# ABE for Circuits with Constant-Size Secret Keys and Adaptive Security

Hanjun Li(✉), Huijia Lin(✉), and Ji Luo(✉)

Paul G. Allen School of Computer Science & Engineering,
University of Washington, Seattle, USA
{hanjul,rachel,luoji}@cs.washington.edu

**Abstract.** An important theme in the research on attribute-based encryption (ABE) is minimizing the sizes of secret keys and ciphertexts. In this work, we present two new ABE schemes with *constant-size* secret keys, i.e., the key size is independent of the sizes of policies or attributes and dependent only on the security parameter $\lambda$.

- We construct the first key-policy ABE scheme for circuits with constant-size secret keys, $|\mathsf{sk}_f| = \mathrm{poly}(\lambda)$, which concretely consist of only three group elements. The previous state-of-the-art scheme by [Boneh et al., Eurocrypt '14] has key size polynomial in the maximum depth $d$ of the policy circuits, $|\mathsf{sk}_f| = \mathrm{poly}(d, \lambda)$. Our new scheme removes this dependency of key size on $d$ while keeping the ciphertext size the same, which grows linearly in the attribute length and polynomially in the maximal depth, $|\mathsf{ct}_{\mathbf{x}}| = |\mathbf{x}|\,\mathrm{poly}(d, \lambda)$.
- We present the first ciphertext-policy ABE scheme for Boolean formulae that simultaneously has constant-size keys and succinct ciphertexts of size independent of the policy formulae, namely, $|\mathsf{sk}_f| = \mathrm{poly}(\lambda)$ and $|\mathsf{ct}_{\mathbf{x}}| = \mathrm{poly}(|\mathbf{x}|, \lambda)$. Concretely, each secret key consists of only two group elements. Previous ciphertext-policy ABE schemes either have succinct ciphertexts but non-constant-size keys [Agrawal–Yamada, Eurocrypt '20, Agrawal–Wichs–Yamada, TCC '20], or constant-size keys but large ciphertexts that grow with the policy size as well as the attribute length. Our second construction is the first ABE scheme achieving *double succinctness*, where both keys and ciphertexts are smaller than the corresponding attributes and policies tied to them.

Our constructions feature new ways of combining lattices with pairing groups for building ABE and are proven selectively secure based on LWE and in the generic (pairing) group model. We further show that when replacing the LWE assumption with its adaptive variant introduced in [Quach–Wee–Wichs FOCS '18], the constructions become adaptively secure.

## 1 Introduction

Attribute-based encryption (ABE) [24,37] is a novel generalization of public-key encryption for enforcing fine-grained access control. In this work, we focus on


E. Kiltz and V. Vaikuntanathan (Eds.): TCC 2022, LNCS 13747, pp. 680–710, 2022.
https://doi.org/10.1007/978-3-031-22318-1_24

improving the efficiency of ABE schemes, especially on minimizing the sizes of secret keys while keeping ciphertexts small. In key-policy (KP) ABE, a secret key $\mathsf{sk}_f$ is tied to a policy $f$ and a ciphertext $\mathsf{ct}_{\mathbf{x}}$ encrypting a message $\mu$ is tied to an attribute $\mathbf{x}$, so that a secret key is only "authorized" to decrypt a ciphertext if the associated attribute $\mathbf{x}$ satisfies the policy $f$. At first glance, since a secret key specifies the associated policy $f$, it appears that the size of the secret key would have to depend at least *linearly* on the (description) size of $f$. Similarly, a ciphertext would have to grow *linearly* with the length of the associated attribute $\mathbf{x}$. Secret keys and ciphertexts with linear dependency of their sizes on the policies and attributes they are tied to are said to be *compact*, and most ABE schemes are indeed compact.

