CP-ABE for Circuits (and more) in the Symmetric Key Setting

Shweta Agrawal[1] and Shota Yamada[2]

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

Abstract. The celebrated work of Gorbunov, Vaikuntanathan and Wee [GVW13] provided the first key policy attribute based encryption scheme (ABE) for circuits from the Learning With Errors (LWE) assumption. However, the arguably more natural *ciphertext policy* variant has remained elusive, and is a central primitive not yet known from LWE.

In this work, we construct the first *symmetric key* ciphertext policy attribute based encryption scheme (CP-ABE) for all polynomial sized circuits from the learning with errors (LWE) assumption. In more detail, the ciphertext for a message m is labelled with an access control policy f, secret keys are labelled with public attributes $\mathbf{x}$ from the domain of f and decryption succeeds to yield the hidden message m if and only if $f(\mathbf{x}) = 1$. The size of our public and secret key do not depend on the size of the circuits supported by the scheme – this enables our construction to support circuits of *unbounded size* (but bounded depth). Our construction is secure against collusions of unbounded size. We note that current best CP-ABE schemes [BSW07, Wat11, LOS+10, OT10, LW12, RW13, Att14, Wee14, AHY15, CGW15, AC17, KW19] rely on pairings and only support circuits in the class NC_1 (albeit in the public key setting).

We adapt our construction to the public key setting for the case of *bounded* size circuits. The size of the ciphertext and secret key as well as running time of encryption, key generation and decryption satisfy the efficiency properties desired from CP-ABE, assuming that all algorithms have RAM access to the public key. However, the running time of the setup algorithm and size of the public key depends on the circuit size bound, restricting the construction to support circuits of a-priori bounded size. We remark that the inefficiency of setup is somewhat mitigated by the fact that setup must only be run once.

We generalize our construction to consider attribute and function hiding. The compiler of lockable obfuscation upgrades any attribute based encryption scheme to predicate encryption, i.e. with attribute hiding [GKW17, WZ17]. Since lockable obfuscation can be constructed from LWE, we achieve ciphertext policy predicate encryption immediately. For function privacy, we show that the most natural notion of function hiding ABE for circuits, even in the symmetric key setting, is sufficient to imply indistinguishability obfuscation. We define a suitable weakening of function hiding to sidestep the implication and provide a construction to achieve this notion for both the key policy and ciphertext policy case. Previously, the largest function class for which function private predicate encryption (supporting unbounded keys) could be achieved was inner product zero testing, by Shen, Shi and Waters [SSW09].

1 Introduction

Attribute based encryption (ABE) [SW05] is a generalization of public key encryption that enables fine grained access control on encrypted data. In attribute based encryption, a message m is encrypted so that decryption succeeds if and only if the secret key holder is authorized to learn the message. Here, authorization is enforced via an access control policy modelled as a Boolean circuit f, which is computed over some public attributes $\mathbf{x}$ associated with the data/user. The access control policy may be embedded either in the key or the ciphertext, yielding key-policy (KP-ABE) or ciphertext-policy (CP-ABE) respectively.

In more detail, in a CP-ABE scheme, a ciphertext for a message m is labelled with an access control policy f, and secret keys are labelled with public attributes $\mathbf{x}$ from the domain of f. Decryption succeeds to yield the hidden message m if and only if the attribute satisfies the function, namely $f(\mathbf{x}) = 1$. In a KP-ABE, the placement of f and $\mathbf{x}$ are swapped.

Ciphertext Policy ABE for Circuits. Both KP-ABE[SW05, GPSW06, BW07, KSW08, LOS+10, OT10, OT12, CW14, AFV11, LW11, LW12, Wat12, GVW13, Wee14, Att14, BGG+14, GVW15, GV15, BV16, AF18] and CP-ABE schemes have received a lot of attention [BSW07, Wat11, LOS+10, OT10, LW12, RW13, Att14, Wee14, AHY15, CGW15, AC17, KW19] in the literature. While KP-ABE for the richest class of functions rely on the Learning With Errors (LWE) assumption and can support all polynomial sized circuits, the most general CP-ABE rely on pairings and can only support circuits in NC_1 [BSW07, Wat11, LOS+10, OT10, LW12, RW13, Att14, Wee14, AHY15, CGW15, AC17, KW19].

Recently, Tsabary [Tsa19] provided a construction of (public key) CP-ABE from Learning With Errors (LWE) for the very restricted class of t-CNF formulae, where t is constant. However, for all polynomial sized circuits, any construction from standard assumptions[3] has remained elusive despite substantial research effort. Very recently, Brakerski and Vaikuntanathan do provide a construction of (public key) CP-ABE using lattice based techniques [BV20], but their construction lacks a security proof. Their work further highlights the technical barriers to providing a construction from LWE. Indeed, constructing CP-ABE for even NC_1 from LWE is widely acknowledged as a central problem in lattice based cryptography and would be considered a major breakthrough.

Function Hiding. An ABE scheme encodes an attribute vector $\mathbf{x}$ and a Boolean circuit f. Hiding the attribute in these constructions, à la *Predicate Encryption* (PE) has met with fantastic success – the celebrated work of Gorbunov, Vaikuntanathan and Wee [GVW15] constructed a predicate encryption system for all circuits from LWE. More recently, Goyal, Koppula and Waters [GKW17] as well as Wichs and Zirdelis [WZ17] provided a powerful compiler for upgrading any ABE to PE by assuming LWE. However,

[3] We note that from strong assumptions such as the the existence of multilinear maps [GGH13a], witness encryption [GTKP+13a] or indistinguishability obfuscation [BGI+01, GGH+13b], attribute based encryption (indeed, even its generalization *functional encryption*) has been constructed for all circuits, but these are not considered standard assumptions.

much less is known about function hiding for ABE. For restricted functionalities such as identity based encryption and subspace membership testing, function hiding has received attention [BRS13a, BRS13b] in the public key setting, but serious technical barriers present themselves for more general function classes. We refer the reader to [BRS13a, BRS13b] for a detailed discussion.

In the symmetric key setting, function hiding for the stronger notion of *functional encryption* has been studied extensively [GTKP+13b, BS15] – however, since functional encryption is known to imply indistinguishability obfuscation [AJ15, BV15, BNPW16, KNT18] even without function hiding, there is limited optimism about achieving this notion for all circuits from standard assumptions, given current state of art. On the other hand, for the restricted inner product functionality, function hiding functional encryption can be achieved from standard assumptions [BJK15, KLM+16]. For the related (but distinct) functionality of inner product zero testing, Shen, Shi and Waters [SSW09] provided a construction of function hiding, symmetric key predicate encryption from bilinear maps.

The above state of affairs is dissatisfying and reveals several gaps in our understanding. Concretely, for general circuits and from standard assumptions, can we achieve function hiding in the symmetric key setting? Note that while attribute based encryption [GVW13, BGG+14] and predicate encryption [GVW15] are achievable from standard assumptions for all circuits, the richest functionality for which function hiding predicate encryption has been achieved is the inner product zero testing functionality [SSW09]. We emphasize that this question is not just of theoretical interest – as noted by Shen et al. [SSW09], function private predicate encryption in the symmetric key setting has many compelling applications. As an example [SSW09], a user may wish to store encrypted files on an untrusted server, and later retrieve only files that satisfy a given predicate. It is a natural security requirement that the server must learn nothing more about the predicate than minimum possible. We refer the reader to [SSW09] for a detailed discussion.

1.1 Our Results

In this work, we make substantial progress on both questions discussed above. Our results are summarized as follows:

1. We construct the first *symmetric key* ciphertext policy attribute based encryption scheme (CP-ABE) for all polynomial sized circuits from the learning with errors (LWE) assumption. The sizes of our public and secret key do not depend on the size of the circuits supported by the scheme – this enables our construction to support circuits of *unbounded size* (but bounded depth). Our construction is secure against collusions of unbounded size in the multi-challenge ciphertext setting.[4]

   This is the first construction of CP-ABE for polynomial circuits of unbounded size, supporting unbounded collusions, from standard assumptions.

[4] In the symmetric key setting, single-challenge ciphertext security and multi-challenge ciphertext security are not equivalent. In our paper, we adopt the latter as the default security notion for symmetric key ABE, since it is stronger and more natural.

2. We adapt our construction to the public key setting for the case of *bounded* size circuits. The size of the ciphertext and secret key as well as the runtime of encryption, key generation and decryption satisfy the efficiency properties desired from CP-ABE. However, the running time of the setup algorithm and the size of the public key depend on the circuit size bound, restricting the construction to support circuits of a-priori bounded size. We remark that this inefficiency is mitigated by the fact that setup must only run once. We summarize our results in Table 1.

Scheme	Assumption	PK/SK	Setup Time	|PK|	Enc Time	|CT|	KeyGen Time	|SK|	Dec Time	Circuit Class
Ideal	Standard	PK	1	1	$\|f\|$	$\|f\|$	$\|\mathbf{x}\|$	$\|\mathbf{x}\|$	$\|f\|$	P
Naive (using [BGG+14])	LWE	PK	$\|f_{\max}\|$	$\|f_{\max}\|$	$\|f_{\max}\|$	$\|f_{\max}\|$	$\|f_{\max}\|$	1	$\|f_{\max}\|$	P
Naive ([BGG+14] & [BV16])[5]	LWE	PK	1	1	$\|f_{\max}\|$	$\|f_{\max}\|$	$\|f_{\max}\|$	1	$\|f_{\max}\|$	P
Section 3	LWE	SK	1	1	$\|f\|$	$\|f\|$	$\|\mathbf{x}\|$	$\|\mathbf{x}\|$	$\|f\|$	P
Section 4	LWE	PK	$\|f_{\max}\|$	$\|f_{\max}\|$	$\|f\|$	$\|f\|$	$\|\mathbf{x}\|$	$\|\mathbf{x}\|$	$\|f\|$	P
[KW19]	Pairings	PK	1	1	$\|f\|$	$\|f\|$	$\|\mathbf{x}\|$	$\|\mathbf{x}\|$	$\|f\|$	NC_1

Table 1 : $|f_{\max}|$ denotes the worst case size bound on circuit size, and $|f|$ denotes the input circuit size. All the entries hide $\mathrm{poly}(\lambda)$ and logarithmic factors in $|f_{\max}|$. Due to space constraints, we include only the most recent pairings based cpABE in the table.

3. We study the notion of function hiding attribute based encryption for circuits, in the symmetric key setting. In Section 5.3, we show that the most natural notion of function hiding ABE, even in the symmetric key setting is sufficient to imply indistinguishability obfuscation. We define a suitable weakening of function hiding to sidestep the implication and provide a construction in Section 5 to achieve this notion for both key policy and ciphertext policy predicate encryption. We instantiate our compiler with known constructions of PE to obtain the following theorems:

   Theorem 1.1. *(Informal) Assuming subexponential LWE, we have function hiding, semi-adaptively secure predicate encryption for all polynomial circuits.*

   Theorem 1.2. *(Informal) Assuming subexponential LWE and DLIN, we have function hiding, adaptively secure predicate encryption for* NC_1 *circuits.*

   Please see Section 5.1 for details.

[5] This construction can be further improved by combining this with the "powers of 2" trick where we run parallel instances of the scheme that can deal with circuits with size at most 2^i for $i = 1, 2, \ldots \log |f_{\max}|$ and use appropriate instance when encrypting a message depending on the size of the circuit. As a result, the encryption time, the ciphertext size, and the (RAM efficiency of the) decryption algorithm can be reduced to be $|f|$ from $|f_{\max}|$.

1.2 Our Techniques

In this section, we provide an overview of our techniques.

CP-ABE for Circuits. For this construction, we leverage techniques developed recently by Agrawal, Maitra and Yamada [AMY19] to handle inputs of unbounded size in the context of ABE for finite automata. We notice that these techniques are quite a bit more general than discussed in that work and can be adapted to the setting of ciphertext policy ABE supporting unbounded collusions.

Folklore Approach. We begin with a folklore transformation of KP-ABE to CP-ABE – namely, via the universal circuit. In more detail, let $U(\cdot,\cdot)$ be the universal circuit such that $f(\mathbf{x}) = U(\mathbf{x}, f)$. Next, let $U[\mathbf{x}]$ be the universal circuit with the input $\mathbf{x}$ hard-wired. Then, we may construct a CP-ABE scheme, denoted by cpABE using a KP-ABE scheme, denoted by kpABE as follows: the cpABE encryptor, given a message m and circuit f may compute kpABE ciphertext for (m, f) where f is viewed as a bit string representing kpABE attributes. The cpABE key generator, given an attribute string $\mathbf{x}$, may compute a kpABE function key for the circuit $U[\mathbf{x}]$. Decryption is straightforward using kpABE decryption as $U[\mathbf{x}](f) = U(\mathbf{x}, f) = f(\mathbf{x})$.

The above generic compiler has the drawback that the input of circuit $U[\mathbf{x}]$ is the circuit f. This limits the construction to only support circuits of a-priori bounded size $|f_{\max}|$ (say) and forces the size of the public key, ciphertext as well as runtime of setup, key generation, encryption and decryption to grow with $|f_{\max}|$ (please see Table 1). We emphasize that even the encryption and decryption algorithms, which must take time proportional to circuit size, now degrade with the worst case bound $|f_{\max}|$, rather than with input circuit $|f|$. The hit taken by key generation is significantly worse[6].

Re-distributing Computation. Note that the only algorithms which are allowed to depend on the size of the circuit length are the encryption and decryption algorithms. Hence, inspired by [AMY19], we re-distribute the computation of $\mathsf{kpABE.KeyGen}(U[\mathbf{x}])$ between the key generator and the encryptor to ensure that each algorithm satisfies the efficiency requirements of CP-ABE.

