathbf{x}_i[j]$  if  $(i,j) \in L$  and  $\mathbf{y}_i^{\nu_0}[j] = \bot$  otherwise. For condition 2, the argument is similar to the previous case. Once we have found an assignment for  $\nu_0, \ldots, \nu_{n-1}$  that satisfies the first condition, the second follows since  $f(\mathbf{x}_0, \ldots, \mathbf{x}_{n-1}) = 1$ .
- type =  $\leq$ const-thr. The argument is the same as in the case type = const-dep except that now all tuples  $(\mathbf{y}_0^{\nu_0}, \ldots, \mathbf{y}_{n-1}^{\nu_0}) \in \prod_{\mathbf{x}_0}^{\leq \text{const-thr}}$  and vectors  $\mathbf{z}_i^{\nu_i} \in \Omega_{\mathbf{x}_i}^{\leq \text{const-thr}}$ , for  $i \in [n-1]$ , have coordinates in  $\{1, \bot\}$  instead of  $\{0, 1, \bot\}$ . For condition 1, this

is irrelevant as we replace the 0 coordinates with  $\perp$  in both the  $\mathbf{y}_i^{\nu_0}$  and the  $\mathbf{z}_i^{\nu_i}$  vectors. So their equality is preserved. For condition 2, we exploit the fact that all policies in  $\mathcal{F}^{\leq \text{const-thr}}$  are monotone. Therefore, if a coordinate of  $\mathbf{y}_i^{\nu_0}$  or  $\mathbf{z}_i^{\nu_i}$  is 0 or  $\perp$  is irrelevant for the fact whether X satisfies  $(c \wedge f)$  or not.

• type =  $\geq$ const-thr. The argument is similar to the case type = log-att. For  $i \in [n-1]$ , we have  $\nu_i = 1$  and for all  $j \in [k]$ ,  $\mathbf{z}_i^1[j] = 1$  if  $\mathbf{x}_i = 1$  and  $\mathbf{z}_i^1[j] = \bot$  otherwise. If  $\sum_{i \in [0;n-1]} \delta(\mathbf{x}_i) \geq kn - \tau$ , then there also exists  $\nu_0 \in [q_0]$  such that  $\mathbf{y}_i^{\nu_0} = \mathbf{z}_i^1$  for all  $i \in [n-1]$ . In contrast, such a  $\nu_0$  does not exist if  $\sum_{i \in [0;n-1]} \delta(\mathbf{x}_i) < kn - \tau$ . But this is irrelevant since  $f(\mathbf{x}_0, \ldots, \mathbf{x}_{n-1}) = 0$  in this case, *i.e.* such a combination of attributes is never authorized regardless of the policy f.

Once the algorithm has chosen indices  $\nu_0, \ldots, \nu_{n-1}$  that satisfy conditions 1 and 2, we can argue as in Construction 2. By the correctness of idFE and iFE, we have  $d_0 = r^{\nu_0} \cdot s + \sigma$  and, for all  $\kappa \in \rho^{-1}(X)$ ,  $d_{\kappa} = r^{\nu_0} \cdot \mathbf{s}[\kappa]$ . Then we conclude from the correctness of the LSS scheme that

$$\mu' = d - d_0 + \sum_{\kappa \in \rho^{-1}(X)} \omega_\kappa d_\kappa = (\sigma + \mu) - (r^{\nu_0}s + \sigma) + r^{\nu_0} \cdot \underbrace{\sum_{\kappa \in \rho^{-1}(X)} \omega_\kappa \mathbf{s}[\kappa]}_{= s} = \mu$$

**Efficiency.** For efficiency, we need to ensure that  $\Pi_{\mathbf{x}_0}^{\mathsf{type}}$  and  $\Omega_{\mathbf{x}_i}^{\mathsf{type}}$  have polynomial size for all  $i \in [n-1]$  and  $\mathsf{type} \in \{\mathsf{log-att}, \mathsf{const-dep}, \leq \mathsf{const-thr}, \geq \mathsf{const-thr}\}.$ 

• type = log-att. By construction, we have  $|\Omega_{\mathbf{x}_i}^{\mathsf{log-att}}| = 1$  and

$$\left|\Pi_{\mathbf{x}_0}^{\mathsf{log-att}}\right| = 2^{k(n-1)} = 2^{O(\log \lambda)} = \operatorname{poly}(\lambda) ,$$

since  $kn = O(\log \lambda)$ .

• type = const-dep. Since  $\tau = 2^d = O(1)$ , we have

$$\begin{split} \left| \Pi_{\mathbf{x}_0}^{\mathsf{const-dep}} \right| &\leq \binom{kn}{\tau} \cdot 2^{\tau} = \mathrm{poly}(\lambda) \\ \left| \Omega_{\mathbf{x}_i}^{\mathsf{const-dep}} \right| &\leq \sum_{\ell \in [0;\tau]} \binom{k}{\ell} \cdot 2^{\ell} = \mathrm{poly}(\lambda) \end{split}$$

• type =  $\leq$  const-thr. Similar to the case type = const-dep, as  $\tau = O(1)$ , we have

$$\begin{aligned} \left| \Pi_{\mathbf{x}_{0}}^{\leq \text{const-thr}} \right| &\leq \binom{kn}{\tau} = \text{poly}(\lambda) \\ \left| \Omega_{\mathbf{x}_{i}}^{\leq \text{const-thr}} \right| &\leq \sum_{\ell \in [0;\tau]} \binom{k}{\ell} = \text{poly}(\lambda) \end{aligned}$$

• type =  $\geq$  const-thr. Since  $\tau = O(1)$ , we have  $|\Omega_{\mathbf{x}_i}^{\geq \text{const-thr}}| = 1$  and

$$\left|\Pi_{\mathbf{x}_0}^{\geq \text{const-thr}}\right| \leq \sum_{\ell \in [0;\tau]} \binom{kn}{kn-\ell} = \text{poly}(\lambda) \ .$$

Security. We prove selective security with repetitions.

**Proposition 6.** Let type  $\in$  {log-att, const-dep,  $\leq$  const-thr,  $\geq$  const-thr}. If the DDH assumption holds in  $\mathbb{G}_2$ , iFE is slot-mode correct and function-hiding and idFE is secure, then Construction 5 is sel-rep-secure.

When instantiating iFE with the slotted ABE from [LL20a] and idFE with the IBE from [CLL $^+13$ ], we obtain the following theorem.

Theorem 7. Assuming SXDH on pairings, there exist

- a sel-rep-secure MC-ABE for the policy class  $NC^1$  where the parameters satisfy  $kn = O(\log \lambda)$ , and
- a sel-rep-secure MC-ABE for the policy classes  $\mathcal{F}^{\text{const-dep}}$ ,  $\mathcal{F}^{\leq \text{const-thr}}$  and  $\mathcal{F}^{\geq \text{const-thr}}$  with  $k = n = \text{poly}(\lambda)$ .

Proof (Proposition 6). Let  $\mathcal{O}\mathsf{Enc}(\mathsf{lab}, \mathbf{x}_0, \llbracket \mu^0 \rrbracket_t, \llbracket \mu^1 \rrbracket_t)$  be the (single) encryption query. We define  $q_0 \coloneqq |\Pi^{\mathsf{type}}_{\mathbf{x}_0}|$  and parse  $\Pi^{\mathsf{type}}_{\mathbf{x}_0} = \{\mathbf{y}^\nu\}_{\nu \in [q_0]}$  where  $\mathbf{y}^\nu = (\mathbf{y}_1^\nu, \dots, \mathbf{y}_n^\nu)$ . For  $i \in [0; n-1]$ , let  $Y_i^\nu = \{i\} \times \{j \in [0; k] : \mathbf{y}_i^\nu[j] \neq \bot\}, Y^\nu = \bigcup_{i \in [0; n-1]} Y_i^\nu$  and  $Y_{\geq 1}^\nu = Y^\nu \setminus Y_0^\nu$ . For the reply to the encryption query, the challenger samples random elements  $\llbracket r^1 \rrbracket_1, \dots, \llbracket r^{q_0} \rrbracket_1, \llbracket \sigma \rrbracket_1 \stackrel{\&}{\leftarrow} \mathbb{G}_1$ , computes  $\llbracket d \rrbracket_t = \llbracket \sigma + \mu \rrbracket_t$  and generates

$$\begin{split} & \left\{ \mathsf{iCT}_{0}^{\nu} \leftarrow \mathsf{iSlotEnc}(\mathsf{iMPK}_{0}, \llbracket (r^{\nu}, \sigma) \rrbracket_{1}) \right\}^{\nu \in [q_{0}]} \\ & \left\{ \mathsf{iCT}_{i,j}^{\nu} \leftarrow \mathsf{iSlotEnc}(\mathsf{iMPK}_{i,j}^{\mathbf{y}_{i}^{\nu}[j]}, \llbracket r^{\nu} \rrbracket_{1}) \right\}_{(i,j) \in Y^{\nu}}^{\nu \in [q_{0}]} \\ & \left\{ \mathsf{idCT}_{i,j}^{\nu} \leftarrow \mathsf{idEnc}(\mathsf{idMPK}_{i}, (\mathsf{lab}, \mathbf{y}_{i}^{\nu}), \mathsf{iCT}_{i,j}^{\nu}) \right\}_{(i,j) \in Y_{\geq 1}}^{\nu \in [q_{0}]} \end{split}$$

Then it sends the following challenge ciphertext  $\mathsf{CT}$  to  $\mathcal{A}$ :

$$\mathsf{CT}_{\mathsf{lab}} = \left( \llbracket d \rrbracket_{\mathsf{t}}, \{\mathsf{iCT}_0^\nu\}^{\nu \in [q_0]}, \{\mathsf{iCT}_{0,j}^\nu\}_{(0,j) \in Y_0^\nu}^{\nu \in [q_0]}, \{\mathsf{idCT}_{i,j}^\nu\}_{(i,j) \in Y_{\geq 1}^\nu}^{\nu \in [q_0]} \right) \ .$$

We consider a sequence of hybrid games  $G^b_{\nu}$ , for  $\nu \in [0; q_0]$  and  $b \in \{0, 1\}$ , where  $G^b_{\nu}$  is the same as  $\operatorname{Exp}_{\mathsf{aFE},\mathcal{A}}^{\mathsf{mc-abe-}b}(1^{\lambda})$  except that, for each  $\nu' \in [q_0]$ , the challenger computes

$$\mathsf{iCT}_{0}^{\nu'} \leftarrow \begin{cases} \mathsf{iSlotEnc}(\mathsf{iMPK}_{0}, \llbracket(r^{\nu'}, \mathbf{0})\rrbracket_{1}) & \text{if } \nu' \leq \nu \\ \mathsf{iSlotEnc}(\mathsf{iMPK}_{0}, \llbracket(r^{\nu'}, \sigma)\rrbracket_{1}) & \text{if } \nu' > \nu \end{cases}$$

We note that  $G_0^b = \mathbf{Exp}_{\mathsf{aFE},\mathcal{A}}^{\mathsf{mc-abe}-b}(1^{\lambda})$  and  $G_{q_0}^0 \equiv G_{q_0}^1$ . For the latter, we observe that in the hybrids  $G_{q_0}^0$  and  $G_{q_0}^1$  the random mask  $\llbracket \sigma \rrbracket_t \stackrel{\text{\tiny{\$}}}{=} \mathbb{G}_t$  appears only in the terms  $\llbracket d \rrbracket_t \coloneqq \llbracket \sigma + \mu^b \rrbracket_t$ , which implies that the distributions  $\llbracket \sigma + \mu^0 \rrbracket_t$  and  $\llbracket \sigma + \mu^1 \rrbracket_t$  are identical. Below we prove the following claim for all  $\nu \in [q_0]$  and  $b \in \{0, 1\}$ .