However, upon closer examination, as ABE does not guarantee privacy of the policies nor the attributes, it is possible to give a description of the policy $f$ in the clear in the secret key, and the *non-trivial* part of the secret key may be smaller than the policy. In this case, the right measure of efficiency should be the size of the non-trivial part (i.e., the overhead), which we now view as *the* secret key. We can now aim for secret keys of size smaller than that of the policy — i.e., $|\mathsf{sk}_f| = o(|f|)$ — referred to as *succinct* keys, or even keys of size independent of that of the policy — i.e., $|\mathsf{sk}_f| = O(1)$ — referred to as *constant-size* keys.[1] Similarly, *succinct* ciphertexts have size smaller than the length of the attributes, $|\mathsf{ct}_{\mathbf{x}}| = o(|\mathbf{x}|)$, and *constant-size* ciphertexts satisfy $|\mathsf{ct}_{\mathbf{x}}| = O(1)$. We further examine the efficiency of ciphertext-policy (CP) ABE [10], which enables instead the ciphertexts $\mathsf{ct}_f$ to specify the policies, so that only secret keys $\mathsf{sk}_{\mathbf{x}}$ with attributes satisfying the policies can decrypt them. Naturally, succinct keys and ciphertexts have size $|\mathsf{sk}_{\mathbf{x}}| = o(|\mathbf{x}|)$ and $|\mathsf{ct}_f| = o(|f|)$, and constant size means the same as in KP-ABE.

How close can we get to the *ideal efficiency* of having both constant-size keys and ciphertexts? Despite tremendous effort, the state-of-the-art is still far from the ideal. Current ABE schemes with either succinct keys or succinct ciphertexts can be broadly classified as follows (see Figs. 1 and 2):

- The work of [11] built KP-ABE based on LWE for polynomial-size circuits with succinct keys $|\mathsf{sk}_f| = \mathrm{poly}(d)$ and ciphertexts of size $|\mathsf{ct}_{\mathbf{x}}| = |\mathbf{x}|\,\mathrm{poly}(d)$, where $d$ is the depth of the circuit.
- Several works [5–7,32,38,42,43] constructed KP-ABE and CP-ABE for low-depth computations with *either* constant-size secret keys *or* constant-size ciphertexts from pairing, i.e., *either* $|\mathsf{sk}| = O(1)$ *or* $|\mathsf{ct}| = O(1)$, at the cost of the other component being much larger, of size $\Omega(|f| \cdot |\mathbf{x}|)$.
- The recent works of [3,4] constructed CP-ABE for Boolean formulae with succinct ciphertexts $|\mathsf{ct}_f| = \Theta(|\mathbf{x}|)$ and compact keys $|\mathsf{sk}_{\mathbf{x}}| = \Theta(|\mathbf{x}|)$. These schemes are based on LWE and strong assumptions on pairing groups — either the generic (pairing) group model [4] or knowledge assumptions [3].

In this work, we set out to improve the state-of-the-art towards the direction of ideal efficiency. We observe that though there are ABE schemes for low-depth

[1] We always ignore polynomial factors in the security parameter.

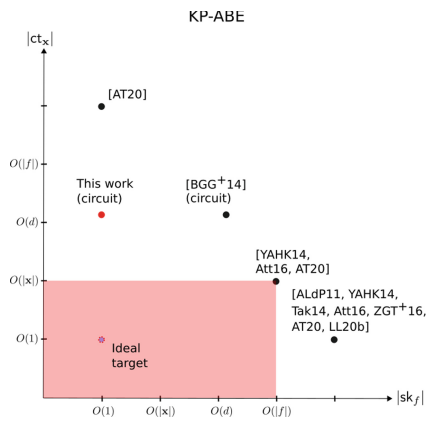



**Fig. 1.** Efficiency comparison for KP-ABE schemes. The pink region highlights succinctness for $|\mathsf{ct}_x|$ and $|\mathsf{sk}_f|$. This work and [BGG$^+$14] are KP-ABE schemes for circuits, while the rest of the schemes are for low-depth computation (Color figure online).