In more detail, the key generator may depend on the size of $\mathbf{x}$ but not on the size of f, while the encryptor and decryptor may depend on the size of f. In order to redistribute computation, we rely on single-key functional encryption (FE), which can be constructed based on the LWE assumption [GKP+13]. Now, the ciphertext of cpABE is $\mathsf{kpABE.CT}(f, m)$ where f is treated as the attribute string. Additionally, the ciphertext contains $\mathsf{FE.KeyGen}(C)$ where the circuit $C(\cdot) = \mathsf{kpABE.KeyGen}(U(\cdot))$. The secret key of cpABE is $\mathsf{FE.Enc}(\mathbf{x})$. Decryption in the cpABE scheme proceeds by first computing FE decryption to obtain $\mathsf{kpABE.SK}(U[\mathbf{x}])$ and then computing kpABE decryption with $\mathsf{kpABE.CT}(f, m)$ to obtain m iff $f(\mathbf{x}) = 1$. Care must be taken that single key security of the underlying FE scheme is not violated. For this, we ensure that the function key is generated for the *same* circuit $C(\cdot) = \mathsf{kpABE.KeyGen}(U(\cdot))$ and

[6] Although using the scheme by [BGG+14] allows for a small function key size.

using the *same* randomness (as specified in the master secret key), across all invocations of FE key generation.

In order to argue that the key generation algorithm does not depend on $|f|$, we rely on special properties of the FE scheme. Recall that the FE scheme of Goldwasser et al. is *succinct* which means that the running time of the encryption algorithm depends on the depth and output length of the circuits supported by the scheme but is independent of their size. The depth of the circuits supported by our construction is bounded by assumption and the depth of the kpABE key generation circuit is at most a polynomial factor larger than the depth of the circuit it supports. Hence, it remains to argue that the output length may be similarly bounded. To see this, note that in our construction, the function key is generated for circuit $C(\cdot) = \mathsf{kpABE.KeyGen}(U(\cdot))$, whose output length depends on the size of the underlying kpABE function key. Fortunately, by using the kpABE scheme of Boneh et al. [BGG+14], we may bound the size of the kpABE scheme by a fixed polynomial.

Supporting Circuits of Unbounded Size. A detail brushed under the carpet in the above description is that the kpABE scheme which is used to encrypt f as an attribute string must be initialized with the length of f during the setup phase. Moreover, this input length is passed to all other kpABE algorithms, notably the key generation algorithm. Since we wish to support f of unbounded size, this poses a dilemma. An immediate question that arises is which algorithm of cpABE should invoke the setup algorithm of kpABE? Evidently, the setup of cpABE does not have the size of f, so it must be the encrypt algorithm. Hence, the cpABE encrypt algorithm samples the kpABE scheme and provides an FE secret key for the circuit $\mathsf{kpABE.KeyGen}(U(\cdot))$. A subtlety is that the kpABE key generation algorithm must depend on the length of f as discussed above. Then, if f is of varying size across different ciphertexts, the description of $\mathsf{kpABE.KeyGen}(U(\cdot))$ and hence $\mathsf{FE.SK}$ varies with the size of f. This is problematic – since FE only satisfies single key security!

We resolve the above conundrum by running $\lambda + 1$ instances of FE and kpABE in parallel – each to support f of length 2^i where $i \in [0, \lambda]$. The circuit size is padded to the next power of two – a trick used in many works, beginning with [GTKP+13a] – so that we only need to deal with $\lambda + 1$ possible FE, each of which supports the issuing of a *single* secret key, which will compute the kpABE key generation circuit for inputs of length 2^i. The cpABE key generator does not know which instance of FE it must encrypt with, so it encrypts with all of them. For details, please see Section 3.

Security. Our cpABE scheme achieves selective, indistinguishability based security. At a high level, security relies on the security of the instances of the single key FE schemes and kpABE schemes. Similarly to [AMY19], we begin by showing that by security of FE adversary cannot get anything beyond $\{\mathsf{FE.Dec}(\mathsf{FE.sk}_i, \mathsf{FE.ct}_i) = \mathsf{kpABE.sk}_i\}$ for $i \in [0, \lambda]$. Next, we rely on the security of kpABE to argue that the message bit is not revealed. As discussed above, we need to ensure that only single FE secret key is revealed to the adversary for each instance of FE. Fortunately, this can be guaranteed by the fact that for a given instance of FE, we must only release a secret key (of the FE) for the key generation algorithm of the corresponding kpABE.

Public Key Setting. Next, we construct a public key ciphertext policy ABE scheme for bounded sized circuits, where $|f_{\max}|$ is set as an upper bound on circuit size. In our construction, the size of the secret key and ciphertext satisfy the efficiency properties desired from CP-ABE (Definition 2.4). Additionally, the running time of the keygen, encrypt and decrypt algorithms depend only on the size of the input circuit f and not on the worst case circuit size $|f_{\max}|$, assuming that they have RAM access to the public key. However, the running time of the setup algorithm and the size of PK grows with the size $|f_{\max}|$ of the circuits supported by the scheme. We note that this inefficiency is mitigated since it must be only run once.

The construction is similar to the secret key cpABE provided in Section 3 but has some important differences. Let us try to adapt the secret key construction of Section 3 to the public key setting. Since the construction makes modular use of single key succinct FE [GKP+13] and key policy ABE [BGG+14], and both these schemes can be instantiated in the public key setting from LWE, a first attempt would be to use public key versions of these building blocks and compile a public key version of the secret key cpABE scheme. However this naive approach runs into multiple difficulties. For the key generation algorithm to be independent of the circuit size, it may not compute the circuit $U[\mathbf{x}]$ – indeed, this would render the role of FE useless and collapse back into the naive transformation of a kpABE to cpABE scheme via universal circuits. To avoid the dependence of keygen on circuit size, it is necessary for the encrypt algorithm to compute the FE secret key for the kpABE key generation algorithm, which in turn requires that the encrypt algorithm possess the master secret key $\mathsf{FE.msk}$.

However, a crucial and useful property of the construction is that it only uses FE for a single fixed circuit – hence, to remove the dependence of Enc on $\mathsf{FE.msk}$, an idea is to let setup compute the FE function key itself and provide it as part of the public key. The cpABE public key can contain the public keys of FE as well as kpABE, along with the FE function key for the kpABE key generation algorithm. Now, the encryptor, given input circuit f and message μ, can use the kpABE public key to compute a kpABE ciphertext for (f, μ). The key generator can compute the FE ciphertext for $\mathbf{x}$ and the decryptor can decrypt as before, by performing FE decryption to recover the kpABE function key, followed by kpABE decryption.

An immediate drawback is that this approach forces the circuit size to be fixed at setup time. Additionally, even if we assume an upper bound $|f_{\max}|$ on the size of supported circuits, this approach has the significant disadvantage that the runtime of encryption and decryption as well as the size of the ciphertext to depend on the upper bound $|f_{\max}|$ rather than the actual size of the circuit. When the input circuit is much smaller, this is a significant price to pay in terms of both communication and computation. Another disadvantage is that the size of the public key now grows with the upper bound $|f_{\max}|$. To see this, note that the kpABE public key in general depends on the size of the inputs supported by the scheme, which in this case can be as large as $|f_{\max}|$. There do exist clever ideas to make the size of the kpABE public key independent of the input size [BV16, GKW16], but they do so, unfortunately, at the expense of making the function key depend linearly on input size $|f_{\max}|$. But if the kpABE function key is large, then the size of the FE ciphertext would degrade to support this, making the cpABE function key large, which is precisely what we are trying to avoid!

These issues may be overcome if we assume that every algorithm has RAM access to $\mathsf{cpABE.mpk}$. For simplicity, let us assume that circuit sizes come in powers of 2 – this assumption can be easily removed by padding circuits appropriately. In this case, we run $\eta := \lceil \log |f_{\max}| \rceil$ instances of kpABE in parallel, and let the i^{th} instance handle inputs of length 2^i, for $i \in [\eta]$. Now, we have η public keys for kpABE, each of length 2^i, which together (along with $\mathsf{FE.mpk}_i$ and $\mathsf{FE.sk}_i$) comprise the final public key. If every algorithm has RAM access to this public key, then it may choose the component according to the actual input length of the circuit, namely it may choose i^* such that $|f| = 2^{i^*}$ and access only the i^{*th} component of the public key. Then, the runtime of the encrypt and decrypt algorithm depend on $|f|$ rather than $|f_{\max}|$. For more details, please see Section 4.

Function Hiding Predicate Encryption. Next, we generalize our construction to consider attribute and function hiding. The compiler of lockable obfuscation upgrades any attribute based encryption scheme to predicate encryption, i.e. with attribute hiding [GKW17, WZ17]. Since lockable obfuscation can be constructed from LWE, we achieve ciphertext policy predicate encryption immediately. We then turn to the question of function hiding predicate encryption for circuits. Here, we show that the natural notion of function hiding predicate encryption, i.e. that considered by [SSW09], when applied to all polynomial sized circuits, is strong enough to imply indistinguishability obfuscation.

Consider a function private ciphertext-policy *attribute based* encryption scheme cpABE[7]. The ciphertext is associated with a circuit f and a message m and the key is associated with an attribute vector $\mathbf{x}$. Intuitively, since the scheme is function hiding, $\mathbf{x}$ is hidden. Note that the attribute f is not hidden, since this an ABE scheme. A natural game of function hiding would allow an adversary to output challenge key queries $(\mathbf{x}_{0i}, \mathbf{x}_{1i})$ and ciphertext queries (f_j, μ_j) so that $f_j(\mathbf{x}_{0i}) = f_j(\mathbf{x}_{1i})$ for all i, j. The challenger responds by choosing a random bit b and returning the corresponding secret keys for $\mathbf{x}_{bi}$, along with ciphertexts for (f_j, μ_j). The adversary wins if she guesses the bit correctly[8].

We now show a reduction from secret key *functional encryption* (FE) to function hiding cpABE. Recall that in functional encryption, the ciphertext is associated with a vector $\mathbf{x}$, the secret key is associated with a circuit f and decryption enables the decryptor to recover $f(\mathbf{x})$. In the security game, the adversary must distinguish between encryptions of $\mathbf{x}_0$ and $\mathbf{x}_1$ given an arbitrary number of secret keys for circuits f_i where $f_i(\mathbf{x}_0) = f_i(\mathbf{x}_1)$. In our reduction, if cpABE supports unbounded ciphertext queries, then FE supports unbounded key queries. Such a functional encryption scheme is known to imply indistinguishability obfuscation (iO) [AJ15, BV15, BNPW16, KNT18].

It remains to outline the reduction. The reduction is remarkably simple: suppose that $\mathsf{FE.Enc}(\mathbf{x}, \mathsf{msk}) = \mathsf{cpABE.KeyGen}(\mathbf{x}, \mathsf{msk})$ and that $\mathsf{FE.KeyGen}(f, \mathsf{msk}) = (m, \mathsf{cpABE.Enc}(f, m, \mathsf{msk}))$ where m is a random bit. $\mathsf{FE.Dec}$ computes $\mathsf{cpABE.Dec}$ and outputs 1 if it recovers m correctly. Now, when the FE adversary outputs $\mathbf{x}_0, \mathbf{x}_1$ as challenge messages, the reduction outputs outputs $\mathbf{x}_0, \mathbf{x}_1$ as challenge keys and obtains

[7] Note that we are starting with a weaker object – this only strengthens our result.

[8] Note that (f_j, μ_j) are ciphertext queries, not challenge ciphertexts, so the adversary is allowed to have decrypting keys for these in a function hiding game.

the cpABE key for $\mathbf{x}_b$. When the FE adversary makes a key request for f_i, the reduction obtains the cpABE ciphertext for (f_i, m_i) where m_i is randomly chosen, and uses these to respond to the FE adversary. It is evident that if the FE adversary is legitimate, then so is the cpABE function hiding adversary. Also, clearly if the cpABE adversary wins the game, this translates to a win for the FE adversary.

To avoid the implication to FE, we weaken the function hiding definition. We provide a restricted definition of function hiding (Definition 2.14), in which the adversary is disallowed from making queries for vectors $\mathbf{x}_0, \mathbf{x}_1$ such that $f_i(\mathbf{x}_0) = f_i(\mathbf{x}_1) = 1$ for any requested f_i. The definition insists that $f_i(\mathbf{x}_0) = f_i(\mathbf{x}_1) = 0$ for all requests. Note that an admissible FE adversary may request keys for any circuits f_i as long as $f_i(\mathbf{x}_0) = f_i(\mathbf{x}_1)$, regardless of whether this value is 0 or 1. However, with the restriction on the function hiding definition, the above reduction fails and we fall back into "one sided security" that characterizes PE and is known to be achievable from standard assumptions. Please see Section 5.3 for the detailed argument.

In Section 5, we provide a construction of predicate encryption for circuits which achieves the above notion of function hiding. Our compiler is analogous to the compiler of Goldwasser et al. [GKP+13], which converts succinct functional encryption to reusable garbled circuits. In more detail, we construct function hiding PE from PE and a symmetric key encryption scheme SKE. For simplicity, we consider the key-policy setting, we show how to extend the argument to the ciphertext-policy setting in Section 5.

Since we are in the symmetric key setting, the SKE secret key SK (say) is known both to the key generation and the encrypt algorithms. Now, the encryptor uses PE to encrypt its message with attribute $(\mathsf{SK}, \mathbf{x})$. The key generator, given input circuit f, computes the SKE encryption $\hat{f}$ of f provides a key for an augmented circuit $U_{\hat{f}}(\cdot)$, which given input $(\mathsf{SK}, \mathbf{x})$, first decrypts $\hat{f}$ to obtain f and then computes $f(\mathbf{x})$. Intuitively, since PE is attribute hiding, SK remains hidden, and since the key only reveals the encryption $\hat{f}$, the circuit f remains hidden. The formal argument is provided in Section 5.

1.3 Perspective and Open Problems.

CP-ABE from LWE, for all polynomial sized circuits (or even NC_1) is a long standing open problem. Our work settles the question in the symmetric key case, and makes significant progress in the public key case. Our constructions use prior constructions of KP-ABE [BGG+14] and FE [GTKP+13a] as building blocks and combine them carefully to obtain the desired efficiency for CP-ABE. These building blocks satisfy certain special properties such as succinctness of ciphertext [GTKP+13a] and short secret key [BGG+14]. By noticing that the efficiency properties of these schemes *compose* in a fortuitous way, we achieve the required efficiency of CP-ABE by doing very little work[9]! Similar tricks were used by [AMY19] in the context of constructing ABE for finite automata – indeed, our constructions are *simpler* than theirs.