Claim 8. We have  $\mathsf{G}_{\nu-1}^b \approx_c \mathsf{G}_{\nu}^b$  under the slot-mode correctness and function-hiding security of iFE, the security of idFE and the DDH assumption in  $\mathbb{G}_2$ .

This concludes the proof of the proposition.

We next prove the claim.

Proof (Claim 8). We consider a sequence of hybrids  $\widehat{\mathsf{G}}_0^0, \ldots, \widehat{\mathsf{G}}_6^0, \widehat{\mathsf{G}}_6^1, \ldots, \widehat{\mathsf{G}}_0^1$ , where  $\widehat{\mathsf{G}}_0^0 = \mathsf{G}_{\nu-1}^b$  and  $\widehat{\mathsf{G}}_0^1 = \mathsf{G}_{\nu}^b$ . We argue  $\widehat{\mathsf{G}}_{\alpha-1}^\beta \approx_c \widehat{\mathsf{G}}_{\alpha}^\beta$ , for  $\alpha \in [6]$  and  $\beta \in \{0, 1\}$ , as well as  $\mathsf{G}_6^0 = \mathsf{G}_6^1$ . Then the claim follows via a hybrid argument. Modifications between consecutive games are highlighted using boxes.

**Game**  $\widehat{\mathsf{G}}_0^{\beta}$ : This is  $\mathsf{G}_{\nu-1+\beta}$ .

Game  $\widehat{\mathsf{G}}_1^\beta$ : This game is the same as  $\widehat{\mathsf{G}}_0^\beta$ , except that the challenger computes

$$\begin{split} & \mathsf{iCT}_0^{\nu} \leftarrow \, \mathsf{iEnc}(\mathsf{iMSK}_0, [\![(r^{\nu}, \rho^{\beta}, 0)]\!]_1) \\ & \big\{ \mathsf{iCT}_{i,j}^{\nu} \leftarrow \, \overline{\mathsf{iEnc}}(\mathsf{iMSK}_{i,j}^{\mathbf{y}_i^{\nu}[j]}, [\![(r^{\nu}, 0)]\!]_1) \, \big\}_{(i,j) \in Y^{\nu}} \end{split}$$

where  $\rho^0 = \sigma$  and  $\rho^1 = 0$ . We have  $\widehat{\mathsf{G}}_0^\beta \equiv \widehat{\mathsf{G}}_1^\beta$  which follows from the slot-mode correctness of iFE.

**Game**  $\widehat{\mathsf{G}}_{2}^{\beta}$ : We define the set  $J \subseteq [0; n-1] \times [0; k]$  containing the indices of all ciphertexts  $\mathsf{iCT}_{i,j}^{\nu}$  known to the adversary:

$$J = \left\{ (i,j) \in Y^{\nu} : \left[ i \in \mathcal{C} \cup \{0\} \right] \lor \left[ \exists (i,\mathsf{lab},\mathbf{x}_i) \in \mathcal{Q}_{\mathsf{akey}} \text{ s.t. } \mathbf{y}_i^{\nu} \in \Omega_{\mathbf{x}_i}^{\mathsf{type}} \right] \right\} \ .$$

This game is the same as  $\widehat{\mathsf{G}}_1^{\beta}$ , except that the challenger now computes

$$\mathsf{iCT}_{i,j}^{\nu} \leftarrow \begin{cases} \mathsf{iEnc}(\mathsf{iMSK}_{i,j}^{\mathbf{y}_i^{\nu}[j]}, \llbracket (r^{\nu}, 0) \rrbracket_1) & \text{if } (i,j) \in J \\ \mathsf{iEnc}(\mathsf{iMSK}_{i,j}^{\mathbf{y}_i^{\nu}[j]}, \llbracket (\mathbf{0}, 0) \rrbracket_1) & \text{if } (i,j) \notin J \end{cases}.$$

We have  $\widehat{\mathsf{G}}_1^\beta \approx_c \widehat{\mathsf{G}}_2^\beta$  from the security of idFE.

**Game**  $\widehat{\mathsf{G}}_3^\beta$ : Next we deal with the ciphertexts  $\mathsf{iCT}_{i,j}^\nu$  such that  $(i,j) \in J$ . Initially, the challenger needs to respond to the selective encryption query. In contrast to the previous game, it now computes

$$\begin{split} \mathsf{iCT}_0 &\leftarrow \mathsf{iEnc}(\mathsf{iMSK}_0, \llbracket (0,0,1) \rrbracket_1) \\ \mathsf{iCT}_{i,j} &\leftarrow \begin{cases} \mathsf{iEnc}(\mathsf{iMSK}_{i,j}^{\mathbf{y}_i^{\nu}[j]}, \llbracket (0,1) \rrbracket_1) & \text{if } (i,j) \in J \\ \mathsf{iEnc}(\mathsf{iMSK}_{i,j}^{\mathbf{y}_i^{\nu}[j]}, \llbracket (0,0) \rrbracket_1) & \text{if } (i,j) \notin J \end{cases}. \end{split}$$

Upon receiving a query  $\mathcal{O}\mathsf{PKeyGen}(f)$  with  $\overline{(c \wedge f)} = (\mathbf{M} \in \mathbb{Z}_p^{m \times \ell}, \rho \colon [\ell] \to [0; n-1] \times [0; k] \times \{0, 1\})$ , the challenger samples  $s \stackrel{\text{\tiny{\&}}}{\leftarrow} \mathbb{Z}_p$ ,  $\mathbf{u} \stackrel{\text{\tiny{\&}}}{\leftarrow} \mathbb{Z}_p^{m-1}$  and computes  $\mathbf{s} = (s, \mathbf{u}^{\top}) \cdot \mathbf{M}$ . Then it sends  $\mathsf{DK}_f = \{\mathsf{i}\mathsf{DK}_\kappa\}_{\kappa \in [0; \ell]}$  to  $\mathcal{A}$  generated as follows:

$$\begin{split} &\mathsf{i}\mathsf{D}\mathsf{K}_{0} \leftarrow \mathsf{i}\mathsf{K}\mathsf{e}\mathsf{y}\mathsf{Gen}(\mathsf{i}\mathsf{M}\mathsf{S}\mathsf{K}_{0}, \llbracket(s, 1, \llbracket r^{\nu} \cdot s + \rho^{\beta}) \rrbracket_{2}) \\ &\mathsf{i}\mathsf{D}\mathsf{K}_{\kappa} \leftarrow \begin{cases} \mathsf{i}\mathsf{K}\mathsf{e}\mathsf{y}\mathsf{Gen}(\mathsf{i}\mathsf{M}\mathsf{S}\mathsf{K}_{\rho(\kappa)}, \llbracket(\mathbf{s}[\kappa], \llbracket r^{\nu} \cdot \mathbf{s}[\kappa]]) \rrbracket_{2}) & \text{if } \rho(\kappa) \in J \\ \mathsf{i}\mathsf{K}\mathsf{e}\mathsf{y}\mathsf{Gen}(\mathsf{i}\mathsf{M}\mathsf{S}\mathsf{K}_{\rho(\kappa)}, \llbracket(\mathbf{s}[\kappa], 0) \rrbracket_{2}) & \text{if } \rho(\kappa) \notin J \end{cases}, \end{split}$$

where the scalar  $r^{\nu} \stackrel{\hspace{0.1em} {\scriptscriptstyle \bullet}}{\underset{\hspace{0.1em} {\scriptscriptstyle \bullet}}{\overset{\hspace{0.1em}}{\scriptscriptstyle \bullet}}} \mathbb{Z}_p$  is sampled during the creation of the challenge ciphertext (but not used). As the public parts of the iFE keys and the inner products between vectors embedded in the iFE keys and ciphertexts do not change, it follows  $\widehat{\mathsf{G}}_2^{\beta} \approx_c \widehat{\mathsf{G}}_3^{\beta}$  from the function-hiding security of iFE.

**Game**  $\widehat{\mathsf{G}}_4^{\beta}$ : This game is the same as  $\widehat{\mathsf{G}}_3^{\beta}$  except that the challenger embeds a fresh secret sharing in each secret key. Specifically, upon receiving a query  $\mathcal{O}\mathsf{PKeyGen}(f)$  with  $\overline{(c \wedge f)} = (\mathbf{M} \in \mathbb{Z}_p^{m \times \ell}, \rho \colon [\ell] \to [0; n-1] \times [0; k] \times \{0,1\})$ , the challenger samples  $s, t \stackrel{\text{\sc{s}}}{=} \mathbb{Z}_p$  and  $\mathbf{u}, \mathbf{v} \stackrel{\text{\sc{s}}}{=} \mathbb{Z}_p^{m-1}$ , computes  $\mathbf{s} = (s, \mathbf{u}^{\top}) \cdot \mathbf{M}$  and  $\mathbf{t} = (t, \mathbf{v}^{\top}) \cdot \mathbf{M}$  and sends  $\mathsf{DK}_f = \{\mathsf{i}\mathsf{DK}_{\kappa}\}_{\kappa \in [0; \ell]}$  to  $\mathcal{A}$  generated as follows:

$$\begin{split} & \mathsf{i}\mathsf{D}\mathsf{K}_{0} \leftarrow \mathsf{i}\mathsf{KeyGen}(\mathsf{i}\mathsf{M}\mathsf{S}\mathsf{K}_{0}, [\![(s, 1, t + \rho^{\beta})]\!]_{2}) \\ & \mathsf{i}\mathsf{D}\mathsf{K}_{\kappa} \leftarrow \begin{cases} \mathsf{i}\mathsf{KeyGen}(\mathsf{i}\mathsf{M}\mathsf{S}\mathsf{K}_{\rho(\kappa)}, [\![(\mathbf{s}[\kappa], t[\kappa]])]\!]_{2}) & \text{if } \rho(\kappa) \in J \\ & \mathsf{i}\mathsf{KeyGen}(\mathsf{i}\mathsf{M}\mathsf{S}\mathsf{K}_{\rho(\kappa)}, [\![(\mathbf{s}[\kappa], 0)]\!]_{2}) & \text{if } \rho(\kappa) \notin J \end{cases}. \end{split}$$

It is not hard to see that  $\widehat{\mathsf{G}}_{3}^{\beta} \approx_{c} \widehat{\mathsf{G}}_{4}^{\beta}$  under the DDH assumption in  $\mathbb{G}_{2}$ . (Note that we can exploit the random self-reducibility of the DDH problem here, so a single DDH instance suffices).