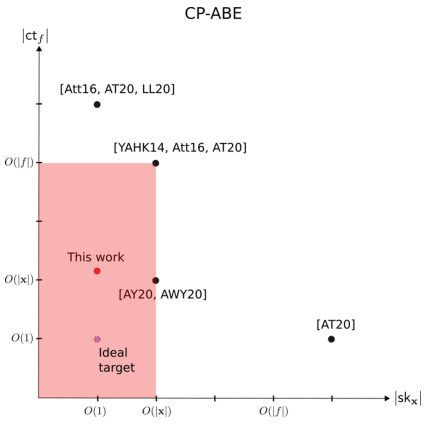



**Fig. 2.** Efficiency comparison for CP-ABE schemes. The pink region highlights succinctness for $|\mathsf{ct}_f|$ and $|\mathsf{sk}_x|$. All the included schemes are CP-ABE for low-depth computation (Color figure online).

computations with constant-size keys, we do not have such ABE for general circuits. We ask:

*Can we construct ABE for circuits with constant-size keys?*

Furthermore, all of the above schemes either have succinct keys or succinct ciphertexts, but never both at the same time. If we were to eventually achieve ideal efficiency, we would have to first overcome the intermediate barrier of simultaneously having succinct keys and ciphertexts — we refer to this as *double succinctness*. We thus ask:

*Can we construct ABE for expressive policies with* both *succinct keys* and *succinct ciphertexts?*

We note that the above questions are unanswered even when assuming the strong primitive of indistinguishability obfuscation (iO). Several works [17,18,27] constructed ABE for circuits (or even functional encryption for circuits) using indistinguishability obfuscation or related primitives. However, they all have large secret keys of size poly$(|f|)$. The only work that manages to obtain ABE for RAM with constant-size keys [20] rely on a strong primitive called extractable witness encryption, which however lacks provably secure instantiation.

**Our Results.** We address both questions. For the former, we construct the first KP-ABE scheme for circuits with constant-size keys while keeping the ciphertext size the same as in [11]. Concretely, each secret key consists of only 3 group

elements. For the latter, we present the first CP-ABE scheme for Boolean formulae achieving double succinctness — it has constant-size keys and succinct ciphertexts. Concretely, each secret key consists of only 2 group elements. Both constructions rely on LWE and the generic (pairing) group model, similar to [4].

**Theorem** (KP-ABE)**.** *Assuming LWE, in the generic (pairing) group model, there is a KP-ABE for circuits (Construction 2) that achieves selective security and has key size* $|\mathsf{sk}_C| = \mathrm{poly}(\lambda)$ *(concretely, containing 3 group elements) and ciphertext size* $|\mathsf{ct}_{\mathbf{x}}| = |\mathbf{x}|\,\mathrm{poly}(\lambda, d)$*, where* $d$ *is the maximum depth of the policy circuits.*

**Theorem** (CP-ABE)**.** *Assuming LWE, in the generic (pairing) group model, there is a CP-ABE for Boolean formulae [31] that achieves very selective security, and has constant-size keys* $|\mathsf{sk}_{\mathbf{x}}| = \mathrm{poly}(\lambda)$ *(concretely, containing 2 group elements) and ciphertexts of size* $|\mathsf{ct}_f| = |\mathbf{x}|^2\,\mathrm{poly}(\lambda)$ *independent of the formula size* $|f|$*.*

**Additional Contribution — Adaptive Security.** The standard security property of ABE is collusion resistance, which stipulates that no information of the message $\mu$ encrypted in a ciphertext should be revealed even when multiple secret keys are issued, as long as none of the keys alone is authorized to decrypt the ciphertext. Adaptive security requires collusion resistance to hold even when attributes and policies tied to the challenge ciphertext and the secret keys are chosen adaptively by the adversary. The weaker selective security restricts the adversary to commit to the attribute (in KP-ABE) or the policy (in CP-ABE) associated with the challenge ciphertext before seeing any parameters of the system, and very selective security further requires all attributes and policies in both the challenge ciphertext and the secret keys to be chosen statically.

Adaptive security guards against more powerful adversaries than selective security. It is known that the latter can be generically lifted to the former via complexity leveraging, at the cost of subexponential hardness assumptions. However, complexity leveraging is undesirable not only because it requires subexponential hardness, but also because it requires scaling the security parameter to be polynomial in the length of the information to be guessed, $\lambda = \mathrm{poly}(|x|)$ in KP-ABE or $\lambda = \mathrm{poly}(|f|)$ in CP-ABE. As a result, complexity leveraging is not a viable solution when aiming for constant-size keys, as key size $\mathrm{poly}(\lambda)$ would already depends on $|x|$ or $|f|$.