An obvious open problem is to close the "efficiency" gap in setup time that remains open in our public key construction. The chief hurdle in doing so is that the computation of the FE secret key is a secret key operation but the only algorithms in the construction

[9] Beyond what is already done by the "heavy hammers" of [BGG+14, GTKP+13b]

that are allowed the time required by this computation, namely encrypt and decrypt, are public key algorithms. An approach may be to delegate the FE secret key generation using garbled circuits, as in [DG17] but a natural implementation of this idea turns out to be insecure. We conjecture that new techniques may be required to overcome this hurdle. In the context of function privacy, we obtain the first attribute based encryption schemes for circuits with function hiding, in the symmetric key setting. A natural open question is to provide constructions in the public key setting. However, as observed by [BRS13a], function privacy in the public key setting is significantly more challenging, with even the right definition being unclear. We conjecture that this problem may require significantly new ideas to resolve.

2 Preliminaries

Notation. We begin by defining the notation that we will use throughout the paper. We use bold letters to denote vectors and the notation $[a, b]$ to denote the set of integers $\{k \in \mathbb{N} \mid a \leq k \leq b\}$. We use $[n]$ to denote the set $[1, n]$. Concatenation is denoted by the symbol $\|$. Vectors will be column vectors unless stated otherwise.

We say a function $f(n)$ is *negligible* if it is $O(n^{-c})$ for all $c > 0$, and we use $\mathrm{negl}(n)$ to denote a negligible function of n. We say $f(n)$ is *polynomial* if it is $O(n^c)$ for some constant $c > 0$, and we use $\mathrm{poly}(n)$ to denote a polynomial function of n. We use the abbreviation PPT for probabilistic polynomial-time. We say an event occurs with *overwhelming probability* if its probability is $1 - \mathrm{negl}(n)$. The function $\log x$ is the base 2 logarithm of x. For any finite set S we denote $\mathcal{P}(S)$ to be the power set of S. For a circuit $C : \{0,1\}^{\ell_1+\ell_2} \to \{0,1\}$ and a string $\mathbf{x} \in \{0,1\}^{\ell_1}$, $C[\mathbf{x}] : \{0,1\}^{\ell_2} \to \{0,1\}$ denotes a circuit that takes $\mathbf{y}$ and outputs $C(\mathbf{x}, \mathbf{y})$. We construct $C[\mathbf{x}]$ in the following specified way. Namely, $C[\mathbf{x}]$ is the circuit that takes as input $\mathbf{y}$ and sets

$$z_i = \begin{cases} y_1 \wedge \neg y_1 & \text{if } x_i = 0 \\ y_1 \vee \neg y_1 & \text{if } x_i = 1 \end{cases}$$

and then computes $C(\mathbf{z}, \mathbf{y})$, where x_i, y_i, and z_i are the i-th bit of $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$, respectively. In the above, it is clear that $z_i = x_i$ and we have $C(\mathbf{z}, \mathbf{y}) = C(\mathbf{x}, \mathbf{y})$. Furthermore, it is also easy to see that $\mathsf{depth}(C[\mathbf{x}]) \leq \mathsf{depth}(C) + O(1)$ holds.

Circuit Classes of Interest. For $\lambda \in \mathbb{N}$, let $\mathcal{C}_{\mathsf{inp},\mathsf{d},\mathsf{s}}$ denote a family of circuits with inp bit inputs, bounded depth d, bounded size s and binary output. When the size s is unspecified, it means that the circuit family $\mathcal{C}_{\mathsf{inp},\mathsf{d}}$ can have unbounded size.

2.1 Attribute Based Encryption for circuits

Attribute based encryption comes in two flavours: key policy or ciphertext policy, depending on where the policy (represented as a Boolean circuit) is embedded. We define these next.

Ciphertext Policy Attribute based Encryption for Circuits. Let $\mathcal{C} = \{\mathcal{C}_{\mathsf{inp}(\lambda),\mathsf{d}(\lambda)}\}_{\lambda\in\mathbb{N}}$. A ciphertext policy attribute-based encryption (ABE) scheme cpABE for $\mathcal{C}$ over a message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ consists of four algorithms:

- $\mathsf{cpABE.Setup}(1^\lambda, 1^{\mathsf{inp}}, 1^{\mathsf{d}})$ is a PPT algorithm takes as input the unary representation of the security parameter, the length $\mathsf{inp} = \mathsf{inp}(\lambda)$ of the input, the depth $\mathsf{d} = \mathsf{d}(\lambda)$ of the circuit family $\mathcal{C}$ to be supported. It outputs the master public key and the master secret key $(\mathsf{cpABE.mpk}, \mathsf{cpABE.msk})$.
- $\mathsf{cpABE.Enc}(\mathsf{cpABE.mpk}, C, m)$ is a PPT algorithm that takes as input the master public key $\mathsf{cpABE.mpk}$, circuit $C \in \mathcal{C}_{\mathsf{inp}(\lambda),\mathsf{d}(\lambda)}$ and a message $m \in \mathcal{M}$. It outputs a ciphertext $\mathsf{cpABE.ct}$.
- $\mathsf{cpABE.KeyGen}(\mathsf{cpABE.mpk}, \mathsf{cpABE.msk}, \mathbf{x})$ is a PPT algorithm that takes as input the master public key $\mathsf{cpABE.mpk}$, the master secret key $\mathsf{cpABE.msk}$, and a a string $\mathbf{x} \in \{0,1\}^{\mathsf{inp}}$ and outputs a corresponding secret key $\mathsf{cpABE.sk}_{\mathbf{x}}$.
- $\mathsf{cpABE.Dec}(\mathsf{cpABE.mpk}, \mathsf{cpABE.sk}_{\mathbf{x}}, \mathbf{x}, \mathsf{cpABE.ct}, C)$ is a deterministic algorithm that takes as input the secret key $\mathsf{cpABE.sk}_{\mathbf{x}}$, its associated attribute string $\mathbf{x}$, a ciphertext $\mathsf{cpABE.ct}$, and its associated circuit C and outputs either a message m' or $\perp$.

Definition 2.1 (Correctness).

A ciphertext policy ABE scheme for circuits cpABE *is correct if for all* $\lambda \in \mathbb{N}$*, polynomially bounded* inp *and* d*, all circuits* $C \in \mathcal{C}_{\mathsf{inp}(\lambda),\mathsf{d}(\lambda)}$*, all* $\mathbf{x} \in \{0,1\}^{\mathsf{inp}}$ *such that* $C(\mathbf{x}) = 1$ *and for all messages* $m \in \mathcal{M}$*,*

$$\Pr\left[\begin{array}{l}(\mathsf{cpABE.mpk}, \mathsf{cpABE.msk}) \leftarrow \mathsf{cpABE.Setup}(1^\lambda, 1^{\mathsf{inp}}, 1^{\mathsf{d}}),\\ \mathsf{cpABE.sk}_{\mathbf{x}} \leftarrow \mathsf{cpABE.KeyGen}(\mathsf{cpABE.mpk}, \mathsf{cpABE.msk}, \mathbf{x}),\\ \mathsf{cpABE.ct} \leftarrow \mathsf{cpABE.Enc}(\mathsf{cpABE.mpk}, C, m) :\\ \mathsf{cpABE.Dec}\Big(\mathsf{cpABE.mpk}, \mathsf{cpABE.sk}_{\mathbf{x}}, \mathbf{x}, \mathsf{cpABE.ct}, C\Big) \neq m\end{array}\right] = \mathrm{negl}(\lambda)$$

where the probability is taken over the coins of $\mathsf{cpABE.Setup}$*,* $\mathsf{cpABE.KeyGen}$*, and* $\mathsf{cpABE.Enc}$*.*

Definition 2.2. *[Selective Security for* cpABE*] The ABE scheme* cpABE *for a circuit family* $\mathcal{C} = \{\mathcal{C}_{\mathsf{inp}(\lambda),\mathsf{d}(\lambda)}\}_{\lambda\in\mathbb{N}}$ *and a message space* $\{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ *is said to satisfy selective security if for any stateful PPT adversary* A*, there exists a negligible function* $\mathrm{negl}(\cdot)$ *such that*

$$\mathsf{Adv}_{\mathsf{cpABE},\mathsf{A}}(1^\lambda) = \Big|\Pr[\mathsf{Exp}^{(0)}_{\mathsf{cpABE},\mathsf{A}}(1^\lambda) = 1] - \Pr[\mathsf{Exp}^{(1)}_{\mathsf{cpABE},\mathsf{A}}(1^\lambda) = 1]\Big| \leq \mathrm{negl}(\lambda),$$

for all sufficiently large $\lambda \in \mathbb{N}$*, where for each* $b \in \{0,1\}$ *and* $\lambda \in \mathbb{N}$*, the experiment* $\mathsf{Exp}^{(b)}_{\mathsf{cpABE},\mathsf{A}}$*, modeled as a game between adversary* A *and a challenger, is defined as follows:*

1. **Setup phase:** *On input* 1^λ*,* A *submits* $(1^{\mathsf{inp}}, 1^{\mathsf{d}})$ *and the target circuit set* $\mathsf{ChalC} \subset \mathcal{C}_{\mathsf{inp}(\lambda),\mathsf{d}(\lambda)}$ *(of possibly varying sizes), to the challenger. The challenger samples* $(\mathsf{cpABE.mpk}, \mathsf{cpABE.msk}) \leftarrow \mathsf{cpABE.Setup}(1^\lambda, 1^{\mathsf{inp}}, 1^{\mathsf{d}})$ *and replies to* A *with* $\mathsf{cpABE.mpk}$*.*

2. **Query phase:** *During the game,* A *adaptively makes the following queries, in an arbitrary order and unbounded many times.*
   (a) **Key Queries:** A *chooses an attribute string* $\mathbf{x} \in \{0,1\}^{\mathsf{inp}}$ *that satisfies* $C(\mathbf{x}) = 0$ *for all* $C \in \mathsf{ChalC}$*. For each such query, the challenger replies with* $\mathsf{cpABE.sk}_{\mathbf{x}} \leftarrow \mathsf{cpABE.KeyGen}(\mathsf{cpABE.mpk}, \mathsf{cpABE.msk}, \mathbf{x})$.
   (b) **Challenge Queries:** A *submits a circuit* $C \in \mathsf{ChalC}$ *and a pair of equal length messages* $(m_0, m_1) \in (\mathcal{M})^2$ *to the challenger. The challenger replies to* A *with* $\mathsf{cpABE.ct} \leftarrow \mathsf{cpABE.Enc}(\mathsf{cpABE.mpk}, C, m_b)$.
3. **Output phase:** A *outputs a guess bit* b' *as the output of the experiment.*

Remark 2.3. The above definition allows an adversary to make challenge queries multiple times. A more standard (equivalent) notion of the security for an ABE restricts the adversary to make only single challenge query. As in [AMY19], we adopt the above definition since it is convenient for our purpose.

Symmetric Key Setting. In the symmetric key setting, the encryption algorithm additionally takes the master secret key as input and the adversary is permitted to make encryption queries in the security game. As for the security definition, we modify the above game so that the adversary is allowed to make the following type of queries in the query phase:

(c) **Encryption Queries:** A *submits a circuit* $C \in \mathcal{C}_{\mathsf{inp}(\lambda),\mathsf{d}(\lambda)}$ *and a pair of equal length messages* $m \in \mathcal{M}$ *to the challenger. The challenger replies to* A *with* $\mathsf{cpABE.ct} \leftarrow \mathsf{cpABE.Enc}(\mathsf{cpABE.msk}, C, m)$.

Unlike challenge queries, there is no restriction on C and the returned ciphertext may be decryptable by the adversary. Note that we did not have to consider above type of queries in the public key setting since the adversary can encrypt any message by itself. We also note that in the symmetric key setting, single-challenge ciphertext security and multi-challenge ciphertext security are not equivalent. We adopt the latter definition as the default security notion since it is stronger and more natural.

Definition 2.4 (Efficiency.). *For* $\lambda \in \mathbb{N}$*, let* $\mathcal{C}_{\mathsf{inp},\mathsf{d}}$ *denote a family of circuits with* inp *bit inputs, bounded depth* d *and binary output. Let* $\mathcal{C} = \{\mathcal{C}_{\mathsf{inp}(\lambda),\mathsf{d}(\lambda)}\}_{\lambda\in\mathbb{N}}$*. We say a ciphertext policy attribute based encryption scheme* cpABE *for circuit class* $\mathcal{C}$ *is efficient if:*

1. **Setup.** *The runtime of the setup algorithm, and the size of the public key depends only on the input length* inp *and depth bound* d *of the supported circuits.*
2. **Key Generation.** *For an attribute* $\mathbf{x}$*, the runtime of the key generation and size of* SK *depends on the attribute size* $|\mathbf{x}|$ *and (possibly) on circuit depth* d.
3. **Encryption and Decryption.** *The runtime of the encrypt and decrypt algorithms, as well as the size of ciphertext depend on the size of the given input circuit* $|C|$.

Our scheme presented in Section 3 supports unbounded circuits with the above efficiency properties.

Relaxation for Bounded Circuits. We also define a relaxed variant of efficiency for circuits of *bounded* size. In more detail, for $\lambda \in \mathbb{N}$, let $\mathcal{C}_{\mathsf{inp},\mathsf{d},\mathsf{s}}$ denote a family of circuits with inp bit inputs, bounded depth d, bounded size s and binary output. Let $\mathcal{C} = \{\mathcal{C}_{\mathsf{inp}(\lambda),\mathsf{d}(\lambda),\mathsf{s}}\}_{\lambda\in\mathbb{N}}$. Then cpABE for circuit class $\mathcal{C}$ allows the setup algorithm to take circuit size bound 1^{s} as input and its runtime depends on this. However, the runtime of the key generation and size of SK depends on the attribute size $|\mathbf{x}|$ and (possibly) on circuit depth d but not circuit size bound s. Similarly, the runtime of the encrypt and decrypt algorithms, as well as the size of ciphertext depend on the size of the given input circuit $|C|$, and not on worst case size bound s. Our scheme presented in Section 4 supports bounded circuits with the aforementioned relaxation in the efficiency properties.