Let  $f \in \mathcal{Q}_{\mathsf{pkey}}$  and  $\overline{(c \wedge f)} = (\mathbf{M} \in \mathbb{Z}_p^{m \times \ell}, \rho \colon [\ell] \to ([0; n - 1] \times [0; k] \times \{0, 1\}))$ . We recall that the adversary can learn the iFE ciphertexts corresponding to the attributes with indices in  $J' \coloneqq \{(i, j, \mathbf{y}_i^{\nu}[j])\}_{(i,j) \in J}$ . Let  $M_f = \operatorname{span}\{\mathbf{M}[\kappa] : \kappa \in [\ell] \land \rho(\kappa) \in J'\}$  be the vector space spanned by the columns of  $\mathbf{M}$  associated with attributes in J'. Furthermore, let  $M_f^{\perp}$  denote the orthogonal complement of  $M_f$  and  $A_f^{\perp}$  the affine space of  $M_f^{\perp}$  containing all vectors whose first coordinate is 1. To argue that  $A_f^{\perp}$  is nonempty, we distinguish two cases:

- 1. If there exists  $i \in [n-1] \setminus \mathcal{C}$  such that  $\mathbf{y}_i^{\nu} \notin \Omega_{\mathbf{x}_i}^{\text{type}}$  for all attributes  $\mathbf{x}_i$  with  $(i, \mathsf{lab}, \mathbf{x}_i) \in \mathcal{Q}_{\mathsf{akey}}$ , then J' does not satisfy c (and thus  $\overline{(c \wedge f)}$ ). Hence, we have  $\mathbf{e}_1 \notin M_f$  which implies that  $A_f^{\perp}$  is nonempty.
- 2. Otherwise, consider any sequence of attributes  $\mathbf{x}_1, \ldots, \mathbf{x}_{n-1} \in \{0, 1\}^k$  such that for all  $i \in [n-1]$ , we have

$$\left[i \in \mathcal{C} \lor (i, \mathsf{lab}, \mathbf{x}_i) \in \mathcal{Q}_{\mathsf{akey}}\right] \land \left[\mathbf{y}_i^{\nu} \in \Omega_{\mathbf{x}_i}^{\mathsf{type}}\right] \ .$$

For the sake of a contradiction, assume that J' satisfies  $(c \wedge f)$ . Then it follows that  $J'' := \{(i,0,1)\}_{i \in [n-1]} \cup \{(i,j,\mathbf{x}_i[j])\}_{(i,j) \in [0;n-1] \times [k]}$  also satisfies  $(c \wedge f)$  since  $J' \subseteq J''$  and  $(c \wedge f)$  is a monotone policy. In this case, the equivalence between  $(c \wedge f)$  and  $(c \wedge f)$  gives that

$$(c \wedge f)((1, \mathbf{x}_0), \dots, (1, \mathbf{x}_{n-1})) = 1$$

which implies in particular that  $f(\mathbf{x}_0, \ldots, \mathbf{x}_{n-1}) = 1$ . This contradicts the admissibility of the adversary  $\mathcal{A}$ . Therefore, J' cannot satisfy  $\overline{(c \wedge f)}$ , and we conclude that  $\mathbf{e}_1 \notin M_f$  and  $A_f^{\perp}$  is nonempty.

**Game**  $\widehat{\mathbf{G}}_{5}^{\beta}$ : This game is the same as  $\widehat{\mathbf{G}}_{4}^{\beta}$ , except that we modify how the challenger generates the share vector  $\mathbf{t}$  for the reply to a query  $\mathcal{O}\mathsf{PKeyGen}(f)$ . Let  $\overline{(c \wedge f)} = (\mathbf{M} \in \mathbb{Z}_{p}^{m \times \ell}, \rho \colon [\ell] \to [0; n-1] \times [0; k] \times \{0, 1\})$ . In  $\widehat{\mathbf{G}}_{4}^{\beta}$ , the challenger samples  $t \stackrel{\text{s}}{\leftarrow} \mathbb{Z}_{p}$ ,  $\mathbf{v} \stackrel{\text{s}}{\leftarrow} \mathbb{Z}_{p}^{m-1}$  and computes  $\mathbf{t} = (t, \mathbf{v}^{\top}) \cdot \mathbf{M}$ . In the current game, the challenger picks an additional vector  $(1, \overline{\mathbf{v}})^{\top} \in A_{f}^{\perp}$  and sets

$$\mathbf{t} = (t, \mathbf{v}^{\top}) \cdot \mathbf{M} + (0, t \overline{\mathbf{v}}^{\top}) \cdot \mathbf{M} \quad .$$

The distributions  $\{\mathbf{v} : \mathbf{v} \stackrel{s}{\leftarrow} \mathbb{Z}_p^{m-1}\}$  and  $\{\mathbf{v} + t\overline{\mathbf{v}} : \mathbf{v} \stackrel{s}{\leftarrow} \mathbb{Z}_p^{m-1}\}$  are identical for any  $\overline{\mathbf{v}} \in \mathbb{Z}_p^{m-1}$ . The first distribution corresponds to  $\widehat{\mathsf{G}}_4^\beta$  and the second one to  $\widehat{\mathsf{G}}_5^\beta$ (followed by the same post-processing), so we have  $\widehat{\mathsf{G}}_4^\beta \equiv \widehat{\mathsf{G}}_5^\beta$ .

**Game**  $\widehat{\mathbf{G}}_{6}^{\beta}$ : This game is the same as  $\widehat{\mathbf{G}}_{5}^{\beta}$  except that we again change the behavior of the key generation oracle when receiving a query  $\mathcal{O}\mathsf{PKeyGen}(f)$ . Let  $\overline{(c \wedge f)} = (\mathbf{M} \in \mathbb{Z}_{p}^{m \times \ell}, \rho: [\ell] \to [0; n-1] \times [0; k] \times \{0, 1\})$ . As in the previous hybrid, the challenger samples  $s, t \stackrel{*}{\leftarrow} \mathbb{Z}_{p}$  and  $\mathbf{u}, \mathbf{v} \stackrel{*}{\leftarrow} \mathbb{Z}_{p}^{m-1}$  and computes  $\mathbf{s} = (s, \mathbf{u}^{\top}) \cdot \mathbf{M}$ . In the current hybrid, it then generates  $\mathsf{DK}_{f} = \{\mathsf{iDK}_{\kappa}\}_{\kappa \in [0; \ell]}$  as follows:

$$\begin{split} &\mathrm{i}\mathsf{D}\mathsf{K}_{0} \leftarrow \mathrm{i}\mathsf{K}\mathrm{e}\mathrm{y}\mathsf{Gen}(\mathrm{i}\mathsf{M}\mathsf{S}\mathsf{K}_{0}, [\![(s,1,t+\rho^{\beta})]\!]_{2}) \\ &\mathrm{i}\mathsf{D}\mathsf{K}_{\kappa} \leftarrow \begin{cases} \mathrm{i}\mathsf{K}\mathrm{e}\mathrm{y}\mathsf{Gen}(\mathrm{i}\mathsf{M}\mathsf{S}\mathsf{K}_{\rho(\kappa)}, [\![(\mathbf{s}[\kappa],(0,\mathbf{v}^{\top})\cdot\mathbf{M}[\kappa]])]\!]_{2}) & \mathrm{if}\;\rho(\kappa) \in J' \\ \mathrm{i}\mathsf{K}\mathrm{e}\mathrm{y}\mathsf{Gen}(\mathrm{i}\mathsf{M}\mathsf{S}\mathsf{K}_{\rho(\kappa)}, [\![(\mathbf{s}[\kappa],0)]\!]_{2}) & \mathrm{if}\;\rho(\kappa) \notin J' \end{cases}, \end{split}$$

whereas in  $\widehat{\mathsf{G}}_5^{\beta}$ , the challenger additionally picks  $(1, \overline{\mathbf{v}})^{\top} \in A_f^{\perp}$  and computes

$$\mathsf{iDK}_{\kappa} \leftarrow \mathsf{iKeyGen}(\mathsf{iMSK}_{\rho(\kappa)}, [\![(\mathbf{s}[\kappa], (t, (\mathbf{v} + t\overline{\mathbf{v}})^{\top}) \cdot \mathbf{M}[\kappa])]\!]_2)$$

for all  $\kappa \in [\ell]$  such that  $\rho(\kappa) \in J'$ . We can observe that

$$(t, (\mathbf{v} + t\bar{\mathbf{v}})^{\top}) \cdot \mathbf{M}[\kappa] = (0, \mathbf{v}^{\top}) \cdot \mathbf{M}[\kappa] + t(1, \bar{\mathbf{v}}^{\top}) \cdot \mathbf{M}[\kappa]$$
$$= (0, \mathbf{v}^{\top}) \cdot \mathbf{M}[\kappa] ,$$

where the last equality follows from the fact that  $(1, \overline{\mathbf{v}}^{\top}) \in M_f^{\perp}$  and  $\mathbf{M}[\kappa] \in M_f$ . The first distribution corresponds to  $\widehat{\mathbf{G}}_5^{\beta}$  and the second one to  $\widehat{\mathbf{G}}_6^{\beta}$  (followed by the same post-processing), thus we have  $\widehat{\mathbf{G}}_5^{\beta} \equiv \widehat{\mathbf{G}}_6^{\beta}$ . We further notice that in  $\widehat{\mathbf{G}}_6^{\beta}$  the random element  $[\![t]\!]_2 \stackrel{\text{<}}{\leftarrow} \mathbb{G}_2$  appears only in iDK<sub>0</sub>, so it perfectly masks  $[\![\rho^{\beta}]\!]_2$  and we conclude that  $\widehat{\mathbf{G}}_6^{0} \equiv \widehat{\mathbf{G}}_6^{1}$ .

## 6 Construction of MC-PE

In this section, we present our new compiler that turns any O(1)-client ABE into an O(1)client PE scheme for the same policy class.

For each  $\ell \in [0; n-1]$ , we define a permutation  $\pi_{\ell}$  of the set [0; n-1] via

$$\pi_{\ell}(i) = \begin{cases} i+1 & \text{if } i \in [0; \ell-1] \\ 0 & \text{if } i = \ell \\ i & \text{if } i \in [\ell+1; n-1] \end{cases}.$$

Correspondingly, for a policy  $f \in \mathcal{F}$  and  $\ell \in [0; n-1]$ , we define  $\pi_{\ell}(f)$  as a variant of f with permuted inputs:

$$\left(\pi_{\ell}(f)\right)\left(\mathbf{x}_{0},\ldots,\mathbf{x}_{n-1}\right)=f(\mathbf{x}_{\pi_{\ell}(0)},\ldots,\mathbf{x}_{\pi_{\ell}(n-1)})$$

**Construction 9** (Multi-Client Predicate Ecryption). The construction uses the following ingredients:

- An MC-ABE scheme aFE = (aSetup, aEnc, aAKeyGen, aPKeyGen, aDec) for a message space *M* = {0,1}<sup>m</sup> for some m = poly(λ), a label space *L*, an attribute universe *X* and a policy class *F*.
- A lockable obfuscation scheme  $\mathsf{LObf} = (\mathsf{Obf}, \mathsf{Eval})$  with lock space  $\mathcal{M}$  and message space  $\mathcal{M}'$ . We write C[x](y) to indicate that a circuit C has the value x hardwired in its description and takes y as input.