Instead, we show that in our constructions of KP- and CP-ABE, if assume adaptive LWE instead of plain LWE, then they achieve *adaptive security* and our reduction only incurs a polynomial amount of security loss. The adaptive LWE assumption [36] postulates that LWE samples of the form $\{\mathbf{s}^\intercal(\mathbf{A}_i - \mathbf{x}[i]\mathbf{G}) + \mathbf{e}_i^\intercal\}_i$ are pseudorandom, even if the adversary adaptively chooses $\mathbf{x}$ depending on the random matrices $\{\mathbf{A}_i\}_i$.

**Theorem** (adaptive security)**.** *Assuming the* polynomial hardness *of adaptive LWE (instead of LWE), in the generic (pairing) group model, the KP-ABE scheme (Construction 2) and the CP-ABE scheme [31] are adaptively secure.*

In the literature, the ABE schemes for circuits based on lattices [11,21] achieve only selective security (without complexity leveraging). Adapting it to have adaptive security has remained a technical barrier, except for very limited classes of policies such as 3-CNF [39]. Alternatively, there are schemes based on indistinguishability obfuscation or functional encryption for all circuits that are adaptively secure [27,40], but requiring stronger assumptions. Our technique can be viewed as making the lattice-based schemes adaptively secure when combined with pairing. Note that this is not trivial, for instance, the recent CP-ABE schemes in [3,4] that combine [11] with pairing groups inherit the selective security of the former (even if assuming adaptive LWE).

**Organization.** In Sect. 2, we present an overview of our techniques. In Sect. 3, we introduce preliminaries. In Sect. 4, we define nearly linear secret sharing schemes and non-annihilability, and construct such a scheme for bounded-depth circuits based on the adaptive LWE assumption. In Sect. 5, we present our adaptively secure KP-ABE for bounded-depth circuits. Due to space constraints, we refer the readers to the full version [31] for our doubly succinct CP-ABE as well as the secret sharing scheme and our new analysis of IPFE of [1] for that.

## 2 Technical Overview

**High-Level Ideas.** Let's focus on our KP-ABE scheme for circuits first. The first known construction of KP-ABE for circuits from LWE [22] has keys of size $|f|\,\mathrm{poly}(d,\lambda)$. The scheme of [11] reduces the key size to $\mathrm{poly}(d,\lambda)$. Both schemes achieve only selective security because they rely on the *lattice trapdoor simulation techniques*. Consider the BGG$^+$ scheme. Its ciphertext encodes the attributes $\mathbf{x}$ and message $\mu$ as follows.

$$\text{BGG}: \quad \mathbf{s}^\intercal\mathbf{A}+\mathbf{e}^\intercal, \quad \mathbf{s}^\intercal\Big(\overbrace{(\mathbf{A}_1||\cdots||\mathbf{A}_\ell)}^{\mathbf{B}}-\mathbf{x}\otimes\mathbf{G}\Big)+(\mathbf{e}')^\intercal, \quad \mathbf{s}^\intercal\mathbf{v}+e''+\mu\lfloor q/2\rceil.$$

One can homomorphically evaluate any circuit $f$ on the attribute encoding to obtain $\mathbf{s}^\intercal(\mathbf{B}_f-f(x)\mathbf{G})+\mathbf{e}_f^\intercal$. To decrypt, the secret key $\mathsf{sk}_f$ simply is a *short* vector $\mathbf{r}_{\mathbf{A},f}$ satisfying $(\mathbf{A}||\mathbf{B}_f)\mathbf{r}_{\mathbf{A},f}=\mathbf{v}$, which can be sampled using a trapdoor $\mathbf{T}_\mathbf{A}$ for $\mathbf{A}$. This approach however has two drawbacks:

- Difficulty towards Constant-Size Keys. The short vector $\mathbf{r}_{\mathbf{A},f}$ contained in the secret key $\mathsf{sk}_f$ has size $\mathrm{poly}(d,\lambda)$. This is because it has dimension $m=n\log q$ for $\log q=\mathrm{poly}(d,\lambda)$ and entries of magnitude exponential in $d$.
- Difficulty towards Adaptive Security. The security proof relies on the ability to simulate trapdoors for these matrices $\mathbf{A}||\mathbf{B}_f$ corresponding to secret keys that are *unauthorized* to decrypt the challenge ciphertext with attribute $\mathbf{x}^*$, that is $f(\mathbf{x}^*)=1$. However, to do so, current technique plants $\mathbf{x}^*$ in the public matrices $\mathbf{A}_i$'s (contained in $\mathsf{mpk}$), leading to selective security. Note that even with the stronger adaptive LWE assumption, it is unclear how to simulate these trapdoors in another way.

Towards constant-size keys and adaptive security, our construction circumvents the use of lattice trapdoors all together. At a high level, we turn attention to a much weaker lattice primitive called attribute-based laconic function evaluation (AB-LFE) [36], and lifts it to a KP-ABE scheme for circuits using pairing. AB-LFE is an interactive protocol where a receiver sends a digest of a function, which is exactly the matrix $\mathbf{B}_f$ in BGG. The sender then encodes the attribute $\mathbf{x}$ and message $\mu$ as follows.

$$\text{AB-LFE}: \quad \mathbf{s}^\intercal\big((\mathbf{A}_1||\cdots||\mathbf{A}_\ell)-\mathbf{x}\otimes\mathbf{G}\big)+(\mathbf{e}')^\intercal, \quad \mathbf{s}^\intercal\mathbf{B}_f\mathbf{r}+e''+\mu\lfloor q/2\rceil.$$

where $\mathbf{r}=\mathbf{G}^{-1}(\mathbf{a})$ is the bit decomposition of a random vector $\mathbf{a}$. Security guarantees that the encoding reveals only the output $f(\mathbf{x})$. At a first glance, the LFE encoding appears the same as BGG, but the novelty is in details. Since the LFE encoding depends on $\mathbf{B}_f$ (and hence $f$), it can be generated without using lattice trapdoors — the short vector $\mathbf{r}$ sampled first, and $\mathbf{B}_f\mathbf{r}$ computed next. When $f(x)=1$, the hiding of $\mu$ follows directly from the pseudorandomness of LWE samples $\mathbf{s}^\intercal((\mathbf{A}_1||\cdots||\mathbf{A}_\ell)-\mathbf{x}\otimes\mathbf{G}))+\mathbf{e}'$ and $\mathbf{s}^\intercal\mathbf{a}+e'''$. When $\mathbf{x}$ is adaptively chosen, security follows naturally from adaptive LWE.

However, AB-LFE is able to avoid lattice trapdoor only because it is significantly weaker than ABE, or even 1-key ABE: *1)* its message encoding depends on $\mathbf{B}_f$ (unknown at ABE encryption time), and *2)* it is only secure for a single function. Our next challenge is lifting AB-LFE back to full ABE, for which we use pairing.

More specifically, we first modify the AB-LFE scheme of [36] to obtain a *nearly linear secret sharing scheme* for circuits. It contains two parts.

$$\begin{aligned}\text{Our LSS encoding:} \quad & L_{\mathbf{x}}=\mathbf{s}^\intercal\big((\mathbf{A}_1||\cdots||\mathbf{A}_\ell)-\mathbf{x}\otimes\mathbf{G}\big)+(\mathbf{e}')^\intercal \bmod q,\\ & L_f=\mathbf{s}^\intercal\mathsf{Round}(\mathbf{B}_f\mathbf{r})+e''+\mu\lfloor p/2\rceil \bmod p.\end{aligned}$$