Key Policy Attribute based Encryption for Circuits. The definition of key policy attribute based encryption (kpABE) is exactly as above, with the role of the circuit C and the attribute $\mathbf{x}$ switched. For completeness, we provide this definition below.

For $\lambda \in \mathbb{N}$, let $\mathcal{C}_{\mathsf{inp},\mathsf{d}}$ denote a family of circuits with inp bit inputs, an a-priori bounded depth d, and binary output and $\mathcal{C} = \{\mathcal{C}_{\mathsf{inp}(\lambda),\mathsf{d}(\lambda)}\}_{\lambda\in\mathbb{N}}$. An attribute-based encryption (ABE) scheme kpABE for $\mathcal{C}$ over a message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ consists of four algorithms:

- $\mathsf{kpABE.Setup}(1^\lambda, 1^{\mathsf{inp}}, 1^{\mathsf{d}})$ is a PPT algorithm takes as input the unary representation of the security parameter, the length $\mathsf{inp} = \mathsf{inp}(\lambda)$ of the input and the depth $\mathsf{d} = \mathsf{d}(\lambda)$ of the circuit family $\mathcal{C}_{\mathsf{inp}(\lambda),\mathsf{d}(\lambda)}$ to be supported. It outputs the master public key and the master secret key $(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk})$.
- $\mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathbf{x}, m)$ is a PPT algorithm that takes as input the master public key $\mathsf{kpABE.mpk}$, a string $\mathbf{x} \in \{0,1\}^{\mathsf{inp}}$ and a message $m \in \mathcal{M}$. It outputs a ciphertext $\mathsf{kpABE.ct}$.
- $\mathsf{kpABE.KeyGen}(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}, C)$ is a PPT algorithm that takes as input the master secret key $\mathsf{kpABE.msk}$ and a circuit $C \in \mathcal{C}_{\mathsf{inp}(\lambda),\mathsf{d}(\lambda)}$ and outputs a corresponding secret key $\mathsf{kpABE.sk}_C$.
- $\mathsf{kpABE.Dec}(\mathsf{kpABE.mpk}, \mathsf{kpABE.sk}_C, C, \mathsf{kpABE.ct}, \mathbf{x})$ is a deterministic algorithm that takes as input the secret key $\mathsf{kpABE.sk}_C$, its associated circuit C, a ciphertext $\mathsf{kpABE.ct}$, and its associated string $\mathbf{x}$ and outputs either a message m' or $\perp$.

Definition 2.5 (Correctness).

An ABE scheme for circuits kpABE *is correct if for all* $\lambda \in \mathbb{N}$*, polynomially bounded* inp *and* d*, all circuits* $C \in \mathcal{C}_{\mathsf{inp}(\lambda),\mathsf{d}(\lambda)}$*, all* $\mathbf{x} \in \{0,1\}^{\mathsf{inp}}$ *such that* $C(\mathbf{x}) = 1$ *and for all messages* $m \in \mathcal{M}$*,*

$$\Pr\left[\begin{array}{l}(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}) \leftarrow \mathsf{kpABE.Setup}(1^\lambda, 1^{\mathsf{inp}}, 1^{\mathsf{d}}),\\ \mathsf{kpABE.sk}_C \leftarrow \mathsf{kpABE.KeyGen}(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}, C),\\ \mathsf{kpABE.ct} \leftarrow \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathbf{x}, m) :\\ \mathsf{kpABE.Dec}\Big(\mathsf{kpABE.mpk}, \mathsf{kpABE.sk}_C, C, \mathsf{kpABE.ct}, \mathbf{x}\Big) \neq m\end{array}\right] = \mathrm{negl}(\lambda)$$

where the probability is taken over the coins of kpABE.Setup*,* kpABE.KeyGen*, and* kpABE.Enc*.*

Definition 2.6 (Selective Security for kpABE). *The ABE scheme kpABE for a circuit family $\mathcal{C} = \{\mathcal{C}_{\mathsf{inp}(\lambda),\mathsf{d}(\lambda)}\}_{\lambda\in\mathbb{N}}$ and a message space $\{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ is said to satisfy selective security if for any stateful PPT adversary A, there exists a negligible function $\mathrm{negl}(\cdot)$ such that*

$$\mathsf{Adv}_{\mathsf{kpABE},\mathsf{A}}(1^\lambda) = \left|\Pr[\mathsf{Exp}^{(0)}_{\mathsf{kpABE},\mathsf{A}}(1^\lambda) = 1] - \Pr[\mathsf{Exp}^{(1)}_{\mathsf{kpABE},\mathsf{A}}(1^\lambda) = 1]\right| \leq \mathrm{negl}(\lambda),$$

for all sufficiently large $\lambda \in \mathbb{N}$, where for each $b \in \{0,1\}$ and $\lambda \in \mathbb{N}$, the experiment $\mathsf{Exp}^{(b)}_{\mathsf{kpABE},\mathsf{A}}$, modeled as a game between adversary A and a challenger, is defined as follows:

1. **Setup phase:** *On input 1^λ, A submits $(1^{\mathsf{inp}}, 1^{\mathsf{d}})$ and the target $X \subset \{0,1\}^{\mathsf{inp}}$, which is a set of binary strings of length inp, to the challenger. The challenger samples $(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}) \leftarrow \mathsf{kpABE.Setup}(1^\lambda, 1^{\mathsf{inp}}, 1^{\mathsf{d}})$ and replies to A with $\mathsf{kpABE.mpk}$.*
2. **Query phase:** *During the game, A adaptively makes the following queries, in an arbitrary order and unbounded many times.*
   (a) **Key Queries:** *A chooses a circuit $C \in \mathcal{C}_{\mathsf{inp},\mathsf{d}}$ that satisfies $C(\mathbf{x}) = 0$ for all $\mathbf{x} \in X$. For each such query, the challenger replies with $\mathsf{kpABE.sk}_C \leftarrow \mathsf{kpABE.KeyGen}(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}, C)$.*
   (b) **Challenge Queries:** *A submits a string $\mathbf{x} \in X$ and a pair of equal length messages $(m_0, m_1) \in (\mathcal{M})^2$ to the challenger. The challenger replies to A with $\mathsf{kpABE.ct} \leftarrow \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathbf{x}, m_b)$.*
3. **Output phase:** *A outputs a guess bit b' as the output of the experiment.*

Remark 2.7. The above definition allows an adversary to make challenge queries multiple times. More standard notion of the security for an ABE restricts the adversary to make only a single challenge query. It is well-known that they are actually equivalent, which is shown by a simple hybrid argument. We adopt the above definition since it is convenient for our purpose.

Boneh et al. [BGG+14] provided a construction of kpABE which we will use in our construction of cpABE. The following theorem, provided in [AMY19] summarizes the efficiency properties of their construction.

Theorem 2.8 (Adapted from [BGG+14]). *There exists a selectively secure ABE scheme $\mathsf{kpABE} = (\mathsf{kpABE.Setup}, \mathsf{kpABE.KeyGen}, \mathsf{kpABE.Enc}, \mathsf{kpABE.Dec})$ with the following properties under the LWE assumption.*

1. *The circuit $\mathsf{kpABE.Setup}(\cdot,\cdot,\cdot;\cdot)$, which takes as input $1^\lambda, 1^{\mathsf{inp}}, 1^{\mathsf{d}}$, and a randomness r and outputs $\mathsf{kpABE.msk} = \mathsf{kpABE.Setup}(1^\lambda, 1^{\mathsf{inp}}, 1^{\mathsf{d}}; r)$, can be implemented with depth $\mathrm{poly}(\lambda, \mathsf{d})$. In particular, the depth of the circuit is independent of inp and the length of the randomness r.*
2. *We have $|\mathsf{kpABE.sk}_C| \leq \mathrm{poly}(\lambda, \mathsf{d})$ for any $C \in \mathcal{C}_{\mathsf{inp},\mathsf{d}}$, where $(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}) \leftarrow \mathsf{kpABE.Setup}(1^\lambda, 1^{\mathsf{inp}}, 1^{\mathsf{d}})$ and $\mathsf{kpABE.sk}_C \leftarrow \mathsf{kpABE.KeyGen}(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}, C)$. In particular, the length of the secret key is independent of the input length inp and the size of the circuit C.*

3. *Let* $C : \{0,1\}^{\mathsf{inp}+\ell} \to \{0,1\}$ *be a circuit such that we have* $C[v] \in \mathcal{C}_{\mathsf{inp},d}$ *for any* $v \in \{0,1\}^{\ell}$*. Then, the circuit* $\mathsf{kpABE.KeyGen}(\cdot,\cdot,C[\cdot];\cdot)$*, that takes as input* $\mathsf{kpABE.mpk}$*,* $\mathsf{kpABE.msk}$*,* v*, and randomness* $\widehat{\mathsf{R}}$ *and outputs* $\mathsf{kpABE.KeyGen}(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}, C[v]; \widehat{\mathsf{R}})$*, can be implemented with depth* $\mathsf{depth}(C) \cdot \mathrm{poly}(\lambda, \mathsf{d})$.

2.2 Key Policy Functional Encryption for Circuits

For $\lambda \in \mathbb{N}$, let $\mathcal{C}_{\mathsf{inp},\mathsf{d},\mathsf{out}}$ denote a family of circuits with inp bit inputs, depth d, and output length out and $\mathcal{C} = \{\mathcal{C}_{\mathsf{inp}(\lambda),\mathsf{d}(\lambda),\mathsf{out}(\lambda)}\}_{\lambda\in\mathbb{N}}$. A functional encryption (FE) scheme $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ for $\mathcal{C}$ consists of four algorithms:

- $\mathsf{FE.Setup}(1^{\lambda}, 1^{\mathsf{inp}}, 1^{\mathsf{d}}, 1^{\mathsf{out}})$ is a PPT algorithm takes as input the unary representation of the security parameter, the length $\mathsf{inp} = \mathsf{inp}(\lambda)$ of the input, depth $\mathsf{d} = \mathsf{d}(\lambda)$, and the length of the output $\mathsf{out} = \mathsf{out}(\lambda)$ of the circuit family $\mathcal{C}_{\mathsf{inp}(\lambda),\mathsf{d}(\lambda),\mathsf{out}(\lambda)}$ to be supported. It outputs the master public key $\mathsf{FE.mpk}$ and the master secret key $\mathsf{FE.msk}$.
- $\mathsf{FE.KeyGen}(\mathsf{FE.mpk}, \mathsf{FE.msk}, C)$ is a PPT algorithm that takes as input the master public key $\mathsf{FE.mpk}$, master secret key $\mathsf{FE.msk}$, and a circuit $C \in \mathcal{C}_{\mathsf{inp}(\lambda),\mathsf{d}(\lambda),\mathsf{out}(\lambda)}$ and outputs a corresponding secret key $\mathsf{FE.sk}_C$. We assume that $\mathsf{FE.sk}_C$ contains C and $\mathsf{FE.mpk}$.
- $\mathsf{FE.Enc}(\mathsf{FE.mpk}, \mathbf{x})$ is a PPT algorithm that takes as input the master public key $\mathsf{FE.mpk}$ and an input message $\mathbf{x} \in \{0,1\}^{\mathsf{inp}(\lambda)}$ and outputs a ciphertext $\mathsf{FE.ct}$.
- $\mathsf{FE.Dec}(\mathsf{FE.mpk}, \mathsf{FE.sk}_C, \mathsf{FE.ct})$ is a deterministic algorithm that takes as input the master public key $\mathsf{FE.mpk}$, a secret key $\mathsf{FE.sk}_C$ and a ciphertext $\mathsf{FE.ct}$ and outputs $C(\mathbf{x})$.

Definition 2.9 (Correctness). *A functional encryption scheme* FE *is correct if for all* $C \in \mathcal{C}_{\mathsf{inp}(\lambda),\mathsf{d}(\lambda),\mathsf{out}(\lambda)}$ *and all* $\mathbf{x} \in \{0,1\}^{\mathsf{inp}(\lambda)}$*,*

$$\Pr\left[\begin{array}{l}(\mathsf{FE.mpk}, \mathsf{FE.msk}) \leftarrow \mathsf{FE.Setup}(1^{\lambda}, 1^{\mathsf{inp}(\lambda)}, 1^{\mathsf{d}(\lambda)}, 1^{\mathsf{out}(\lambda)});\\ \mathsf{ct} \leftarrow \mathsf{FE.Enc}(\mathsf{FE.mpk}, \mathbf{x});\\ \mathsf{FE.Dec}\Big(\mathsf{FE.mpk}, \mathsf{FE.KeyGen}(\mathsf{FE.mpk}, \mathsf{FE.msk}, C), \mathsf{ct}\Big) \neq C(\mathbf{x})\end{array}\right] = \mathrm{negl}(\lambda)$$

where the probability is taken over the coins of $\mathsf{FE.Setup}$*,* $\mathsf{FE.KeyGen}$*,* $\mathsf{FE.Enc}$ *and,* $\mathsf{FE.Dec}$).

We then define full simulation based security for single key FE as in [GKP+13, Defn 2.13].