The MC-PE scheme pFE for message space  $\mathcal{M}'$ , label space  $\mathcal{L}$ , attribute universe  $\mathcal{X}$  and policy class  $\mathcal{F}$  works as follows:

Setup(1<sup> $\lambda$ </sup>) takes as input the security parameter 1<sup> $\lambda$ </sup> and generates n aFE instances

$$\left\{(\mathsf{aMPK}_{\ell},\mathsf{aMSK}_{\ell},\{\mathsf{aSK}_{\ell,i}\}_{i\in[n-1]})\leftarrow\mathsf{aSetup}(1^{\lambda})\right\}_{\ell\in[0;n-1]}$$

Then the algorithm outputs  $({SK_i}_{i \in [0;n-1]}, MSK)$  as follows:

 $\left\{\mathsf{SK}_i := (\mathsf{aMPK}_i, \{\mathsf{aSK}_{\ell,j}\}_{(\ell,j)\in J_i})\right\}_{i\in[0:n-1]} , \quad \mathsf{MSK} := \{\mathsf{aMSK}_\ell\}_{\ell\in[0:n-1]} ,$ 

where  $J_i = \{(\ell, \pi_\ell(i))\}_{\ell \in [0; n-1] \setminus \{i\}}$ . We implicitly parse these keys in the algorithms below.

hardwired values : • an aFE ciphertext aCT<sub>i</sub> • a set of aFE attribute decryption keys  $\{aDK_{\ell,i+1}\}_{\ell \in [i+1;n-1]}$ input : • a set of obfuscated circuits  $\{\widetilde{C}_{\ell}\}_{\ell \in [i+1;n-1]}$ • a set of aFE attribute decryption keys  $\{aDK_{\ell,j}\}_{\ell \in [i;n-1]}^{j \in [i]}$ • a set of aFE policy decryption keys  $\{aDK_{\ell,j}\}_{\ell \in [i;n-1]}^{j \in [i]}$ • a set of aFE policy decryption keys  $\{aDK_{\ell,f}\}_{\ell \in [i;n-1]}$ output : an element of the lock space  $\sigma_i \in \mathcal{M}$  or  $\perp$ initialize  $\{aDK_{\ell,j} \leftarrow \perp\}_{\ell \in [0;j-1]}^{j \in [i+1;n-1]}$ for  $k \leftarrow i+1$  to n-1 do  $\begin{bmatrix} c_k \leftarrow Eval(\widetilde{C}_k, (\{\widetilde{C}_\ell\}_{\ell \in [k+1;n-1]}, \{aDK_{\ell,j}\}_{\ell \in [k;n-1]}, \{aDK_{\ell,f}\}_{\ell \in [k;n-1]})) \end{bmatrix}$ if  $c_k = \perp$  then return  $\perp$ else parse  $\{aDK_{\ell,k}\}_{\ell \in [0;k-1]} \leftarrow c_k$ end return  $\sigma_i \leftarrow aDec(aDK_{i,f}, \{aDK_{i,j}\}_{j \in [n-1]}, aCT_i)$ 

Figure 5: Definition of the circuit  $C_i[\mathsf{aCT}_i, \{\mathsf{aDK}_{\ell,i+1}\}_{\ell \in [i+1;n-1]}]$  on input  $(\{\widetilde{C}_\ell\}_{\ell \in [i+1;n-1]}, \{\mathsf{aDK}_{\ell,j}\}_{\ell \in [i;n-1]}^{j \in [i]}, \{\mathsf{aDK}_{\ell,f}\}_{\ell \in [i;n-1]})$ 

 $\mathsf{Enc}(\mathsf{SK}_0,\mathsf{lab},x_0,\mu)$  takes as input  $\mathsf{SK}_0$ , a label  $\mathsf{lab} \in \mathcal{L}$ , an attribute  $x_0 \in \mathcal{X}$  and a message  $\mu \in \mathcal{M}'$ . The algorithm samples  $\sigma_0 \notin \mathcal{M}$ , runs

$$\begin{split} &\mathsf{aCT}_0 \leftarrow \mathsf{aEnc}(\mathsf{aMPK}_0,\mathsf{lab},x_0,\sigma_0) \\ \big\{\mathsf{aDK}_{\ell,1} \leftarrow \mathsf{aAKeyGen}(\mathsf{aSK}_{\ell,1},\mathsf{lab},x_0)\big\}_{\ell \in [n-1]} \\ & \widetilde{C}_0 \leftarrow \mathsf{Obf}(1^\lambda,C_0[\mathsf{aCT}_0,\{\mathsf{aDK}_{\ell,1}\}_{\ell \in [n-1]}],\mu,\sigma_0) \end{split}$$

and outputs  $\mathsf{CT}_{\mathsf{lab}} := \widetilde{C}_0$ . The circuit  $C_0$  is described in Fig. 5.

 $\mathsf{AKeyGen}(\mathsf{SK}_i, \mathsf{lab}, x_i)$  takes as input  $\mathsf{SK}_i$  for some  $i \in [n-1]$ , a label  $\mathsf{lab} \in \mathcal{L}$  and an attribute  $x_i \in \mathcal{X}$ . The algorithm samples  $\sigma_i \stackrel{\$}{\leftarrow} \mathcal{M}$ , runs

$$\begin{split} \mathsf{aCT}_i &\leftarrow \mathsf{aEnc}(\mathsf{aMPK}_i, \mathsf{lab}, x_i, \sigma_i) \\ \big\{ \mathsf{aDK}_{\ell, j} &\leftarrow \mathsf{aAKeyGen}(\mathsf{aSK}_{\ell, j}, \mathsf{lab}, x_i) \big\}_{(\ell, j) \in J_i} \\ & \widetilde{C}_i \leftarrow \mathsf{Obf}(1^{\lambda}, C_i[\mathsf{aCT}_i, \{\mathsf{aDK}_{\ell, i+1}\}_{\ell \in [i+1; n-1]}], \{\mathsf{aDK}_{\ell, i}\}_{\ell \in [0; i-1]}, \sigma_i) \end{split}$$

and outputs  $\mathsf{DK}_{\mathsf{lab},i} \coloneqq \widetilde{C}_i$  with  $C_i$  being described in Fig. 5.

 $\mathsf{PKeyGen}(\mathsf{MSK}, f)$  takes as input  $\mathsf{MSK}$  and a policy  $f \in \mathcal{F}$ , runs

 $\left\{ \mathsf{aDK}_{\ell,f} \leftarrow \mathsf{aPKeyGen}(\mathsf{aMSK}_{\ell}, \pi_{\ell}(f)) \right\}_{\ell \in [0;n-1]}$ ,

and outputs  $\mathsf{DK}_f \coloneqq \{\mathsf{a}\mathsf{DK}_{\ell,f}\}_{\ell \in [n]}$ .

$$\begin{split} \mathsf{Dec}(\mathsf{DK}_f,\{\mathsf{DK}_{\mathsf{lab},i}\}_{i\in[n-1]},\mathsf{CT}_{\mathsf{lab}}) \text{ takes a policy decryption key } \mathsf{DK}_f &= \{\mathsf{a}\mathsf{DK}_{\ell,f}\}_{\ell\in[0;n-1]},\\ \text{a set of attribute decryption keys } \{\mathsf{DK}_{\mathsf{lab},i} = \widetilde{C}_i\}_{i\in[n-1]} \text{ and a ciphertext } \mathsf{CT}_{\mathsf{lab}} = \widetilde{C}_0\\ \text{ as input. The algorithm outputs } \mu' \text{ computed as follows:} \end{split}$$

 $\mu' \leftarrow \mathsf{Eval}(\widetilde{C}_0, (\{\widetilde{C}_i\}_{i \in [n-1]}, \varnothing, \{\mathsf{aDK}_{\ell, f}\}_{\ell \in [0; n-1]})) \ .$ 

**Correctness.** Let  $\lambda, n \in \mathbb{N}$ ,  $\mathsf{lab} \in \mathcal{L}$ ,  $\mu \in \mathcal{M}$ ,  $f \in \mathcal{F}$  and  $x_0, \ldots, x_{n-1} \in \mathcal{X}$  such that  $f(x_0, \ldots, x_{n-1}) = 1$ . Furthermore, let  $\mathsf{DK}_f = \{\mathsf{aDK}_{\ell,f}\}_{\ell \in [0;n-1]}$ ,  $\{\mathsf{DK}_{\mathsf{lab},i} = \widetilde{C}_i\}_{i \in [n-1]}$  and  $\mathsf{CT}_{\mathsf{lab}} = \widetilde{C}_0$  be created as in Construction 9. To establish correctness, we prove the following statement.

**Lemma 10.** For  $x_0, \ldots, x_{n-1}$  such that  $f(x_0, \ldots, x_{n-1}) = 1$  and  $i \in [0; n-1]$ , we have

$$\mathsf{Eval}(\widetilde{C}_{i}, (\{\widetilde{C}_{\ell}\}_{\ell \in [i+1;n-1]}, \{\mathsf{aDK}_{\ell,j}\}_{\ell \in [i;n-1]}^{j \in [i]}, \{\mathsf{aDK}_{\ell,f}\}_{\ell \in [i;n-1]})) = \mu_{i}$$

where  $\mu_0 = \mu$  and  $\mu_i = \{aDK_{\ell,i}\}_{\ell \in [0;i-1]}$  for  $i \in [n-1]$ . In particular, correctness corresponds to the case i = 0.

Furthermore, due to the nested evaluations, the scheme is efficient for constant arity, *i.e.* n = O(1).

*Proof* (Lemma 10). We prove the lemma by induction over *i* from n-1 to 0.

**Base Case.** Let i = n - 1. By construction, we have

$$C_{n-1} \leftarrow \mathsf{Obf}(1^{\lambda}, C_{n-1}[\mathsf{aCT}_{n-1}, \varnothing], \{\mathsf{aDK}_{\ell, n-1}\}_{\ell \in [0; n-2]}, \sigma_{n-1}) ,$$

where  $\sigma_{n-1} \stackrel{\hspace{0.1em}{\leftarrow}\hspace{0.1em}}{\mathcal{M}}$  is the lock value of the obfuscated circuit. By definition, the circuit  $C_{n-1}[\mathsf{aCT}_{n-1}, \emptyset]$  takes as input  $(\emptyset, \{\mathsf{aDK}_{n-1,j}\}_{j \in [n-1]}, \{\mathsf{aDK}_{n-1,f}\})$  and outputs the result of the decryption

$$\sigma'_{n-1} \leftarrow \mathsf{aDec}(\mathsf{aDK}_{n-1,f}, \{\mathsf{aDK}_{n-1,j}\}_{j \in [n-1]}, \mathsf{aCT}_{n-1})$$

By the correctness of  $\mathsf{aFE}$ , we have  $\sigma_n = \sigma'_n$  and, by the correctness of  $\mathsf{LObf}$ , the evaluation of  $\widetilde{C}_{n-1}$  outputs  $\mu_{n-1} = \{\mathsf{aDK}_{\ell,n-1}\}_{\ell \in [0;n-2]}$ .