Note that we round $\mathbf{B}_f\mathbf{r}$ from modulus $q$ of $\mathrm{poly}(d)$ length to $p$ of $\mathrm{poly}(\lambda)$ length so that the component $L_f$ in the secret sharing that depends on $f$ and $\mu$ has *constant size*, which is the key towards constant-size ABE keys. To solve the problem that $L_f$ requires knowledge of $\mathbf{B}_f$ unknown at encryption time, we use a pairing-based inner-product functional encryption (IPFE) to compute $L_f$ in the exponent, by viewing it as inner product $L_f=\langle\mathbf{s}^\intercal||\mu\lfloor p/2\rceil,\mathsf{Round}(\mathbf{B}_f\mathbf{r})||1\rangle$, where the two vectors are known respectively at ABE encryption and key generation time. To overcome that AB-LFE only guarantees security for a single $L_f$. We follow the idea of [3,4] to compute $\delta_f\cdot L_f$ in the exponent instead, where $\delta_f$ is an independent and random scalar chosen at key generation time. In GGM, the presence of $\delta_f$ prevents adversaries from meaningfully "combining" information from multiple $L_f$ for different $f$.

**Comparison with [3,4].** Our way of combining lattice-based LSS with pairing-based IPFE differs from that of [3,4], in order to address unique technical difficulties. To start with, they use an LSS scheme based on the BGG ABE and inherits the selective security. Second, our KP-ABE scheme reveals part of the secret h $L_{\mathbf{x}}$ *in the clear* (in ciphertext), and only compute $L_f$ in the exponent, whereas [3,4]

computes the entire LSS in the exponent. This is because the decryptor needs to perform the non-linear rounding operation on the result of homomorphic evaluation on $L_{\mathbf{x}}$, in order to obtain $\mathsf{Round}(\mathbf{s}^\intercal(\mathbf{B}_f - f(x)\mathbf{G})\mathbf{r} + \mathbf{e}_f^\intercal)$ for decryption. Keeping $L_{\mathbf{x}}$ in the clear allows rounding, but renders security harder to prove.

Furthermore, the security proof of AB-LFE relies on noise flooding — their technique can only show that $L_f + \tilde{e}$ is secure for a super-polynomially large $\tilde{e}$. But noise flooding is incompatible with computing $L_f$ in the exponent, since we must keep noises polynomially small in order for decryption to be efficient (which performs discrete logarithm). Without noise flooding, we cannot prove that unauthorized shares are pseudorandom as in [36]. Nevertheless, we show that unauthorized shares are "entropic", captured by a new notion called *non-annihilability*, and that the "entropic" $L_f$ computed in the exponent still hides the message $\mu$. The proof of non-annihilability combines techniques from AB-LFE and leakage simulation [16,25]. The work of [3,4] does not encounter issues with super-polynomial noises.

We add a note on our doubly succinct CP-ABE for Boolean formulae. It is closer to the CP-ABE scheme of [3,4]. However, to obtain constant-size keys, we rely on an IPFE scheme with strong (selective) simulation security — it enables simultaneously simulating a polynomial number $k$ of ciphertexts, by programming $k$ inner products for every secret key, while keeping the secret key *constant-size* (independent of $k$). Such strong simulation is impossible in the standard model following an incompressibility argument. We show that this is possible in GGM, in particular, the IPFE scheme of [1] satisfies it. IPFE with such strong simulation may find other applications.

Next, we explain our ideas in more details.

**Combining LSS with IPFE.** An IPFE scheme enables generating keys $\mathsf{isk}(\mathbf{v}_j)$ and ciphertexts $\mathsf{ict}(\mathbf{u}_i)$ associated with vectors $\mathbf{v}_j, \mathbf{u}_i \in \mathbb{Z}_p^N$ such that decryption reveals only their inner products $\langle \mathbf{u}_i, \mathbf{v}_j \rangle$ and hides all other information about $\mathbf{u}_i$ encrypted in the ciphertexts (whereas $\mathbf{v}_j$ associated with the keys are public). It can be based on a variety of assumptions such as MDDH, LWE, or DCR [1,2].