Definition 2.10 (FULL-SIM Security). *Let* FE *be a functional encryption scheme for a circuits. For a stateful PPT adversary* A *and a stateless PPT simulator* Sim*, consider the following two experiments:*

$\mathsf{Exp}^{\mathsf{real}}_{\mathsf{FE},\mathsf{A}}(1^\lambda)$:	$\mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{FE},\mathrm{Sim}}(1^\lambda)$:
1: $(1^{\mathsf{inp}}, 1^{\mathsf{d}}, 1^{\mathsf{out}}) \leftarrow \mathsf{A}(1^\lambda)$	1: $(1^{\mathsf{inp}}, 1^{\mathsf{d}}, 1^{\mathsf{out}}) \leftarrow \mathsf{A}(1^\lambda)$
2: $(\mathsf{FE.mpk}, \mathsf{FE.msk})$ $\leftarrow \mathsf{FE.Setup}(1^\lambda, 1^{\mathsf{inp}}, 1^{\mathsf{d}}, 1^{\mathsf{out}})$	2: $(\mathsf{FE.mpk}, \mathsf{FE.msk})$ $\leftarrow \mathsf{FE.Setup}(1^\lambda, 1^{\mathsf{inp}}, 1^{\mathsf{d}}, 1^{\mathsf{out}})$
3: $C \leftarrow \mathsf{A}(\mathsf{FE.mpk})$	3: $C \leftarrow \mathsf{A}(\mathsf{FE.mpk})$
4: $\mathsf{FE.sk}_C$ $\leftarrow \mathsf{FE.KeyGen}(\mathsf{FE.mpk}, \mathsf{FE.msk}, C)$	4: $\mathsf{FE.sk}_C$ $\leftarrow \mathsf{FE.KeyGen}(\mathsf{FE.mpk}, \mathsf{FE.msk}, C)$
5: $\alpha \leftarrow \mathsf{A}^{\mathsf{FE.Enc}(\mathsf{FE.mpk},\cdot)}(\mathsf{FE.mpk}, \mathsf{FE.sk}_C)$	5: $\alpha \leftarrow \mathsf{A}^{\mathsf{O}(\cdot)}(\mathsf{FE.mpk}, \mathsf{FE.sk}_C)$

Here, $\mathsf{O}(\cdot)$ *is an oracle that on input* $\mathbf{x}$ *from* A*, runs* Sim *with inputs* $(\mathsf{FE.mpk}, \mathsf{sk}_C, C, C(\mathbf{x}), 1^{\mathsf{inp}})$ *to obtain a ciphertext* $\mathsf{FE.ct}$ *and returns it to the adversary* A*.*

The functional encryption scheme FE *is then said to be single query* FULL-SIM *secure if there exists a PPT simulator* Sim *such that for every PPT adversary* A*, the following two distributions are computationally indistinguishable:*

$$\left\{\mathsf{Exp}^{\mathsf{real}}_{\mathsf{FE},\mathsf{A}}(1^\lambda)\right\}_{\lambda\in\mathbb{N}} \overset{c}{\approx} \left\{\mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{FE},\mathrm{Sim}}(1^\lambda)\right\}_{\lambda\in\mathbb{N}}$$

Remark 2.11. Our definition of FULL-SIM *security game for FE differs from [GKP+13] in that we allow the adversary to access challenge oracle (either* $\mathsf{O}(\cdot)$ *or* $\mathsf{FE.Enc}(\mathsf{FE.mpk}, \cdot)$*) as many times as it wants whereas they only allow one-time access. However, it can be seen that these definitions are equivalent by a simple hybrid argument because the simulation of* $\mathsf{FE.Enc}(\cdot)$ *and* $\mathsf{O}(\cdot)$ *does not require any secret information.*

Gorbunov et al. [GKP+13] provided a construction of single key functional encryption from the learning with errors assumption. The following theorem summarizes the efficiency properties of their construction.

Theorem 2.12 ([GKP+13]). *There exists an FE scheme* $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ *with the following properties.*

1. *For any polynomially bounded* $\mathsf{inp}(\lambda), \mathsf{d}(\lambda), \mathsf{out}(\lambda)$*, all the algorithms in* FE *run in polynomial time. Namely, the running time of* $\mathsf{FE.Setup}$ *and* $\mathsf{FE.Enc}$ *do not depend on the size of circuit description to be supported by the scheme.*
2. *Assuming the subexponential hardness of the LWE problem, the scheme satisfies full-simulation-based security.*

We note that the first property above is called succinctness or semi-compactness of FE. A stronger version of the efficiency property called compactness requires the running time of the encryption algorithm to be dependent only on the length of input message $\mathbf{x}$*. An FE with compactness is known to imply indistinguishability obfuscation [AJ15, BV15].*

IND Based Security for Unbounded Keys. *A functional encryption scheme* FE *for a function family* $\mathcal{C}$ *is secure in the adaptive indistinguishability game, denoted as* ind *secure, if for all probabilistic polynomial-time adversaries* Adv*, the advantage of* Adv *in the following experiment is negligible in the security parameter* λ*:*

1. **Public Key.** *Challenger* Ch *returns* $\mathsf{FE.mpk}$ *to* Adv.
2. **Pre-Challenge Key Queries.** Adv *may adaptively request keys for any circuits* $C_1, \ldots, C_\ell \in \mathcal{C}$. *In response,* Adv *is given the corresponding keys* $\mathsf{FE.sk}_{C_i}$.
3. **Challenge.** Adv *outputs the challenges* $(\mathbf{x}_0, \mathbf{x}_1)$ *to the challenger, subject to the restriction that* $C_i(\mathbf{x}_0) = C_i(\mathbf{x}_1)$ *for all* $i \in [\ell]$. *The challenger chooses a random bit* b, *and returns the ciphertext* $\mathsf{CT}_{\mathbf{x}_b}$.
4. **Post-Challenge Key Queries.** *The adversary may continue to request keys for additional functions* C_i, *subject to the restriction that* $C_i(\mathbf{x}_0) = C_i(\mathbf{x}_1)$ *for all* i. *In response,* Adv *is given the corresponding keys* $\mathsf{FE.sk}_{C_i}$.
5. **Guess.** Adv *outputs a bit* b', *and succeeds if* $b' = b$.

The advantage *of* Adv *is the absolute value of the difference between its success probability and* $1/2$. *In the* selective *game, the adversary must announce the challenge in the first step, before receiving the public key. Note that without loss of generality, in the selective game, the challenge ciphertext can be returned along with the public key. In the* semi-adaptive *game, the adversary must announce the challenge after seeing the public key but before making any key requests.*

Symmetric Key Variant. The symmetric key variant of the above definition follows naturally by removing the public key $\mathsf{FE.mpk}$ from all the algorithms, and providing the encryptor the master secret key $\mathsf{FE.msk}$. In the security definition, the adversary may request encryption queries in addition to the key queries.

2.3 Predicate Encryption for Circuits

A (Key-Policy) Predicate Encryption scheme PE for an attribute universe $\mathcal{X}$, a predicate universe $\mathcal{C}$, and a message space $\mathcal{M}$, consists of four algorithms $(\mathsf{PE.Setup}, \mathsf{PE.Enc}, \mathsf{PE.KeyGen}, \mathsf{PE.Dec})$:

- $\mathsf{PE.Setup}(1^\lambda, \mathcal{X}, \mathcal{C}, \mathcal{M}) \rightarrow (\mathsf{PE.mpk}, \mathsf{PE.msk})$. The setup algorithm gets as input the security parameter λ and a description of $(\mathcal{X}, \mathcal{C}, \mathcal{M})$ and outputs the public parameter $\mathsf{PE.mpk}$, and the master key $\mathsf{PE.msk}$.
- $\mathsf{PE.Enc}(\mathsf{PE.mpk}, \mathbf{x}, \mu) \rightarrow \mathsf{CT}$. The encryption algorithm gets as input $\mathsf{PE.mpk}$, an attribute $\mathbf{x} \in \mathcal{X}$ and a message $\mu \in \mathcal{M}$. It outputs a ciphertext CT.
- $\mathsf{PE.KeyGen}(\mathsf{PE.msk}, C) \rightarrow \mathsf{SK}_C$. The key generation algorithm gets as input $\mathsf{PE.msk}$ and a predicate $C \in \mathcal{C}$. It outputs a secret key SK_C.
- $\mathsf{PE.Dec}((\mathsf{SK}_C, C), \mathsf{CT}) \rightarrow \mu \vee \perp$. The decryption algorithm gets as input the secret key SK_C, a predicate C, and a ciphertext CT. It outputs a message $\mu \in \mathcal{M}$ or $\perp$.

Correctness. We require that for all $(\mathsf{PE.mpk}, \mathsf{PE.msk}) \leftarrow \mathsf{PE.Setup}(1^\lambda, \mathcal{X}, \mathcal{C}, \mathcal{M})$, for all $(\mathbf{x}, C) \in \mathcal{X} \times \mathcal{C}$ and for all $\mu \in \mathcal{M}$,

- For 1-queries, namely $C(\mathbf{x}) = 1$, $\left[\mathsf{PE.Dec}\big((\mathsf{SK}_C, C), \mathsf{CT}\big) = \mu\right] \geq 1 - \mathrm{negl}(\lambda)$
- For 0-queries, namely $C(\mathbf{x}) = 0$, $\left[\mathsf{PE.Dec}\big((\mathsf{SK}_C, C), \mathsf{CT}\big) = \perp\right] \geq 1 - \mathrm{negl}(\lambda)$

Semi-Adaptive Simulation Security. Below, we define the SA-SIM security experiment for predicate encryption (PE) similarly to Gorbunov et al. [GVW15].

Definition 2.13 (SA-SIM **Security).**

Let PE *be a predicate encryption scheme for a circuit family* $\mathcal{C}$*. For every stateful p.p.t. adversary* Adv *and a stateful p.p.t. simulator* Sim*, consider the following two experiments:*

$\underline{\mathsf{Exp}^{\mathsf{real}}_{\mathsf{PE},\mathsf{Adv}}(1^\lambda)\textbf{:}}$

1: $(\mathsf{PE.mpk}, \mathsf{PE.msk}) \leftarrow \mathsf{PE.Setup}(1^\lambda)$
2: $\mathbf{x} \leftarrow \mathsf{Adv}(\mathsf{PE.mpk})$
3: $\mu \leftarrow \mathsf{Adv}^{\mathsf{PE.KeyGen}(\mathsf{PE.msk},\cdot)}(\mathsf{PE.mpk})$
4: $\mathsf{CT} \leftarrow \mathsf{PE.Enc}\big(\mathsf{PE.mpk}, \mathbf{x}, \mu\big)$
5: $\alpha \leftarrow \mathsf{Adv}^{\mathsf{PE.KeyGen}(\mathsf{PE.msk},\cdot)}(\mathsf{CT})$
6: *Output* $(\mathbf{x}, \mu, \alpha)$

$\underline{\mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{PE},\mathrm{Sim}}(1^\lambda)\textbf{:}}$

1: $\mathsf{PE.mpk} \leftarrow \mathrm{Sim}(1^\lambda)$
2: $\mathbf{x} \leftarrow \mathsf{Adv}(\mathsf{PE.mpk})$
3: $\mu \leftarrow \mathsf{Adv}^{\mathrm{Sim}}(\mathsf{PE.mpk})$
4: $\mathsf{CT} \leftarrow \mathrm{Sim}\,(\mathsf{PE.mpk}, 1^{|\mathbf{x}|}, 1^{|\mu|})$
5: $\alpha \leftarrow \mathsf{Adv}^{\mathrm{Sim}}(\mathsf{CT})$
6: *Output* $(\mathbf{x}, \mu, \alpha)$

We say an adversary Adv *is admissible if for all queries* C *that it makes, it holds that* $C(\mathbf{x}) = 0$*.*

The predicate encryption scheme PE *is said to be* SA-SIM*-attribute hiding if there exists a p.p.t. simulator* Sim *such that for every admissible p.p.t. adversary* Adv*, the following two distributions are computationally indistinguishable:*

$$\left\{\mathsf{Exp}^{\mathsf{real}}_{\mathsf{PE},\mathsf{Adv}}(1^\lambda)\right\}_{\lambda\in\mathbb{N}} \overset{c}{\approx} \left\{\mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{PE},\mathrm{Sim}}(1^\lambda)\right\}_{\lambda\in\mathbb{N}}$$

Symmetric Key Variant. The symmetric key variant of the above definition follows naturally by removing the public key PE.mpk from all the algorithms, and providing the encryptor the master secret key PE.msk. In the security definition, the adversary is given access to the encryption oracle in addition to the key generation oracle.

Ciphertext Policy Variant. The ciphertext policy variant of the above definition reverses the role of the ciphertext and key. In more detail, the ciphertext encodes the circuit C along with message μ, and the secret key contains the attribute $\mathbf{x}$. We require that the running time of the key generation algorithm does not depend on the size of the circuit $|C|$ (but may depend on its depth).

2.4 Function Hiding Symmetric Key Predicate Encryption

A Function Hiding Symmetric Key Predicate Encryption scheme FHPE for an attribute universe $\mathcal{X}$, a predicate universe $\mathcal{C}$, and a message space $\mathcal{M}$, consists of four algorithms $(\mathsf{FHPE.Setup}, \mathsf{FHPE.Enc}, \mathsf{FHPE.KeyGen}, \mathsf{FHPE.Dec})$:

$\mathsf{FHPE.Setup}(1^\lambda, \mathcal{X}, \mathcal{C}, \mathcal{M}) \rightarrow \mathsf{FHPE.msk}$. The setup algorithm gets as input the security parameter λ and a description of $(\mathcal{X}, \mathcal{C}, \mathcal{M})$ and outputs the master key FHPE.msk.

$\mathsf{FHPE.Enc}(\mathsf{FHPE.msk}, \mathbf{x}, \mu) \to \mathsf{CT}$. The encryption algorithm gets as input $\mathsf{FHPE.msk}$, an attribute $\mathbf{x} \in \mathcal{X}$ and a message $\mu \in \mathcal{M}$. It outputs a ciphertext CT.

$\mathsf{FHPE.KeyGen}(\mathsf{FHPE.msk}, C) \to \mathsf{SK}_C$. The key generation algorithm gets as input $\mathsf{FHPE.msk}$ and a predicate $C \in \mathcal{C}$. It outputs a secret key SK_C.

$\mathsf{FHPE.Dec}(\mathsf{SK}_C, \mathsf{CT}) \to \mu \vee \perp$. The decryption algorithm gets as input the secret key SK_C and a ciphertext CT. It outputs a message $\mu \in \mathcal{M}$ or $\perp$.