**Induction Step.** Let  $i \in [0; n-2]$ . We show that, if

$$\mathsf{Eval}(\widetilde{C}_{k}, (\{\widetilde{C}_{\ell}\}_{\ell \in [k+1;n-1]}, \{\mathsf{aDK}_{\ell,j}\}_{\ell \in [k;n-1]}^{j \in [k]}, \{\mathsf{aDK}_{\ell,f}\}_{\ell \in [k;n-1]})) = \mu_{k}$$

for all  $k \in [i+1; n-1]$ , then

$$\mathsf{Eval}(\widetilde{C}_i, (\{\widetilde{C}_\ell\}_{\ell \in [i+1;n-1]}, \{\mathsf{aCT}_{\ell,j}\}_{\ell \in [i;n-1]}^{j \in [i]}, \{\mathsf{aDK}_{\ell,f}\}_{\ell \in [i;n-1]})) = \mu_i \ .$$

By construction,  $\tilde{C}_i$  is an obfuscation of  $C_i[\mathsf{aCT}_i, \{\mathsf{aDK}_{\ell,i+1}\}_{\ell \in [i+1;n-1]}]$  under the lock value  $\sigma_i$  for the message  $\mu_i$ . From the induction hypothesis, we obtain that the for loop in the definition of  $C_i[\mathsf{aCT}_i, \{\mathsf{aDK}_{\ell,i+1}\}_{\ell \in [i+1;n-1]}]$  correctly recovers  $\{\mathsf{aDK}_{i,j}\}_{j \in [i+1;n-1]}$ . Given in addition  $\mathsf{aCT}_i$  hardwired in the description of the circuit and  $\{\mathsf{aDK}_{i,j}\}_{j \in [i]}$  from the input, the circuit returns the decryption result

$$\sigma'_i \leftarrow \mathsf{aDec}(\mathsf{aDK}_{i,f}, \{\mathsf{aDK}_{i,j}\}_{j \in [n-1]}, \mathsf{aCT}_i)$$
.

The correctness of aFE implies  $\sigma_i = \sigma'_i$ , and we can conclude from the correctness of LObf that the evaluation of the obfuscated circuit indeed yields  $\mu_i$  as desired.

**Security.** We show that the compiler preserves the security level of the underlying MC-ABE.

**Proposition 11.** Let  $xxx \in \{sel, adap\}$  and  $yyy \in \{rep, norep\}$ . If aFE is xxx-yyy-secure and LObf is secure, then the MC-PE scheme pFE in Construction 9 is also xxx-yyy-secure.

*Proof.* The proof proceeds via a sequence of hybrid games  $G_{0.1}^b, G_{1.0}^b, G_{1.1}^b, G_{2.0}^b, G_{2.1}^b, \ldots$ ,  $G_{n.0}^b, G_{n.1}^b$  with  $G_{0.1}^b = \mathbf{Exp}_{\mathsf{pFE},\mathcal{A}}^{\mathsf{mc-pe-b}}(1^{\lambda})$  for  $b \in \{0,1\}$  and  $G_{n.1}^0 \equiv G_{n.1}^1$ . We argue that  $G_{\kappa-1.1}^b \approx_c G_{\kappa,0}^b$  and  $G_{\kappa,0}^b \approx_c G_{\kappa,1}^b$  for all  $\kappa \in [n]$  and  $b \in \{0,1\}$ . Then the proposition follows via a hybrid argument. Modifications between consecutive games are highlighted using boxes.

Game  $G_{0,1}^b$  for  $b \in \{0,1\}$ : This game is the same as  $\operatorname{Exp}_{\mathsf{pFE},\mathcal{A}}^{\mathsf{mc-pe-}b}(1^{\lambda})$ .

**Game**  $G_{\kappa,0}^{b}$  for  $\kappa \in [n-1]$  and  $b \in \{0,1\}$ : Let  $i = n - \kappa$ . This game is the same as  $G_{\kappa-1,1}^{b}$  except that in replies to queries of the form  $\mathcal{O}\mathsf{AKeyGen}(i,\mathsf{lab},x_{i}^{0},x_{i}^{1})$  with  $x_{i}^{0} \neq x_{i}^{1}$ , the challenger computes

 $aCT_i \leftarrow aEnc(aMPK_i, lab, x_i^b, 0^m)$ ,

as opposed to  $\mathsf{aCT}_i \leftarrow \mathsf{aEnc}(\mathsf{aMPK}_i, \mathsf{lab}, x_i^b, \sigma_i)$  for  $\sigma_i \stackrel{\$}{\leftarrow} \mathcal{M}$ .

Claim 12. If aFE is secure, then  $\mathsf{G}^b_{\kappa-1.1} \approx_c \mathsf{G}^b_{\kappa.0}$ .

**Game**  $G_{\kappa,1}^b$  for  $\kappa \in [n-1]$  and  $b \in \{0,1\}$ : Let  $i = n - \kappa$ . The security of LObf guarantees the existence of an efficient simulator Sim. This game is the same as  $G_{\kappa,0}^b$  except that in replies to queries of the form  $\mathcal{O}\mathsf{AKeyGen}(i,\mathsf{lab},x_i^0,x_i^1)$  with  $x_i^0 \neq x_i^1$ , the challenger computes

 $\widetilde{C}_i \leftarrow \operatorname{Sim}(1^{\lambda}, 1^{|C_i[\mathsf{aCT}_i, \{\mathsf{aDK}_{\ell, i+1}\}_{\ell \in [i+1;n-1]}]|}, 1^{|\{\mathsf{aDK}_{\ell, i}\}_{\ell \in [0;i-1]}|}) \ ,$ 

instead of  $\widetilde{C}_i \leftarrow \mathsf{Obf}(1^{\lambda}, C_i[\mathsf{aCT}_i, \{\mathsf{aDK}_{\ell,i+1}\}_{\ell \in [i+1;n-1]}], \{\mathsf{aDK}_{\ell,i}\}_{\ell \in [0;i-1]}, \sigma_i).$ Claim 13. If LObf is secure, then  $\mathsf{G}^b_{\kappa,0} \approx_c \mathsf{G}^b_{\kappa,1}$ .

**Game**  $G_{n,0}^b$  for  $b \in \{0,1\}$ : This game is the same as  $G_{n-1,1}^b$  except that in replies to queries of the form  $\mathcal{O}\mathsf{Enc}(\mathsf{lab}, x_0^0, x_0^1, \mu^0, \mu^1)$  with  $x_0^0 \neq x_0^1$ , the challenger computes

 $\mathsf{aCT}_0 \leftarrow \mathsf{aEnc}(\mathsf{aMPK}_0, \mathsf{lab}, x_0^b, 0^m)$ ,

as opposed to  $\mathsf{aCT}_0 \leftarrow \mathsf{aEnc}(\mathsf{aMPK}_0, \mathsf{lab}, x_0^b, \sigma_0)$ .

Claim 14. If aFE is secure, then  $G_{n-1,1}^b \approx_c G_{n,0}^b$ .

The proof of Claim 14 is similar to that of Claim 12.

**Game**  $G_{n,1}^b$  for  $b \in \{0,1\}$ : This game is the same as  $G_{n,0}^b$  except that in replies to queries of the form  $\mathcal{O}\mathsf{Enc}(\mathsf{lab}, x_0^0, x_0^1, \mu^0, \mu^1)$  with  $x_0^0 \neq x_0^1$  or  $\mu^0 \neq \mu^1$ , the challenger computes

 $\widetilde{C}_0 \leftarrow \mathsf{Obf}(1^{\lambda}, 1^{|C_0[\mathsf{aCT}_0, \{\mathsf{aDK}_{\ell,1}\}_{\ell \in [n-1]}]|}, 1^{|\mu^b|}) \ ,$ 

as opposed to  $\widetilde{C}_0 \leftarrow \mathsf{Obf}(1^{\lambda}, C_0[\mathsf{aCT}_0, \{\mathsf{aDK}_{\ell,1}\}_{\ell \in [n-1]}], \mu^b, \sigma_0).$ Claim 15. If LObf is secure, then  $\mathsf{G}_{n,0}^b \approx_c \mathsf{G}_{n,1}^b$ .

The proof of Claim 15 is similar to that of Claim 13. It is not hard to see that  $G_{n.1}^0 \equiv G_{n.1}^1$  as  $|\mu^0| = |\mu^1|$  for all encryption queries  $\mathcal{O}\mathsf{Enc}(\mathsf{lab}, x_0^0, x_0^1, \mu^0, \mu^1)$  and, thus, all responses of the challenger are independent of the challenge bit *b*. This concludes the proof of Proposition 11.

We now turn to the claims.

*Proof* (Claim 12). Let  $i = n - \kappa$  as above. We show that if a PPT adversary  $\mathcal{A}$  distinguishes between  $\mathsf{G}^b_{\kappa-1,1}$  and  $\mathsf{G}^b_{\kappa,0}$  with non-negligible probability, then there exists a PPT adversary  $\mathcal{B}$  that can break the security of the *i*-th aFE instance using  $\mathcal{A}$ . The reduction is as follows:

• Initialization. Upon  $\mathcal{A}$  calling Initialize $(1^{\lambda})$ ,  $\mathcal{B}$  calls the initialization procedure  $\mathsf{aMPK}_i \leftarrow \mathsf{Initialize}(1^{\lambda})$  of its own game  $\mathbf{Exp}_{\mathsf{aFE},\mathcal{B}}^{\mathsf{mc-abe-}\beta}(1^{\lambda})$  and generates keys for n-1 aFE instances by running

 $\left\{(\mathsf{aMPK}_{\ell},\mathsf{aMSK}_{\ell},\{\mathsf{aSK}_{\ell,i}\}_{i\in[n-1]}) \leftarrow \mathsf{aSetup}(1^{\lambda})\right\}_{\ell\in[0;n-1]\setminus\{i\}} \ .$ 