A *nearly* linear secret sharing scheme enables generating shares $L_f, L_0, \{L_i^b\}$ associated with a policy $f$ and some secret $\mu$, such that for any input $\mathbf{x} \in \{0,1\}^\ell$, its corresponding subset of shares $L^{\mathbf{x}} = (L_0, \{L_i^{\mathbf{x}[i]}\})$, together with $L_f$ can be used to *approximately* reconstruct the secret $\mu$ if and only if $f(\mathbf{x}) = 0$:

$$(L_f, L_0, \{L_i^b\}_{i\in[\ell],b\in\{0,1\}}) \leftarrow \mathsf{Share}(f, \mu; \mathbf{r})$$
$$f(x) = 0 \implies \mu \approx \mathsf{Recon}(f, \mathbf{x}, L_f, L^{\mathbf{x}}).$$

Near linearity means that $\mathsf{Recon}$ is linear in the shares $L_f, L^{\mathbf{x}}$ and that its output is close to the secret $\mu$.

How can we combine these two primitives to construct a KP-ABE? We require $L_0, \{L_i^b\}$ to be independent of $f$ and $\mu$, and $L_f$ to be linear in $\mu$ and the randomness $\mathbf{r}$ of $\mathsf{Share}$. The first requirement allows us to simply put $L^{\mathbf{x}}$ in the

ciphertext. The second requirement allows us to encode $\mu, \mathbf{r}$ into $\mathsf{ict}$'s and the coefficients (of $L_f$ as a function of $\mu, \mathbf{r}$) into $\mathsf{isk}$'s, so that their inner product is exactly $L_f$. For convenience, we write $[\![x]\!]_i$ for $g_i^x$ and use additive notation for the groups. The idea is as follows:

$$\left.\begin{array}{lll} \mathsf{kp.sk}_f: & [\![\delta]\!]_2, & \mathsf{isk}(\text{coefficients of } \delta L_f) \\ \mathsf{kp.ct}_{\mathbf{x}}: & L^{\mathbf{x}}, & \mathsf{ict}(\mu, \mathbf{r}) \end{array}\right\} \quad [\![\delta L_f]\!]_{\mathrm{T}} \text{ and } L^{\mathbf{x}}. \tag{1}$$

If $f(\mathbf{x}) = 0$, the linear reconstruction can be carried out in the exponents to approximately obtain $[\![\delta\mu]\!]_{\mathrm{T}}$. Decryption enumerates all possible errors to recover $\mu$ exactly. We stress again that different from [3,4], we keep $L_{\mathbf{x}}$ in the clear (in the ciphertext), instead of computing the entire secret sharing $L_{\mathbf{x}}, L_f$ in the exponent, which is important for achieving constant-size keys, but makes proving security more difficult.

We construct a secret sharing scheme that features $L_f$ of *constant size*, which translates to KP-ABE with constant-size secret keys.

Combining secret sharing and IPFE to construct CP-ABE is similar. We can encode $L_0, \{L_i^b\}$ in $\mathsf{ict}$'s, and a "selection" vector according to $\mathbf{x}$ in $\mathsf{isk}$'s, so that their inner products are exactly $L^{\mathbf{x}}$:

$$\left.\begin{array}{lll} \mathsf{cp.sk}_{\mathbf{x}}: & [\![\delta]\!]_2, & \mathsf{isk}(\delta \cdot \text{ selection vector for } \mathbf{x}) \\ \mathsf{cp.ct}_f: & [\![L_f]\!]_1, & \mathsf{ict}(L_0, \{L_i^b\}). \end{array}\right\} \quad [\![\delta L_f]\!]_{\mathrm{T}} \text{ and } [\![\delta L^{\mathbf{x}}]\!]_{\mathrm{T}}. \tag{2}$$

We use an IPFE scheme with secret keys of constant size, independent of the vector dimension or the number of ciphertexts, and a secret sharing scheme whose $L_f, L^{\mathbf{x}}$ grows only with the input length $|\mathbf{x}|$. This translates to CP-ABE with double succinctness.