Correctness. We require that for all $(\mathsf{FHPE.msk}) \leftarrow \mathsf{FHPE.Setup}(1^\lambda, \mathcal{X}, \mathcal{C}, \mathcal{M})$, for all $(\mathbf{x}, C) \in \mathcal{X} \times \mathcal{C}$ and for all $\mu \in \mathcal{M}$,

- For 1-queries, namely $C(\mathbf{x}) = 1$, $\Pr\left[\mathsf{PE.Dec}\big(\mathsf{SK}_C, \mathsf{CT}\big) = \mu\right] \geq 1 - \mathrm{negl}(\lambda)$
- For 0-queries, namely $C(\mathbf{x}) = 0$, $\Pr\left[\mathsf{PE.Dec}\big(\mathsf{SK}_C, \mathsf{CT}\big) = \perp\right] \geq 1 - \mathrm{negl}(\lambda)$

Function Hiding IND Security. The standard function hiding indistinguishability game for secret key predicate encryption may be defined as follows.

Definition 2.14 (Function hiding IND Security). *A symmetric key predicate encryption scheme* PE *is function-hiding, if every admissible PPT adversary* Adv *has negligible advantage in the following game:*

1. *Key Generation. The challenger* Ch *samples* $\mathsf{msk} \leftarrow \mathsf{FHPE.Setup}(1^\lambda)$.
2. *The challenger* Ch *chooses a random bit* b *and repeats the following with* Adv *for an arbitrary number of times determined by* Adv*:*
   - *Function Queries. Upon* Adv *choosing a pair of functions* (C_0, C_1)*,* Ch *sends* Adv *a function key* $\mathsf{SK} \leftarrow \mathsf{FHPE.KeyGen}(\mathsf{msk}, C_b)$.
   - *Message Queries. Upon* Adv *choosing a pair of attribute vectors* $(\mathbf{x}_0, \mathbf{x}_1)$ *and a message* μ*,* Ch *sends* Adv *a ciphertext* $\mathsf{CT} \leftarrow \mathsf{FHPE.Enc}(\mathsf{msk}, \mathbf{x}_b, \mu)$.
3. *The adversary outputs a guess* b' *for the bit* b *and wins if* $b = b'$.

We say an adversary is admissible if for all function and message queries, it holds that $C_0(\mathbf{x}_0) = C_1(\mathbf{x}_1) = 0$.

On Ciphertext Queries. A natural game would also allow the adversary to request ciphertexts for attribute vectors $\mathbf{x}_0, \mathbf{x}_1$ and message $\mu_0 = \mu_1 = \mu$ such that $C_0(\mathbf{x}_0) = C_1(\mathbf{x}_1) = 1$, enabling the adversary to recover μ. However, as we show in Section 5.3, such a game renders the primitive strong enough to imply symmetric key functional encryption, which in turn is sufficient to imply iO [BNPW16].

Function Hiding SIM Security. Below, we define attribute and function hiding $\mathsf{SA\text{-}SIM}$ security for predicate encryption (FHPE).

Definition 2.15 (Function Hiding $\mathsf{SA\text{-}SIM}$ Security).

Let FHPE *be a function hiding, symmetric key predicate encryption scheme for a circuit family* $\mathcal{C}$*. For every stateful p.p.t. adversary* Adv *and a stateful p.p.t. simulator* Sim*, consider the following two experiments:*

$\underline{\mathsf{Exp}^{\mathsf{real}}_{\mathsf{PE},\mathsf{Adv}}(1^\lambda)\textbf{:}}$

1: $\mathsf{FHPE.msk} \leftarrow \mathsf{FHPE.Setup}(1^\lambda)$
2: $\{\mathbf{x}^*_i\}_{i\in\mathrm{poly}} \leftarrow \mathsf{Adv}(1^\lambda)$
3: $\{\mu^*_i\}_{i\in\mathrm{poly}}, \{C^*_i\}_{i\in\mathrm{poly}} \leftarrow \mathsf{Adv}^{\mathcal{O}(\mathsf{msk},\cdot)}$
4: $\{\mathsf{CT}_i \leftarrow \mathsf{FHPE.Enc}(\mathsf{msk}, \mathbf{x}_i, \mu^*_i)\}_i$
5: $\{\mathsf{SK}_{C^*_i} \leftarrow \mathsf{FHPE.KeyGen}(\mathsf{msk}, C^*_i)\}_i$
6: $\alpha \leftarrow \mathsf{Adv}^{\mathcal{O}(\mathsf{msk},\cdot)}(\{\mathsf{CT}\}_i, \{\mathsf{SK}_{C^*_i}\}_i)$
7: *Output* $(\{\mathbf{x}^*_i, \mu^*_i\}_i, \{C^*_i\}_i, \alpha)$

$\underline{\mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{PE},\mathrm{Sim}}(1^\lambda)\textbf{:}}$

1: $\{\mathbf{x}^*_i\}_{i\in\mathrm{poly}} \leftarrow \mathsf{Adv}(1^\lambda)$
2: $\{\mu^*_i\}_{i\in\mathrm{poly}}, \{C^*_i\}_{i\in\mathrm{poly}} \leftarrow \mathsf{Adv}^{\mathrm{Sim}}$
3: $\{\mathsf{CT}\}_i, \{\mathsf{SK}_{C^*_i}\}_i$
$\leftarrow \mathrm{Sim}\,(\{1^{|\mathbf{x}^*_i|}, 1^{|\mu^*_i|}\}_i, \{1^{|C^*_i|}\}_i)$
4: $\alpha \leftarrow \mathsf{Adv}^{\mathrm{Sim}}(\{\mathsf{CT}_i\}_i, \{\mathsf{SK}_{C^*_i}\}_i)$
5: *Output* $(\{\mathbf{x}^*_i, \mu_i\}_i, \{C^*_i\}_i, \alpha)$

Above, $\mathcal{O}$ is an oracle that upon receiving attribute and circuit queries from the adversary, returns ciphertexts and keys by running FHPE.Enc *and* FHPE.KeyGen *respectively.*

We say an adversary Adv *is admissible if for all circuit queries C_i and challenge circuits C^*_i, and for all attribute queries $\mathbf{x}_j$ and challenge attributes $\mathbf{x}^*_j$, it holds that $C_i(\mathbf{x}_j) = C^*_i(\mathbf{x}_j) = C_i(\mathbf{x}^*_j) = C^*_i(\mathbf{x}^*_j) = 0$.*

The symmetric key predicate encryption scheme PE *is said to be* SA-SIM *secure with attribute and function hiding if there exists a p.p.t. simulator* Sim *such that for every admissible p.p.t. adversary* Adv*, the following two distributions are computationally indistinguishable:*

$$\left\{\mathsf{Exp}^{\mathsf{real}}_{\mathsf{PE},\mathsf{Adv}}(1^\lambda)\right\}_{\lambda\in\mathbb{N}} \overset{c}{\approx} \left\{\mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{PE},\mathrm{Sim}}(1^\lambda)\right\}_{\lambda\in\mathbb{N}}$$

Adaptive Variant of Security. We can consider stronger variant of the above security definition where the adversary interleaves the challenge queries $\mathbf{x}^*_i$ and C^*_i in an arbitrary order instead of submitting them at the beginning of the game. We call this security notion adaptive simulation function hiding security.

On Ciphertext Queries. We note that the above definition restricts the adversary in its encryption queries. A more natural game would allow an adversary to request a key for a circuit C and encryption for pair $(\mathbf{x}, \mu)$ such that $C(\mathbf{x}) = 1$. This enables the adversary to recover μ but intuitively does not violate security since μ was picked by the adversary. However, as discussed in the case of IND based function hiding, such a game renders the primitive strong enough to imply symmetric key functional encryption, which in turn is sufficient to imply iO [BNPW16].

3 Secret Key CP-ABE for Unbounded Circuits

We construct a secret key ciphertext policy ABE scheme for a family of circuits $\mathcal{C}_{\mathsf{n},\mathsf{d}}$ with n bit inputs, an a-priori bounded depth d, and binary output. Our scheme is denoted by $\mathsf{cpABE} = (\mathsf{cpABE.Setup}, \mathsf{cpABE.KeyGen}, \mathsf{cpABE.Enc}, \mathsf{cpABE.Dec})$ and is constructed using the following ingredients:

1. $\mathsf{PRF} = (\mathsf{PRF.Setup}, \mathsf{PRF.Eval})$: a pseudorandom function, where a PRF key $\mathsf{K} \leftarrow \mathsf{PRF.Setup}(1^\lambda)$ defines a function $\mathsf{PRF.Eval}(\mathsf{K}, \cdot) : \{0,1\}^\lambda \to \{0,1\}$. We denote the length of K by $|\mathsf{K}|$.
2. $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$: a functional encryption scheme for circuit with the efficiency property described in Item 1 of Theorem 2.12. We can instantiate FE with the scheme proposed by Goldwasser et al. [GKP+13].
3. $\mathsf{kpABE} = (\mathsf{kpABE.Setup}, \mathsf{kpABE.KeyGen}, \mathsf{kpABE.Enc}, \mathsf{kpABE.Dec})$: An ABE scheme that satisfies the efficiency properties described in Theorem 2.8. We can instantiate kpABE with the scheme proposed by Boneh et al. [BGG+14].
4. $U(\cdot,\cdot)$: a universal circuit [CH85] that takes as input a circuit C of fixed depth and size and an input $\mathbf{x}$ to the circuit and outputs $C(\mathbf{x})$. We will denote by $U_y(\cdot,\cdot)$ the above circuit when the size of the first input C is y. We denote by $U_y[\mathbf{x}](\cdot) = U(\cdot, \mathbf{x})$ the above circuit with the second input $\mathbf{x}$ being hardwired. By the construction of universal circuit [CH85], we have $\mathsf{depth}(U) \leq O(\mathsf{depth}(C))$.

Below we provide our construction for secret key CP-ABE for circuits. Below, we overload notation and denote the randomness used in a PPT algorithm by a key K of a pseudorandom function PRF. Namely, for a PPT algorithm (or circuit) A that takes as input x and a randomness $r \in \{0,1\}^\ell$ and outputs y, $\mathsf{A}(x;\mathsf{K})$ denotes an algorithm that computes $r := \mathsf{PRF.Eval}(\mathsf{K},1) \| \mathsf{PRF.Eval}(\mathsf{K},2) \| \cdots \| \mathsf{PRF.Eval}(\mathsf{K},\ell)$ and runs $\mathsf{A}(x;r)$.

$\mathsf{cpABE.Setup}(1^\lambda, 1^{\mathsf{n}}, 1^{\mathsf{d}})$: On input the security parameter 1^λ and the input length n and depth d of the circuit family, do the following:

1. For all $j \in [0,\lambda]$, sample PRF keys $\widehat{\mathsf{K}}_j, \mathsf{R}_j \leftarrow \mathsf{PRF.Setup}(1^\lambda)$.
2. For all $j \in [0,\lambda]$, sample $(\mathsf{FE.mpk}_j, \mathsf{FE.msk}_j) \leftarrow \mathsf{FE.Setup}(1^\lambda, 1^{\mathsf{inp}(\lambda)}, 1^{\mathsf{out}(\lambda)}, 1^{\mathsf{d}(\lambda)})$.

   Here, we generate $\lambda+1$ instances of FE. Note that all instances support a circuit class with input length $\mathsf{inp}(\lambda) = \mathsf{n} + 2|\mathsf{K}|$, output length $\mathsf{out}(\lambda)$, and depth $\mathsf{d}(\lambda)$, where $\mathsf{out}(\lambda)$ and $\mathsf{d}(\lambda)$ are polynomials in the security parameter that will be specified later.
3. Output $\mathsf{cpABE.msk} = (\{\widehat{\mathsf{K}}_j, \mathsf{R}_j, \mathsf{FE.mpk}_j, \mathsf{FE.msk}_j\}_{j\in[0,\lambda]})$.

$\mathsf{cpABE.Enc}(\mathsf{cpABE.msk}, C, m)$: On input the master secret key $\mathsf{cpABE.msk}$, a circuit $C \in \mathcal{C}_{\mathsf{n},\mathsf{d}}$, and a message $m \in \mathcal{M}$, do the following:

1. Parse the master secret key as $\mathsf{cpABE.msk} \to (\{\widehat{\mathsf{K}}_j, \mathsf{R}_j, \mathsf{FE.mpk}_j, \mathsf{FE.msk}_j\}_{j\in[0,\lambda]})$.
2. Pad the circuit length to the next power of two: Let $\ell = |C|$ and $i = \lceil \log \ell \rceil$. Set $\hat{C} = C \| \perp^{2^i - \ell}$.
3. Sample a fresh kpABE scheme to support inputs of size $|\hat{C}|$: Compute a kpABE key pair

$$(\mathsf{kpABE.mpk}_i, \mathsf{kpABE.msk}_i) = \mathsf{kpABE.Setup}(1^\lambda, 1^{2^i}, 1^{\hat{\mathsf{d}}}; \widehat{\mathsf{K}}_i)$$

   Here $\widehat{\mathsf{K}}_i$ is the randomness and $\hat{\mathsf{d}}$ is a parameter chosen later.
4. Compute $\mathsf{kpABE.ct} \leftarrow \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}_i, \hat{C}, m)$ as an kpABE ciphertext for the message m under attribute $\hat{C}$.

5. Obtain $\mathsf{FE.sk}_i = \mathsf{FE.KeyGen}(\mathsf{FE.mpk}_i, \mathsf{FE.msk}_i, F_{\mathsf{n},2^i}; \mathsf{R}_i)$, where $F_{\mathsf{n},2^i}$ is a circuit described in Figure 1.
6. Output $\mathsf{cpABE.ct} = (\mathsf{FE.sk}_i, \mathsf{kpABE.mpk}_i, \mathsf{kpABE.ct})$.