- Corruption Queries. Upon  $\mathcal{A}$  calling  $\mathcal{O}\mathsf{Corrupt}(k)$  for  $k \in [0; n-1]$ ,  $\mathcal{B}$  first checks if  $k \neq i$ , in which case it submits a query  $\mathsf{aSK}_{i,\rho_i(k)} \leftarrow \mathcal{O}\mathsf{Corrupt}(\rho_i(k))$  to its own challenger. Then it sends  $\mathsf{SK}_k = (\mathsf{aMPK}_k, \{\mathsf{aSK}_{\ell,j}\}_{(\ell,j)\in J_k})$  to  $\mathcal{A}$ .
- Encryption Queries. Upon  $\mathcal{A}$  calling  $\mathcal{O}\mathsf{Enc}(\mathsf{lab}, x_0^0, x_0^1, \mu^0, \mu^1)$ ,  $\mathcal{B}$  samples  $\sigma_0 \stackrel{\$}{\leftarrow} \mathcal{M}$ , submits a query  $\mathsf{aDK}_{i,1} \leftarrow \mathcal{O}\mathsf{AKeyGen}(1, \mathsf{lab}, x_0^b)$  to its own game  $\mathbf{Exp}_{\mathsf{aFE},\mathcal{B}}^{\mathsf{mc-abe},\beta}(1^{\lambda})$  and returns  $\mathsf{CT}_{\mathsf{lab}} = \widetilde{C}_0$  computed as follows

$$\begin{split} \mathsf{aCT}_0 &\leftarrow \mathsf{aEnc}(\mathsf{aMPK}_0, \mathsf{lab}, x_0^b, \sigma_0) \\ \big\{ \mathsf{aDK}_{\ell,1} &\leftarrow \mathsf{aAKeyGen}(\mathsf{aSK}_{\ell,1}, \mathsf{lab}, x_0^b) \big\}_{\ell \in [n-1] \setminus \{i\}} \\ & \widetilde{C}_0 \leftarrow \mathsf{Obf}(1^\lambda, C_0[\mathsf{aCT}_0, \{\mathsf{aDK}_{\ell,1}\}_{\ell \in [n-1]}], \mu^b, \sigma_0) \end{split}$$

- Attribute Key Generation Queries. Upon  $\mathcal{A}$  calling  $\mathcal{O}\mathsf{AKeyGen}(k, \mathsf{lab}, x_k^0, x_k^1)$  for  $k \in [n-1], \mathcal{B}$  behaves according to the following case distinction:
  - If  $k \in [i-1]$ , then  $\mathcal{B}$  samples  $\sigma_k \stackrel{\$}{\leftarrow} \mathcal{M}$ , submits a query  $\mathsf{aDK}_{i,k+1} \leftarrow \mathcal{O}\mathsf{AKeyGen}(k+1,\mathsf{lab},x^b_k)$  to its own game  $\mathbf{Exp}_{\mathsf{aFE},\mathcal{B}}^{\mathsf{mc-abe},\beta}(1^{\lambda})$  and returns  $\mathsf{DK}_{\mathsf{lab},k} = \widetilde{C}_k$  computed as follows

$$\begin{split} \mathsf{aCT}_k &\leftarrow \mathsf{aEnc}(\mathsf{aMPK}_k, \mathsf{lab}, x_k^b, \sigma_k) \\ \big\{\mathsf{aDK}_{\ell,j} &\leftarrow \mathsf{aAKeyGen}(\mathsf{aSK}_{\ell,j}, \mathsf{lab}, x_k^b)\big\}_{(\ell,j) \in J_k \setminus \{(i,k+1)\}} \\ & \widetilde{C}_k \leftarrow \mathsf{Obf}(1^\lambda, C_k[\mathsf{aCT}_k, \{\mathsf{aDK}_{\ell,k+1}\}_{\ell \in [k+1;n-1]}], \{\mathsf{aDK}_{\ell,k}\}_{\ell \in [0;k-1]}, \sigma_k). \end{split}$$

• If k = i, then  $\mathcal{B}$  samples  $\sigma_i \stackrel{\hspace{0.1em}\mathsf{\scriptscriptstyle\$}}{\leftarrow} \mathcal{M}$ , sets

$$(\mathbf{v}_0, \mathbf{v}_1) = \begin{cases} (\sigma_i, 0^m) & \text{if } x_i^0 \neq x_i^1 \\ (\sigma_i, \sigma_i) & \text{if } x_i^0 = x_i^1 \end{cases},$$

submits a query  $\mathsf{aCT}_i \leftarrow \mathcal{O}\mathsf{Enc}(\mathsf{lab}, x_i^b, \mathbf{v}_0, \mathbf{v}_1)$  to its own challenger  $\mathbf{Exp}_{\mathsf{aFE}, \mathcal{B}}^{\mathsf{mc-abe}, \beta}(1^{\lambda})$ and returns  $\mathsf{DK}_{\mathsf{lab}, i} = \widetilde{C}_i$  computed as follows

$$\begin{split} \big\{\mathsf{a}\mathsf{D}\mathsf{K}_{\ell,j} \leftarrow \mathsf{a}\mathsf{A}\mathsf{KeyGen}(\mathsf{a}\mathsf{S}\mathsf{K}_{\ell,j},\mathsf{Iab},x_i^b)\big\}_{(\ell,j)\in J_i} \\ & \widetilde{C}_i \leftarrow \mathsf{Obf}(1^\lambda,C_i[\mathsf{a}\mathsf{C}\mathsf{T}_i,\{\mathsf{a}\mathsf{D}\mathsf{K}_{\ell,i+1}\}_{\ell\in[i+1;n-1]}],\{\mathsf{a}\mathsf{D}\mathsf{K}_{\ell,i}\}_{\ell\in[0;i-1]},\sigma_i) \end{split}$$

• If  $k \in [i + 1; n - 1]$ , then  $\mathcal{B}$  submits a query  $\mathsf{aDK}_{i,k} \leftarrow \mathcal{O}\mathsf{AKeyGen}(k, \mathsf{lab}, x_k^b)$  to its own game  $\mathsf{Exp}_{\mathsf{aFE},\mathcal{B}}^{\mathsf{mc-abe},\beta}(1^\lambda)$  and generates decryption keys  $\mathsf{aDK}_{\ell,j} \leftarrow \mathsf{aAKeyGen}(\mathsf{aSK}_{\ell,j}, \mathsf{lab}, x_k^b)$  for all  $(\ell, j) \in J_k \setminus \{(i, k)\}$  by itself. If  $x_k^0 \neq x_k^1$ , it computes

$$\begin{split} \mathsf{aCT}_k &\leftarrow \mathsf{aEnc}(\mathsf{aMPK}_k, \mathsf{lab}, x_k^b, 0^m) \\ & \widetilde{C}_k \leftarrow \mathsf{Sim}(1^{\lambda}, 1^{|C_k[\mathsf{aCT}_k, \{\mathsf{aDK}_{\ell,k+1}\}_{\ell \in [k+1;n-1]}]|}, 1^{|\{\mathsf{aDK}_{\ell,k}\}_{\ell \in [0;k-1]}|}) \end{split}$$

Otherwise,  $\mathcal{B}$  samples  $\sigma_k \stackrel{\hspace{0.1em}\mathsf{\scriptscriptstyle\$}}{\leftarrow} \mathcal{M}$  and computes

$$\begin{split} \mathsf{aCT}_k &\leftarrow \mathsf{aEnc}(\mathsf{aMPK}_k, \mathsf{lab}, x_k^b, \sigma_k) \\ & \widetilde{C}_k \leftarrow \mathsf{Obf}(1^\lambda, C_k[\mathsf{aCT}_k, \{\mathsf{aDK}_{\ell,k+1}\}_{\ell \in [k+1;n-1]}], \{\mathsf{aDK}_{\ell,k}\}_{\ell \in [0;k-1]}, \sigma_k). \end{split}$$

Finally,  $\mathcal{B}$  sends  $\mathsf{DK}_{\mathsf{lab},k} = \widetilde{C}_k$  to  $\mathcal{A}$ .

• Policy Key Generation Queries. Upon  $\mathcal{A}$  calling  $\mathsf{PKeyGen}(f)$ ,  $\mathcal{B}$  submits a query  $\mathsf{aDK}_{i,f} \leftarrow \mathcal{O}\mathsf{PKeyGen}(\pi_i(f))$  to its own game  $\mathbf{Exp}_{\mathsf{aFE},\mathcal{B}}^{\mathsf{mc-abe-}\beta}(1^{\lambda})$ , runs

 $\left\{ \mathsf{aDK}_{\ell,f} \leftarrow \mathsf{aPKeyGen}(\mathsf{aMSK}_{\ell}, \pi_{\ell}(f)) \right\}_{\ell \in [0; n-1] \setminus \{i\}}$ ,

and sends  $\mathsf{DK}_f = \{\mathsf{a}\mathsf{DK}_{\ell,f}\}_{\ell \in [0:n-1]}$  to  $\mathcal{A}$ .

• Finalization. Upon  $\mathcal{A}$  calling Finalize(b'),  $\mathcal{B}$  calls the finalization procedure Finalize(b') of its own game  $\operatorname{Exp}_{\mathsf{aFE},\mathcal{B}}^{\mathsf{mc-abe}-\beta}(1^{\lambda})$ .

We observe that if  $\mathcal{A}$  is an admissible adversary in  $\mathbf{Exp}_{\mathsf{pFE},\mathcal{A}}^{\mathsf{mc-pe-b}}(1^{\lambda})$ , then  $\mathcal{B}$  is admissible in  $\mathbf{Exp}_{\mathsf{aFE},\mathcal{B}}^{\mathsf{mc-abe},\beta}(1^{\lambda})$ . We further observe that if  $\mathcal{B}$ 's challenger returns an encryption of  $\mathbf{v}_0$ (*i.e.*  $\beta = 0$ ), then  $\mathcal{B}$  simulates  $\mathsf{G}_{\kappa-1,1}^b$ ; and if the challenger returns an encryption of  $\mathbf{v}_1$ (*i.e.*  $\beta = 1$ ), then  $\mathcal{B}$  simulates  $\mathsf{G}_{\kappa,0}^b$ .