Function $F_{\mathsf{n},2^i}$

1. Parse the input $\mathbf{w} = (\mathbf{x}, \widehat{\mathsf{K}}, \widehat{\mathsf{R}})$, where $\mathbf{x}$ is an input string of length n and $\widehat{\mathsf{K}}$ and $\widehat{\mathsf{R}}$ are PRF keys.
2. Compute $(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}) = \mathsf{kpABE.Setup}(1^\lambda, 1^{2^i}, 1^{\hat{\mathsf{d}}}; \widehat{\mathsf{K}})$.
3. Compute and output

$$\mathsf{kpABE.sk}_{U_{2^i}[\mathbf{x}]} = \mathsf{kpABE.KeyGen}(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}, U_{2^i}[\mathbf{x}]; \widehat{\mathsf{R}})$$

Fig. 1

$\mathsf{cpABE.KeyGen}(\mathsf{cpABE.msk}, \mathbf{x})$: On input the master secret key $\mathsf{cpABE.msk}$ and the attribute vector $\mathbf{x}$, do the following:

1. Parse the master secret key as $\mathsf{cpABE.msk} \to (\{\widehat{\mathsf{K}}_j, \mathsf{R}_j, \mathsf{FE.mpk}_j, \mathsf{FE.msk}_j\}_{j\in[0,\lambda]})$.
2. Sample $\widehat{\mathsf{R}}_j \leftarrow \mathsf{PRF.Setup}(1^\lambda)$ for all $j \in [0, \lambda]$.
3. Compute $\mathsf{FE.ct}_j = \mathsf{FE.Enc}(\mathsf{FE.mpk}_j, (\mathbf{x}, \widehat{\mathsf{K}}_j, \widehat{\mathsf{R}}_j))$ for all $j \in [0, \lambda]$.
4. Output $\mathsf{cpABE.sk}_{\mathbf{x}} = \{\mathsf{FE.ct}_j\}_{j\in[0,\lambda]}$.

$\mathsf{cpABE.Dec}(\mathsf{cpABE.sk}_{\mathbf{x}}, \mathbf{x}, \mathsf{cpABE.ct}, C)$: On input a secret key for attribute vector $\mathbf{x}$ and a ciphertext encoded for circuit C, do the following:

1. Parse the secret key as $\mathsf{cpABE.sk}_{\mathbf{x}} = \{\mathsf{FE.ct}_j\}_{j\in[0,\lambda]}$ and the ciphertext as $\mathsf{cpABE.ct} = (\mathsf{FE.sk}_i, \mathsf{kpABE.mpk}_i, \mathsf{kpABE.ct})$.
2. Set $\ell = |C|$ and choose $\mathsf{FE.ct}_i$ from $\mathsf{cpABE.sk}_{\mathbf{x}} = \{\mathsf{FE.ct}_j\}_{j\in[0,\lambda]}$ such that $i = \lceil \log \ell \rceil < \lambda$.
3. Compute $y = \mathsf{FE.Dec}(\mathsf{FE.mpk}_i, \mathsf{FE.sk}_i, \mathsf{FE.ct}_i)$.
4. Compute and output $z = \mathsf{kpABE.Dec}(\mathsf{kpABE.mpk}_i, y, U_{2^i}[\mathbf{x}], \mathsf{kpABE.ct}_i, \hat{C})$, where we interpret y as an ABE secret key and $\hat{C} = C \| \perp^{2^i - \ell}$.

Efficiency. The following theorem asserts that our scheme is efficient.

Theorem 3.1. *For appropriately chosen* $\hat{\mathsf{d}}(\lambda)$, $\mathsf{out}(\lambda)$, *and* $\mathsf{d}(\lambda)$, *each algorithm of our scheme* cpABE *runs in polynomial time of input length.*

Correctness. Intuitively, correctness follows directly from the correctness of kpABE and FE. The following theorem shows that our scheme is correct.

Theorem 3.2. *For appropriately chosen* $\hat{\mathsf{d}}(\lambda)$, $\mathsf{out}(\lambda)$, *and* $\mathsf{d}(\lambda)$, *our scheme* cpABE *is correct for any polynomially bounded* $\mathsf{n}(\lambda)$.

Security. We can prove that if FE and kpABE are secure then so is the cpABE defined above. Formally, we have the following theorem.

Theorem 3.3. *Assume that* FE *satisfies full simulation based security,* kpABE *is selectively secure, and that* PRF *is a secure pseudorandom function. Then,* cpABE *satisfies selective security.*

The proof of the above theorems will appear in the full version.

4 Public Key CP-ABE for Bounded Circuits

In this section, we construct a public key ciphertext policy ABE scheme for bounded sized circuits $\mathcal{C}_{\mathsf{n},\mathsf{d},\mathsf{s}}$, where n is the input length, d is the depth and s is the upper bound of the size . In our construction, the size of the secret key and ciphertext satisfy the efficiency properties desired from CP-ABE (Definition 2.4). Additionally, the running time of the encrypt and decrypt algorithms depend only on the size of the circuit C and not on the worst case circuit size s. However, the running time of the setup algorithm grows with the size s of the circuits supported by the scheme. We note that the inefficiency of setup is mitigated since it is only run once.

We provide the construction next.

$\mathsf{cpABE.Setup}(1^\lambda, 1^\mathsf{n}, 1^\mathsf{d}, 1^\mathsf{s})$: On input the security parameter λ and the input length n, depth d and the upper bound of the size s of the circuit family, set $\eta := \lceil \log \mathsf{s} \rceil$ and do the following:

1. For all $j \in [0, \eta]$, sample PRF keys $\widehat{\mathsf{K}}_j, \mathsf{R}_j \leftarrow \mathsf{PRF.Setup}(1^\lambda)$.
2. For all $j \in [0, \eta]$, sample $(\mathsf{kpABE.mpk}_j, \mathsf{kpABE.msk}_j) = \mathsf{kpABE.Setup}(1^\lambda, 1^{2^j}, 1^{\hat{\mathsf{d}}}; \widehat{\mathsf{K}}_j)$. Here, $\hat{\mathsf{d}}$ is the depth of the universal circuit $U(\cdot,\cdot)$ for circuits of size $\mathsf{s} \geq 2^j$ and depth d.
3. For all $j \in [0, \eta]$, sample $(\mathsf{FE.mpk}_j, \mathsf{FE.msk}_j) \leftarrow \mathsf{FE.Setup}(1^\lambda, 1^{\mathsf{inp}(\lambda)}, 1^{\mathsf{out}(\lambda)}, 1^{\mathsf{d}(\lambda)})$. Here, input length $\mathsf{inp} = \mathsf{n} + 2|\mathsf{K}|$, output length out is the length of the kpABE secret key, and depth $\tilde{\mathsf{d}}$ is the depth of the $\mathsf{kpABE.KeyGen}$ algorithm.
4. For all $j \in [0, \eta]$, obtain $\mathsf{FE.sk}_j = \mathsf{FE.KeyGen}(\mathsf{FE.mpk}_j, \mathsf{FE.msk}_j, F_{\mathsf{n},2^j}; \mathsf{R}_j)$, where $F_{\mathsf{n},2^j}$ is a circuit described in Figure 2.
5. Output $\mathsf{cpABE.mpk} = (\{\mathsf{FE.mpk}_j, \mathsf{kpABE.mpk}_j, \mathsf{FE.sk}_j\}_{j\in[0,\eta]})$ and $\mathsf{cpABE.msk} = (\{\widehat{\mathsf{K}}_j\}_{j\in[0,\eta]})$.

$\mathsf{cpABE.Enc}(\mathsf{cpABE.mpk}, C, m)$: On input the master public key $\mathsf{cpABE.mpk}$, a circuit C of size $|C| = \ell$, and a message $m \in \mathcal{M}$, do the following:

1. Parse the master public key as $\mathsf{cpABE.mpk} \rightarrow (\{\mathsf{FE.mpk}_j, \mathsf{kpABE.mpk}_j, \mathsf{FE.sk}_j\}_{j\in[0,\eta]})$.
2. Pad the circuit length to the next power of two: Set $i = \lceil \log \ell \rceil$ and $\hat{C} = C \| \perp^{2^i - \ell}$.
3. Compute $\mathsf{kpABE.ct} \leftarrow \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}_i, \hat{C}, m)$ as an kpABE ciphertext for the message m under attribute $\hat{C}$.
4. Output $\mathsf{cpABE.ct} = \mathsf{kpABE.ct}$.

Function $F_{\mathsf{n},2^i}$

(a) Parse the input $\mathbf{w} = (\mathbf{x}, \widehat{\mathsf{K}}, \widehat{\mathsf{R}})$, where $\mathbf{x}$ is an input string of length n and $\widehat{\mathsf{K}}$ and $\widehat{\mathsf{R}}$ are PRF keys.

(b) Compute $(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}) = \mathsf{kpABE.Setup}(1^\lambda, 1^{2^i}, 1^{\hat{\mathsf{d}}}; \widehat{\mathsf{K}})$.

(c) Compute and output

$$\mathsf{kpABE.sk}_{U_{2^i}[\mathbf{x}]} = \mathsf{kpABE.KeyGen}(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}, U_{2^i}[\mathbf{x}]; \widehat{\mathsf{R}})$$

Fig. 2

$\mathsf{cpABE.KeyGen}(\mathsf{cpABE.mpk}, \mathsf{cpABE.msk}, \mathbf{x})$: On input the master secret key $\mathsf{cpABE.msk}$ and the attribute vector $\mathbf{x}$, do the following:

1. Parse the master public key as $\mathsf{cpABE.mpk} \to (\{\mathsf{FE.mpk}_j, \mathsf{kpABE.mpk}_j, \mathsf{FE.sk}_j\}_{j\in[0,\eta]})$ and the master secret key as $\mathsf{cpABE.msk} \to (\{\widehat{\mathsf{K}}_j\}_{j\in[0,\eta]})$.
2. Sample $\widehat{\mathsf{R}}_j \leftarrow \mathsf{PRF.Setup}(1^\lambda)$ for all $j \in [0, \eta]$.
3. Compute $\mathsf{FE.ct}_j = \mathsf{FE.Enc}(\mathsf{FE.mpk}_j, (\mathbf{x}, \widehat{\mathsf{K}}_j, \widehat{\mathsf{R}}_j))$ for all $j \in [0, \eta]$.
4. Output $\mathsf{cpABE.sk}_{\mathbf{x}} = \{\mathsf{FE.ct}_j\}_{j\in[0,\eta]}$.

$\mathsf{cpABE.Dec}(\mathsf{cpABE.mpk}, \mathsf{cpABE.sk}_{\mathbf{x}}, \mathbf{x}, \mathsf{cpABE.ct}, C)$: On input a secret key for attribute vector $\mathbf{x}$ and a ciphertext encoded for circuit C, do the following:

1. Parse the secret key as $\mathsf{cpABE.sk}_{\mathbf{x}} = \{\mathsf{FE.ct}_j\}_{j\in[0,\lambda]}$ and the ciphertext as $\mathsf{cpABE.ct} = \mathsf{kpABE.ct}$.
2. Compute $y = \mathsf{FE.Dec}(\mathsf{FE.mpk}_i, \mathsf{FE.sk}_i, \mathsf{FE.ct}_i)$.
3. Compute and output $z = \mathsf{kpABE.Dec}(\mathsf{kpABE.mpk}_i, y, U_{2^i}[\mathbf{x}], \mathsf{kpABE.ct}, C)$, where we interpret y as an ABE secret key.

Correctness and Efficiency. Correctness is evident from correctness of FE and kpABE. By correctness of FE, we get that $y = \mathsf{kpABE.sk}_{U_{2^i}[\mathbf{x}]}$. By correctness of kpABE we get that $z = m$ iff $U_{2^i}[\mathbf{x}](C) = C(\mathbf{x}) = 1$.

Next, we discuss the efficiency of the above scheme. We assume that each algorithm has RAM access to cpABE.mpk. Note that the encryption algorithm runs in time that depends only on the size of the input circuit $|C|$ and not on s. The key generation algorithm runs in polynomial time in $|\mathbf{x}|$ and λ, and the decryption algorithm runs in polynomial time in $|C|$, $|\mathbf{x}|$, and λ. Thus, the above scheme satisfies the relaxed efficiency of Definition 2.4. Note that this efficiency property does not hold if we remove the assumption that each algorithm has RAM access to cpABE.mpk, since the length of cpABE.mpk, which is input to these algorithms, is polynomially dependent on s.

Security. The proof of security directly follows from the secret key case (Section 3). In more detail, we have the following theorem. The proof of the theorem will appear in the full version.

Theorem 4.1. *Assume that* FE *satisfies full simulation based security (Definition 2.10),* kpABE *satisfies selectively security (Definition 2.6), and that* PRF *is a secure*

pseudorandom function. Then, the public key cpABE *described above satisfies selective security (Definition 2.2).*

5 Function Hiding Predicate Encryption for Circuits

In this section, we provide a construction for function hiding predicate encryption in the symmetric key setting. Let the attribute universe be $\mathcal{X}$, the predicate universe be $\mathcal{C}$, the message space be $\mathcal{M}$. Then, we construct the algorithms $(\mathsf{FHPE.Setup}, \mathsf{FHPE.Enc}, \mathsf{FHPE.KeyGen}, \mathsf{FHPE.Dec})$ as follows:

$\mathsf{FHPE.Setup}(1^\lambda, \mathcal{X}, \mathcal{C}, \mathcal{M})$: The setup algorithm gets as input the security parameter λ and a description of $(\mathcal{X}, \mathcal{C}, \mathcal{M})$ and does the following:

1. Sample a symmetric key encryption scheme SKE. Let $\mathsf{SKE.SK} \leftarrow \mathsf{SKE.Setup}(1^\lambda)$.
2. Sample a symmetric key predicate encryption scheme PE without function hiding. Let $\mathsf{PE.msk} \leftarrow \mathsf{PE.Setup}(1^\lambda)$.
3. Output $\mathsf{FHPE.msk} = (\mathsf{PE.msk}, \mathsf{SKE.SK})$.

$\mathsf{FHPE.Enc}(\mathsf{FHPE.msk}, \mathbf{x}, \mu)$: The encryption algorithm gets as input $\mathsf{FHPE.msk}$, an attribute $\mathbf{x} \in \mathcal{X}$, a message $\mu \in \mathcal{M}$, and does the following:

1. Interpret $\mathsf{FHPE.msk} = (\mathsf{PE.msk}, \mathsf{SKE.SK})$.
2. Define $\mathbf{a} = (\mathbf{x}, \mathsf{SKE.SK})$ and compute $\mathsf{CT} \leftarrow \mathsf{PE.Enc}(\mathsf{PE.msk}, \mathbf{a}, \mu)$.
3. Output CT.

$\mathsf{FHPE.KeyGen}(\mathsf{FHPE.msk}, C)$: The key generation algorithm gets as input $\mathsf{FHPE.msk}$, a predicate $C \in \mathcal{C}$ and does the following:

1. Let $\hat{C} = \mathsf{SKE.Enc}(\mathsf{SKE.SK}, C)$.
2. Define the circuit $U_{\hat{C}}(\cdot)$ as in Figure 3.

Function $U_{\hat{C}}$

(a) Parse the input $\mathbf{a} = (\mathbf{x}, \mathbf{k})$, where $\mathbf{x} \in \mathcal{X}$ is an input string and $\mathbf{k}$ is an SKE secret key of length λ.
(b) Compute $C = \mathsf{SKE.Dec}(\hat{C}, \mathbf{k})$
(c) Compute and output $C(\mathbf{x})$

Fig. 3

3. Compute $\mathsf{SK}_C = \mathsf{PE.KeyGen}(\mathsf{PE.msk}, U_{\hat{C}})$ and output it.

$\mathsf{FHPE.Dec}(\mathsf{SK}_C, \mathsf{CT})$: The decryption algorithm gets as input the secret key SK_C and a ciphertext CT, runs $\mathsf{PE.Dec}(\mathsf{SK}_C, \mathsf{CT})$ and outputs it.

Correctness. Correctness follows directly from the correctness of PE and SKE. Note that, by correctness of PE we have that $\mathsf{PE.Dec}(\mathsf{SK}_C, \mathsf{CT}) = U_{\hat{C}}(\mathbf{x}, \mathsf{SKE.SK})$. Next, by correctness of SKE we have $\mathsf{SKE.Dec}(\hat{C}, \mathsf{SKE.SK}) = C$. Hence decryption outputs μ if and only if $U_{\hat{C}}(\mathbf{x}, \mathsf{SKE.SK}) = C(\mathbf{x}) = 1$.

Security. Next, we prove that the above construction satisfies function hiding as defined in Section 2.4. In more detail, we have:

Theorem 5.1. *Suppose that* PE *is a symmetric key predicate encryption scheme satisfying* SA-SIM[10] *attribute hiding (Definition 2.13) and* SKE *is a semantically secure symmetric key encryption scheme. Then the function hiding predicate encryption scheme* FHPE *described above satisfies* SA-SIM *attribute and function hiding (Definition 2.15).*

The proof of the theorem will appear in the full version.

5.1 Instantiating Function Hiding PE from Concrete Assumptions.

In this section, we provide instantiations of function hiding predicate encryption from concrete assumptions.

Semi-adaptively Secure Constructions for Circuits from LWE. Here, we explain that we can construct adaptively secure function hiding PE scheme for circuits from LWE. To do so, we start with semi-adaptively secure ABE for circuits [BV16, GKW16]. This construction can be upgraded to be PE by using lockable obfuscation [GKW17, WZ17]. Plugging the obtained PE scheme into our construction, we obtain the following theorem:

Theorem 5.2. *Assuming LWE, we have function hiding* SA-SIM *secure predicate encryption for all polynomial sized circuits.*

Adaptive Simulation Secure Constructions for NC_1 *Circuits from Bilinear Maps and LWE.* The above construction only achieves selective security. Here, we explain that we can construct adaptive simulation secure function hiding PE scheme for NC_1 circuits by additionally using bilinear maps. To do so, we start with adaptively secure KP-ABE scheme for NC_1 circuits [CGW15, KW19] from the decisional linear (DLIN) assumption on bilinear groups. By applying the ABE-to-PE conversion using lockable obfuscation [GKW17, WZ17], we obtain an adaptively secure (key-policy) PE scheme for NC_1 circuits from the DLIN assumption and the LWE assumption. We can further upgrade its security to adaptive simulation security by the conversion shown by [GKW17, Appendix F]. We then instantiate our construction with this PE scheme. To do so, we need that $U_{\hat{C}}$ is implementable by an NC_1 circuit. It suffices to show that we can implement Step 2a and 2c of $U_{\hat{C}}$ by an NC_1 circuit. The former is possible by instantiating the underlying SKE scheme with the secret key version of the Regev encryption scheme [Reg09], which has NC_1 decryption circuit. The latter is also possible by using the depth-preserving universal circuit [CH85] that takes as input C and x and outputs $C(x)$ and whose depth is only constant time deeper than the depth of C. Summarizing the above discussion, we have the following theorem.

Theorem 5.3. *Assuming LWE assumption and DLIN, we have function hiding adaptive simulation secure predicate encryption for* NC_1 *circuits.*

[10] We note that for PE, IND based security can be bootstrapped into SIM based security as shown by [GKW17, Appendix F].

5.2 Ciphertext Policy Predicate Encryption with Function Hiding

Above, we presented a construction for function hiding predicate encryption in the key policy setting. Now, we leverage this to provide a construction for function hiding predicate encryption in the ciphertext policy setting. Note that the construction for cpABE presented in Section 3 constructions uses a single key functional encryption scheme (FE) along with a key policy attribute based encryption scheme (kpABE) in a modular way. We claim that if we replace the kpABE scheme with a function hiding predicate encryption scheme constructed above, then the resultant scheme achieves attribute and function hiding as well. We refer the reader to the full version for more details.

5.3 Strong Function Hiding Implies iO

The function hiding predicate encryption scheme we constructed above achieves the weaker notion of security of definition 2.14. As discussed in Section 1, if we have a scheme that satisfies a stronger, more natural version of the security, we can construct an iO from this scheme. We refer the reader to the full version for more details.

Acknowledgements. We would like to thank the anonymous reviewers of TCC 2020 for helpful comments. We would also like to thank the Simons Institute for the Theory of Computing, for hosting both authors during the program entitled Lattices: Algorithms, Complexity, and Cryptography. Dr. Agrawal is supported by the DST "Swarnajayanti" fellowship, an Indo-French CEFIPRA project and an "Indo-Israel" ISF-UGC project. The first author thanks Zvika Brakerski for suggesting that CP-ABE is interesting even for the case of bounded sized circuits which led to the construction of Section 4. The second author is supported by JST CREST Grant Number JPMJCR19F6 and JSPS KAKENHI Grant Number 19H01109.

References

AC17. S. Agrawal and M. Chase. Simplifying design and analysis of complex predicate encryption schemes. In *EUROCRYPT*, pages 627–656, 2017.

AF18. Prabhanjan Ananth and Xiong Fan. Attribute based encryption with sublinear decryption from lwe. Cryptology ePrint Archive, Report 2018/273, 2018. https://eprint.iacr.org/2018/273.

AFV11. Shweta Agrawal, David Mandell Freeman, and Vinod Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In *Asiacrypt*, 2011.

AHY15. Nuttapong Attrapadung, Goichiro Hanaoka, and Shota Yamada. Conversions among several classes of predicate encryption and applications to ABE with various compactness tradeoffs. In *ASIACRYPT*, pages 575–601, 2015.

AJ15. Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, pages 308–326, 2015.

AMY19. Shweta Agrawal, Monosij Maitra, and Shota Yamada. Attribute based encryption (and more) for nondeterministic finite automata from learning with errors. In *Crypto*, 2019.

Att14. Nuttapong Attrapadung. Dual system encryption via doubly selective security: Framework, fully secure functional encryption for regular languages, and more. In *Eurocrypt*, 2014.

BGG+14. D. Boneh, C. Gentry, S. Gorbunov, S. Halevi, V. Nikolaenko, G. Segev, V. Vaikuntanathan, and D. Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In *EUROCRYPT*, pages 533–556, 2014.

BGI+01. B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. In *CRYPTO*, 2001.

BJK15. Allison Bishop, Abhishek Jain, and Lucas Kowalczyk. Function-hiding inner product encryption. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology – ASIACRYPT 2015*, 2015.

BNPW16. Nir Bitansky, Ryo Nishimaki, Alain Passelègue, and Daniel Wichs. From cryptomania to obfustopia through secret-key functional encryption. In *Proceedings, Part II, of the 14th International Conference on Theory of Cryptography - Volume 9986*, page 391?418, 2016.

BRS13a. Dan Boneh, Ananth Raghunathan, and Gil Segev. Function-private identity-based encryption: Hiding the function in functional encryption. In *CRYPTO*, 2013.

BRS13b. Dan Boneh, Ananth Raghunathan, and Gil Segev. Function-private subspace-membership encryption and its applications. In *Asiacrypt*, 2013.

BRS13c. Dan Boneh, Ananth Raghunathan, and Gil Segev. Function-private subspace-membership encryption and its applications. In *Proc. of ASIACRYPT 2013, Part I*, 2013.

BS15. Zvika Brakerski and Gil Segev. Function-private functional encryption in the private-key setting. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *Theory of Cryptography*, 2015.

BSW07. John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *IEEE Symposium on Security and Privacy*, pages 321–334, 2007.

BV15. Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. *FOCS*, 2015:163, 2015.

BV16. Zvika Brakerski and Vinod Vaikuntanathan. Circuit-abe from lwe: Unbounded attributes and semi-adaptive security. In *Crypto*, 2016.

BV20. Zvika Brakerski and Vinod Vaikuntanathan. Lattice-inspired broadcast encryption and succinct ciphertext-policy abe. Cryptology ePrint Archive, Report 2020/191, 2020. https://eprint.iacr.org/2020/191.

BW07. Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In *TCC*, pages 535–554, 2007.

CGW15. Jie Chen, Romain Gay, and Hoeteck Wee. Improved dual system ABE in prime-order groups via predicate encodings. In *EUROCRYPT*, pages 595–624, 2015.

CH85. Stephen A. Cook and H. James Hoover. A depth-universal circuit. *SIAM J. Comput.*, 14(4):833–839, 1985.

CW14. Jie Chen and Hoeteck Wee. Semi-adaptive attribute-based encryption and improved delegation for boolean formula. In *SCN 2014*, 2014.

DG17. Nico Dottling and Sanjam Garg. Identity-based encryption from the diffie-hellman assumption. In *Crypto*, 2017.

GGH13a. Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *EUROCRYPT*, 2013.

GGH+13b. Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, 2013. http://eprint.iacr.org/.

GKP+13. Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In *STOC*, pages 555–564, 2013.

GKW16. Rishab Goyal, Venkata Koppula, and Brent Waters. Semi-adaptive security and bundling functionalities made generic and easy. In *TCC*, 2016.

GKW17. Rishab Goyal, Venkata Koppula, and Brent Waters. Lockable obfuscation. In *FOCS*, 2017.

GPSW06. Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM Conference on Computer and Communications Security*, pages 89–98, 2006.

GTKP+13a. S. Goldwasser, Y. Tauman Kalai, R. Popa, V. Vaikuntanathan, and N. Zeldovich. How to run turing machines on encrypted data. In *CRYPTO (2)*, pages 536–553, 2013.

GTKP+13b. S. Goldwasser, Y. Tauman Kalai, R. Popa, V. Vaikuntanathan, and N. Zeldovich. Reusable garbled circuits and succinct functional encryption. In *Proc. of STOC*, pages 555–564. ACM Press, 2013.

GV15. Sergey Gorbunov and Dhinakaran Vinayagamurthy. Riding on asymmetry: Efficient abe for branching programs. In *Proceedings, Part I, of the 21st International Conference on Advances in Cryptology – ASIACRYPT 2015 - Volume 9452*, 2015.

GVW13. Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute based encryption for circuits. In *STOC*, 2013.

GVW15. Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from lwe. In *Crypto*, 2015.

KLM+16. Sam Kim, Kevin Lewi, Avradip Mandal, Hart Montgomery, Arnab Roy, and David Wu. Function-hiding inner product encryption is practical. In *SCN*, 07 2016.

KNT18. Fuyuki Kitagawa, Ryo Nishimaki, and Keisuke Tanaka. Obfustopia built on secret-key functional encryption. In *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II*, pages 603–648, 2018.

KSW08. Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT*, pages 146–162, 2008.

KW19. Lucas Kowalczyk and Hoeteck Wee. Compact adaptively secure ABE for \mathsf nc^1 from k-lin. In *Eurocrypt*, 2019.

LOS+10. Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *EUROCRYPT*, pages 62–91, 2010.

LW11. Allison B. Lewko and Brent Waters. Unbounded HIBE and attribute-based encryption. In *Eurocrypt*, pages 547–567, 2011.

LW12. Allison B. Lewko and Brent Waters. New proof methods for attribute-based encryption: Achieving full security through selective techniques. In *Crypto*, pages 180–198, 2012.

OT10. Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In *CRYPTO*, pages 191–208, 2010.

OT12. Tatsuaki Okamoto and Katsuyuki Takashima. Adaptively attribute-hiding (hierarchical) inner product encryption. In *Advances in Cryptology EUROCRYPT 2012*, pages 591–608. 2012. Full version available at http://eprint.iacr.org/2011/543.

Reg09. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J.ACM*, 56(6), 2009. extended abstract in STOC'05.

RW13. Yannis Rouselakis and Brent Waters. Practical constructions and new proof methods for large universe attribute-based encryption. In *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, pages 463–474, 2013.

SSW09. Emily Shen, Elaine Shi, and Brent Waters. Predicate privacy in encryption systems. In *TCC*, pages 457–473, 2009.

SW05. Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, pages 457–473, 2005.

Tsa19. Rotem Tsabary. Fully secure attribute-based encryption for t-cnf from LWE. In *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part I*, pages 62–85, 2019.

Wat11. Brent Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *Public Key Cryptography – PKC 2011*, 2011.

Wat12. Brent Waters. Functional encryption for regular languages. In *Crypto*, 2012.

Wee14. Hoeteck Wee. Dual system encryption via predicate encodings. In *TCC*, pages 616–637, 2014.

WZ17. Daniel Wichs and Giorgos Zirdelis. Obfuscating compute-and-compare programs under LWE. In *FOCS*, 2017.