*Proof* (Claim 13). Let  $i = n - \kappa$  as above. We show that if a PPT adversary  $\mathcal{A}$  distinguishes between  $\mathsf{G}_{\kappa,0}$  and  $\mathsf{G}_{\kappa,1}$  with non-negligible probability, then there exists a PPT adversary  $\mathcal{B}$  that can break the strong security of LObf using  $\mathcal{A}$ . The reduction is as follows:

• Initialization. Upon  $\mathcal{A}$  calling Initialize $(1^{\lambda})$ ,  $\mathcal{B}$  generates n aFE instances

 $\left\{(\mathsf{aMPK}_\ell,\mathsf{aMSK}_\ell,\{\mathsf{aSK}_{\ell,i}\}_{i\in[n-1]})\leftarrow\mathsf{aSetup}(1^\lambda)\right\}_{\ell\in[0;n-1]}\ .$ 

- Corruption Queries. Upon  $\mathcal{A}$  calling  $\mathcal{O}$ Corrupt(k) for  $k \in [0; n-1]$ ,  $\mathcal{B}$  sends  $\mathsf{SK}_i = (\mathsf{aMPK}_i, \{\mathsf{aSK}_{\ell,j}\}_{(\ell,j)\in J_i})$  to  $\mathcal{A}$ .
- Encryption Queries. Upon  $\mathcal{A}$  calling  $\mathcal{O}\mathsf{Enc}(\mathsf{lab}, x_0^0, x_0^1, \mu^0, \mu^1)$ ,  $\mathcal{B}$  samples  $\sigma_0 \notin \mathcal{M}$ and returns  $\mathsf{CT}_{\mathsf{lab}} = \widetilde{C}_0$  computed as follows:

$$\begin{split} & \mathsf{aCT}_0 \leftarrow \mathsf{aEnc}(\mathsf{aMPK}_0,\mathsf{lab},x_0^b,\sigma_0) \\ & \left\{ \mathsf{aDK}_{\ell,1} \leftarrow \mathsf{aAKeyGen}(\mathsf{aSK}_{\ell,1},\mathsf{lab},x_0^b) \right\}_{\ell \in [n-1]} \\ & \widetilde{C}_0 \leftarrow \mathsf{Obf}(1^\lambda,C_0[\mathsf{aCT}_0,\{\mathsf{aDK}_{\ell,1}\}_{\ell \in [n-1]}],\mu,\sigma_0) \end{split}$$

- Attribute Key Generation Queries. Upon  $\mathcal{A}$  calling  $\mathcal{O}\mathsf{AKeyGen}(k, \mathsf{lab}, x_k^0, x_k^1)$  for  $k \in [n-1], \mathcal{B}$  behaves according to the following case distinction:
  - If  $k \in [i-1]$ , then  $\mathcal{B}$  samples  $\sigma_k \stackrel{\$}{\leftarrow} \mathcal{M}$  and returns  $\mathsf{CT}_k = C_k$  computed as follows:

$$\begin{split} &\mathsf{aCT}_k \leftarrow \mathsf{aEnc}(\mathsf{aMPK}_k, \mathsf{lab}, x_k^b, \sigma_k) \\ \big\{\mathsf{aDK}_{\ell,j} \leftarrow \mathsf{aAKeyGen}(\mathsf{aSK}_{\ell,j}, \mathsf{lab}, x_k^b)\big\}_{(\ell,j) \in J_k} \\ & \widetilde{C}_k \leftarrow \mathsf{Obf}(1^\lambda, C_k[\mathsf{aCT}_k, \{\mathsf{aDK}_{\ell,k+1}\}_{\ell \in [k+1;n-1]}], \{\mathsf{aDK}_{\ell,k}\}_{\ell \in [0;k-1]}, \sigma_k). \end{split}$$

• If k = i and  $x_k^0 \neq x_k^1$ , then  $\mathcal{B}$  runs

$$\begin{aligned} \mathsf{aCT}_i &\leftarrow \mathsf{aEnc}(\mathsf{aMPK}_i, \mathsf{lab}, x_i^b, 0^m) \\ \big\{\mathsf{aDK}_{\ell, j} &\leftarrow \mathsf{aAKeyGen}(\mathsf{aSK}_{\ell, j}, \mathsf{lab}, x_i^b)\big\}_{(\ell, j) \in J_i} \end{aligned}$$

and returns  $\widetilde{C}_i$  obtained from its own LObf challenger

$$C_i \leftarrow \mathsf{LObf}.\mathcal{O}\mathsf{Obf}^\beta(C_i[\mathsf{aCT}_i, \{\mathsf{aDK}_{\ell,i+1}\}_{\ell \in [i+1;n-1]}], \{\mathsf{aDK}_{\ell,i}\}_{\ell \in [0;i-1]}) .$$

Otherwise, if k = i and  $x_k^0 = x_k^1$ , then  $\mathcal{B}$  samples  $\sigma_i \stackrel{s}{\leftarrow} \mathcal{M}$  and returns  $\mathsf{aCT}_i = \widetilde{C}_i$  computed as follows:

$$\begin{split} \mathsf{aCT}_i &\leftarrow \mathsf{aEnc}(\mathsf{aMPK}_i, \mathsf{lab}, x_i^b, \sigma_i) \\ \big\{ \mathsf{aDK}_{\ell,j} &\leftarrow \mathsf{aAKeyGen}(\mathsf{aSK}_{\ell,j}, \mathsf{lab}, x_i^b) \big\}_{(\ell,j) \in J_i} \\ & \widetilde{C}_i \leftarrow \mathsf{Obf}(1^\lambda, C_i[\mathsf{aCT}_i, \{\mathsf{aDK}_{\ell,i+1}\}_{\ell \in [i+1;n-1]}], \{\mathsf{aDK}_{\ell,i}\}_{\ell \in [0;i-1]}, \sigma_i) \end{split}$$

• If  $k \in [i+1;n]$  and  $x_k^0 \neq x_k^1$ , then  $\mathcal{B}$  sends  $\mathsf{aCT}_k = \widetilde{C}_k$  to  $\mathcal{A}$  computed as follows:

$$\begin{aligned} \mathsf{aCT}_k &\leftarrow \mathsf{aEnc}(\mathsf{aMPK}_k, \mathsf{lab}, x_k^b, 0^m) \\ \left\{ \mathsf{aDK}_{\ell,j} &\leftarrow \mathsf{aAKeyGen}(\mathsf{aSK}_{\ell,j}, \mathsf{lab}, x_k^b) \right\}_{(\ell,j) \in J_k} \\ & \widetilde{C}_k \leftarrow \mathsf{Sim}(1^\lambda, 1^{|C_k[\mathsf{aCT}_k, \{\mathsf{aDK}_{\ell,k+1}\}_{\ell \in [k+1;n-1]}]|}, 1^{|\{\mathsf{aDK}_{\ell,k}\}_{\ell \in [0;k-1]}|}) \end{aligned}$$

Otherwise, if  $k \in [i+1; n]$  and  $x_k^0 = x_k^1$ , it samples  $\sigma_k \stackrel{\text{\tiny{\&}}}{\leftarrow} \mathcal{M}$  and returns  $\mathsf{aCT}_k = \widetilde{C}_k$  computed as follows:

$$\begin{split} &\mathsf{aCT}_k \leftarrow \mathsf{aEnc}(\mathsf{aMPK}_k, \mathsf{lab}, x^b_k, \sigma_k) \\ \big\{\mathsf{aDK}_{\ell,j} \leftarrow \mathsf{aAKeyGen}(\mathsf{aSK}_{\ell,j}, \mathsf{lab}, x^b_k)\big\}_{(\ell,j) \in J_k} \\ & \widetilde{C}_k \leftarrow \mathsf{Obf}(1^\lambda, C_k[\mathsf{aCT}_k, \{\mathsf{aDK}_{\ell,k+1}\}_{\ell \in [k+1;n-1]}], \{\mathsf{aDK}_{\ell,k}\}_{\ell \in [0;k-1]}, \sigma_k) \end{split}$$

• Policy Key Generation Queries. Upon  $\mathcal{A}$  calling  $\mathcal{O}\mathsf{PKeyGen}(f)$ ,  $\mathcal{B}$  runs

 $\left\{ \mathsf{aDK}_{\ell,f} \leftarrow \mathsf{aPKeyGen}(\mathsf{aMSK}_{\ell}, f) \right\}_{\ell \in [0:n-1]}$ ,

and sends  $\mathsf{DK}_f = \{\mathsf{a}\mathsf{DK}_{\ell,\pi_\ell(f)}\}_{\ell\in[0;n-1]}$  to  $\mathcal{A}$ .

• Finalization. Upon  $\mathcal{A}$  calling Finalize(b'),  $\mathcal{B}$  sends b' to the LObf challenger.

We observe that if the challenge bit  $\beta$  of the LObf challenger is 0, *i.e.* the LObf challenger returns a real obfuscation of  $C_i[\{\mathsf{aCT}_{\ell,i}\}_{\ell \in [i;n]}]$ , then  $\mathcal{B}$  simulates  $\mathsf{G}_{\kappa,0}$ . Conversely, if  $\beta = 1$ , then the LObf challenger returns a simulated obfuscation and  $\mathcal{B}$  simulates  $\mathsf{G}_{\kappa,1}$ .

## Acknowledgments

This work was supported in part by the France 2030 ANR Project ANR-22-PECY-003 SecureCompute.

### References

- [ABG19] Michel Abdalla, Fabrice Benhamouda, and Romain Gay. From single-input to multi-client inner-product functional encryption. In Steven D. Galbraith and Shiho Moriai, editors, ASIACRYPT 2019, Part III, volume 11923 of LNCS, pages 552–582. Springer, Heidelberg, December 2019. doi:10.1007/978-3 -030-34618-8\_19.
- [ABKW19] Michel Abdalla, Fabrice Benhamouda, Markulf Kohlweiss, and Hendrik Waldner. Decentralizing inner-product functional encryption. In Dongdai Lin and Kazue Sako, editors, PKC 2019, Part II, volume 11443 of LNCS, pages 128– 157. Springer, Heidelberg, April 2019. doi:10.1007/978-3-030-17259-6\_5.
- [ACF<sup>+</sup>18] Michel Abdalla, Dario Catalano, Dario Fiore, Romain Gay, and Bogdan Ursu. Multi-input functional encryption for inner products: Function-hiding realizations and constructions without pairings. In Hovav Shacham and Alexandra Boldyreva, editors, CRYPTO 2018, Part I, volume 10991 of LNCS, pages 597–627. Springer, Heidelberg, August 2018. doi:10.1007/978-3-319 -96884-1\_20.
- [AG21] Miguel Ambrona and Romain Gay. Multi-authority ABE, revisited. Cryptology ePrint Archive, Report 2021/1381, 2021. https://eprint.iacr.org/20 21/1381.
- [AG23] Miguel Ambrona and Romain Gay. Multi-authority ABE for non-monotonic access structures. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC 2023, Part II*, volume 13941 of *LNCS*, pages 306–335. Springer, Heidelberg, May 2023. doi:10.1007/978-3-031-31371-4\_11.
- [AGRW17] Michel Abdalla, Romain Gay, Mariana Raykova, and Hoeteck Wee. Multiinput inner-product functional encryption from pairings. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 601–626. Springer, Heidelberg, April / May 2017. doi:10.1007/978-3-319-56620-7\_21.
- [AGT21] Shweta Agrawal, Rishab Goyal, and Junichi Tomida. Multi-input quadratic functional encryption from pairings. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part IV*, volume 12828 of *LNCS*, pages 208–238, Virtual Event, August 2021. Springer, Heidelberg. doi:10.1007/978-3-030-84259-8\_8.
- [AGT22] Shweta Agrawal, Rishab Goyal, and Junichi Tomida. Multi-input quadratic functional encryption: Stronger security, broader functionality. In Eike Kiltz and Vinod Vaikuntanathan, editors, TCC 2022, Part I, volume 13747 of LNCS, pages 711–740. Springer, Heidelberg, November 2022. doi:10.1007/ 978-3-031-22318-1\_25.
- [AJ15] Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, CRYPTO 2015, Part I, volume 9215 of LNCS, pages 308– 326. Springer, Heidelberg, August 2015. doi:10.1007/978-3-662-47989-6 \_15.
- [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. doi: 10.1007/978-3-662-53015-3\_12.

- [ARYY23] Shweta Agrawal, Mélissa Rossi, Anshu Yadav, and Shota Yamada. Constant input attribute based (and predicate) encryption from evasive and tensor LWE. In Helena Handschuh and Anna Lysyanskaya, editors, CRYPTO 2023, Part IV, volume 14084 of LNCS, pages 532–564. Springer, Heidelberg, August 2023. doi:10.1007/978-3-031-38551-3\_17.
- [ATY23] Shweta Agrawal, Junichi Tomida, and Anshu Yadav. Attribute-based multiinput FE (and more) for attribute-weighted sums. In Helena Handschuh and Anna Lysyanskaya, editors, CRYPTO 2023, Part IV, volume 14084 of LNCS, pages 464–497. Springer, Heidelberg, August 2023. doi:10.1007/978-3-031 -38551-3\_15.
- [AWY20] Shweta Agrawal, Daniel Wichs, and Shota Yamada. Optimal broadcast encryption from LWE and pairings in the standard model. In Rafael Pass and Krzysztof Pietrzak, editors, TCC 2020, Part I, volume 12550 of LNCS, pages 149–178. Springer, Heidelberg, November 2020. doi:10.1007/978-3 -030-64375-1\_6.
- [AYY22] Shweta Agrawal, Anshu Yadav, and Shota Yamada. Multi-input attribute based encryption and predicate encryption. In Yevgeniy Dodis and Thomas Shrimpton, editors, CRYPTO 2022, Part I, volume 13507 of LNCS, pages 590–621. Springer, Heidelberg, August 2022. doi:10.1007/978-3-031-158 02-5\_21.
- [Bei96] Amos Beimel. Secure Schemes for Secret Sharing and Key Distribution. PhD thesis, Technion - Israel Institute of Technology, Haifa, Israel, 1996. https://www.cs.bgu.ac.il/~beimel/Papers/thesis.pdf.
- [BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 1–18. Springer, Heidelberg, August 2001. doi:10.1007/3-540-44647-8\_1.
- [BJK<sup>+</sup>18] Zvika Brakerski, Aayush Jain, Ilan Komargodski, Alain Passelègue, and Daniel Wichs. Non-trivial witness encryption and null-iO from standard assumptions. In Dario Catalano and Roberto De Prisco, editors, SCN 18, volume 11035 of LNCS, pages 425–441. Springer, Heidelberg, September 2018. doi:10.1007/978-3-319-98113-0\_23.
- [BL90] Josh Cohen Benaloh and Jerry Leichter. Generalized secret sharing and monotone functions. In Shafi Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 27–35. Springer, Heidelberg, August 1990. doi:10.1007/0-387 -34799-2\_3.
- [BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, TCC 2011, volume 6597 of LNCS, pages 253–273. Springer, Heidelberg, March 2011. doi:10.1007/978-3-642 -19571-6\_16.
- [BW07] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In Salil P. Vadhan, editor, TCC 2007, volume 4392 of LNCS, pages 535–554. Springer, Heidelberg, February 2007. doi:10.1007/978-3-5 40-70936-7\_29.
- [CDG<sup>+</sup>18] Jérémy Chotard, Edouard Dufour Sans, Romain Gay, Duong Hieu Phan, and David Pointcheval. Decentralized multi-client functional encryption for inner

product. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018*, *Part II*, volume 11273 of *LNCS*, pages 703–732. Springer, Heidelberg, December 2018. doi:10.1007/978-3-030-03329-3\_24.

- [CDSG<sup>+</sup>20] Jérémy Chotard, Edouard Dufour-Sans, Romain Gay, Duong Hieu Phan, and David Pointcheval. Dynamic decentralized functional encryption. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 747–775. Springer, Heidelberg, August 2020. doi:10.1007/978-3-030-56784-2\_25.
- [CLL<sup>+</sup>13] Jie Chen, Hoon Wei Lim, San Ling, Huaxiong Wang, and Hoeteck Wee. Shorter IBE and signatures via asymmetric pairings. In Michel Abdalla and Tanja Lange, editors, *PAIRING 2012*, volume 7708 of *LNCS*, pages 122–140. Springer, Heidelberg, May 2013. doi:10.1007/978-3-642-36334-4\_8.
- [DOT18] Pratish Datta, Tatsuaki Okamoto, and Junichi Tomida. Full-hiding (unbounded) multi-input inner product functional encryption from the k-Linear assumption. In Michel Abdalla and Ricardo Dahab, editors, PKC 2018, Part II, volume 10770 of LNCS, pages 245–277. Springer, Heidelberg, March 2018. doi:10.1007/978-3-319-76581-5\_9.
- [EHK<sup>+</sup>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. doi:10.1007/978-3-642-400 84-1\_8.
- [FFMV23] Danilo Francati, Daniele Friolo, Giulio Malavolta, and Daniele Venturi. Multikey and multi-input predicate encryption from learning with errors. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part III*, volume 14006 of *LNCS*, pages 573–604. Springer, Heidelberg, April 2023. doi:10.1007/978-3-031-30620-4\_19.
- [GGG<sup>+</sup>14] Shafi Goldwasser, S. Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. Multiinput functional encryption. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 578–602. Springer, Heidelberg, May 2014. doi:10.1007/978-3-642-55220-5\_32.
- [GGH<sup>+</sup>13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In 54th FOCS, pages 40–49. IEEE Computer Society Press, October 2013. doi:10.1109/F0CS.2013.13.
- [GKVW20] Rishab Goyal, Venkata Koppula, Satyanarayana Vusirikala, and Brent Waters. On perfect correctness in (lockable) obfuscation. In Rafael Pass and Krzysztof Pietrzak, editors, TCC 2020, Part I, volume 12550 of LNCS, pages 229–259. Springer, Heidelberg, November 2020. doi:10.1007/978-3-030-64375-1\_9.
- [GKW17] Rishab Goyal, Venkata Koppula, and Brent Waters. Lockable obfuscation. In Chris Umans, editor, 58th FOCS, pages 612–621. IEEE Computer Society Press, October 2017. doi:10.1109/F0CS.2017.62.
- [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. doi:10.1145/1180405.1180418.

- [GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from LWE. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 503–523. Springer, Heidelberg, August 2015. doi:10.1007/978-3-662-480 00-7\_25.
- [JR17] Charanjit S. Jutla and Arnab Roy. Shorter quasi-adaptive NIZK proofs for linear subspaces. *Journal of Cryptology*, 30(4):1116–1156, October 2017. doi:10.1007/s00145-016-9243-7.
- [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. doi:10.1007/978-3-319-63688-7\_20.
- [LL20a] Huijia Lin and Ji Luo. Compact adaptively secure ABE from k-Lin: Beyond NC<sup>1</sup> and towards NL. In Anne Canteaut and Yuval Ishai, editors, EURO-CRYPT 2020, Part III, volume 12107 of LNCS, pages 247–277. Springer, Heidelberg, May 2020. doi:10.1007/978-3-030-45727-3\_9.
- [LL20b] Huijia Lin and Ji Luo. Succinct and adaptively secure ABE for ABP from k-Lin. In Shiho Moriai and Huaxiong Wang, editors, ASIACRYPT 2020, Part III, volume 12493 of LNCS, pages 437–466. Springer, Heidelberg, December 2020. doi:10.1007/978-3-030-64840-4\_15.
- [LLL22] Hanjun Li, Huijia Lin, and Ji Luo. ABE for circuits with constant-size secret keys and adaptive security. In Eike Kiltz and Vinod Vaikuntanathan, editors, TCC 2022, Part I, volume 13747 of LNCS, pages 680–710. Springer, Heidelberg, November 2022. doi:10.1007/978-3-031-22318-1\_24.
- [LT19] Benoît Libert and Radu Titiu. Multi-client functional encryption for linear functions in the standard model from LWE. In Steven D. Galbraith and Shiho Moriai, editors, ASIACRYPT 2019, Part III, volume 11923 of LNCS, pages 520–551. Springer, Heidelberg, December 2019. doi:10.1007/978-3-030-3 4618-8\_18.
- [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. doi:10.1109/FOCS.2016.11.
- [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. doi:10.1007/978-3-642-11799-2\_27.
- [LW11] Allison B. Lewko and Brent Waters. Decentralizing attribute-based encryption. In Kenneth G. Paterson, editor, EUROCRYPT 2011, volume 6632 of LNCS, pages 568–588. Springer, Heidelberg, May 2011. doi:10.1007/978-3-642-2 0465-4\_31.

- [NPP22] Ky Nguyen, Duong Hieu Phan, and David Pointcheval. Multi-client functional encryption with fine-grained access control. In Shweta Agrawal and Dongdai Lin, editors, ASIACRYPT 2022, Part I, volume 13791 of LNCS, pages 95–125. Springer, Heidelberg, December 2022. doi:10.1007/978-3-031-22963-3\_4.
- [PS25] David Pointcheval and Robert Schädlich. Multi-client attribute-based and predicate encryption from standard assumptions. In Elette Boyle and Mohammad Mahmoody, editors, *Theory of Cryptography*, pages 31–64, Cham, 2025. Springer Nature Switzerland. doi:10.1007/978-3-031-78020-2\_2.
- [SBC<sup>+</sup>07] Elaine Shi, John Bethencourt, Hubert T.-H. Chan, Dawn Xiaodong Song, and Adrian Perrig. Multi-dimensional range query over encrypted data. In 2007 IEEE Symposium on Security and Privacy, pages 350–364. IEEE Computer Society Press, May 2007. doi:10.1109/SP.2007.29.
- [SW05] Amit Sahai and Brent R. Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, EUROCRYPT 2005, volume 3494 of LNCS, pages 457–473. Springer, Heidelberg, May 2005. doi:10.1007/11426639\_27.
- [Tom19] Junichi Tomida. Tightly secure inner product functional encryption: Multiinput and function-hiding constructions. In Steven D. Galbraith and Shiho Moriai, editors, ASIACRYPT 2019, Part III, volume 11923 of LNCS, pages 459–488. Springer, Heidelberg, December 2019. doi:10.1007/978-3-030-3 4618-8\_16.
- [Tsa22] Rotem Tsabary. Candidate witness encryption from lattice techniques. In Yevgeniy Dodis and Thomas Shrimpton, editors, CRYPTO 2022, Part I, volume 13507 of LNCS, pages 535–559. Springer, Heidelberg, August 2022. doi:10.1007/978-3-031-15802-5\_19.
- [VWW22] Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. Witness encryption and null-IO from evasive LWE. In Shweta Agrawal and Dongdai Lin, editors, ASIACRYPT 2022, Part I, volume 13791 of LNCS, pages 195–221. Springer, Heidelberg, December 2022. doi:10.1007/978-3-031-22963-3\_7.
- [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. doi:10.1007/978-3-642-03356-8\_36.
- [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. doi:10.1007/978-3-319-70500-2\_8.
- [WZ17] Daniel Wichs and Giorgos Zirdelis. Obfuscating compute-and-compare programs under LWE. In Chris Umans, editor, 58th FOCS, pages 600–611. IEEE Computer Society Press, October 2017. doi:10.1109/F0CS.2017.61.