**Lattice-Based Nearly Linear Secret Sharing.** The BGG$^+$ ABE scheme introduces an important homomorphic evaluation procedure: Given public matrices $\mathbf{B} = (\mathbf{A}_1 || \cdots || \mathbf{A}_{|\mathbf{x}|})$, and the following encoding of an input $\mathbf{x}$, one can homomorphically evaluate any circuit $f$ on the encodings to obtain an encoding of the output.

$$\begin{aligned} \mathbf{c}^\intercal &= \mathbf{s}^\intercal(\mathbf{B} - (1, \mathbf{x}) \otimes \mathbf{G}) + \mathbf{e}_2^\intercal, \\ \mathsf{EvalCX}(\mathbf{c}_2, f, \mathbf{x}) = \mathbf{c}_f^\intercal &= \mathbf{s}^\intercal(\mathbf{B}_f - f(\mathbf{x})\mathbf{G}) + \mathbf{e}_f^\intercal, \text{ where } \mathsf{EvalC}(\mathbf{B}, f) = \mathbf{B}_f. \end{aligned} \tag{3}$$

As discussed before, the BGG$^+$ ABE scheme uses lattice trapdoor simulation technique, which we try to avoid in order to get constant-size key and adaptive security.

We hence turn to using the weaker primitive of AB-LFE scheme introduced by [36]. It is a two-party protocol between a sender and a receiver who share the LWE public matrix $\mathbf{B}$ as the common reference string. The receiver first computes a digest $\mathbf{B}_f = \mathsf{EvalC}(\mathbf{B}, f)$ for a function $f$ and sends it to the sender.

Upon receiving the digest, the sender masks a message $\mu$ by an LWE sample $c_0 = \mathbf{s}^\intercal \mathbf{v}_f + e + \mu\lfloor q/2 \rceil$, where $\mathbf{r} = \mathbf{G}^{-1}(\mathbf{a})$ and $\mathbf{v}_f = \mathbf{B}_f\mathbf{r}$ are analogues of $\mathbf{r}_{\mathbf{A},f}$ and $\mathbf{v}$ in BGG$^+$. It also encodes an attribute $\mathbf{x}$ into LWE samples $\mathbf{c}_1$ as described below.

$$\begin{array}{l|l} \text{AB-LFE.crs} : \mathbf{B} & \text{AB-LFE.ct}_{f,\mathbf{x}}(\mu): \\ \text{AB-LFE.digest} : \mathbf{B}_f = \mathsf{EvalC}(\mathbf{B}, f) & \mathbf{a} \xleftarrow{\$} \mathbb{Z}_q^n \\ & c_0 = \mathbf{s}^\intercal \underbrace{\mathbf{B}_f\mathbf{G}^{-1}(\mathbf{a})}_{\mathbf{v}_f} + \mu\lfloor q/2 \rceil + e \\ & \mathbf{c}_1^\intercal = \mathbf{s}^\intercal(\mathbf{B} - (1, \mathbf{x}) \otimes \mathbf{G}) + \mathbf{e}_1^\intercal \end{array}$$

To decrypt, first run $\mathsf{EvalCX}(\mathbf{c}_1, f, \mathbf{x})$ to obtain $\mathbf{c}_f = \mathbf{s}^\intercal(\mathbf{B}_f - f(\mathbf{x})\mathbf{G}) + \mathbf{e}_f^\intercal$. If $f(\mathbf{x}) = 0$, the decryptor can compute $c_0 - \mathbf{c}_f^\intercal\mathbf{r} = \mu\lfloor q/2 \rceil + (e - \mathbf{e}_f^\intercal\mathbf{r})$ and round it to recover $\mu$.

Observe that the above scheme can be viewed as a nearly linear secret sharing scheme, where the shares chosen by $\mathbf{x}$ are exactly $\mathbf{L}^{\mathbf{x}} = \mathbf{c}_1$ and the shares dependent on $f$ and $\mu$ is $L_f = c_0$. At the moment, the bit-length of $L_f$ is $\Theta(\log q)$. Since the noise growth during the homomorphic evaluation is exponential to the depth of the computation, $q$ is a $\mathrm{poly}(d, \lambda)$-bit modulus in order to accommodate for the noise growth. We next turn to reducing the size of $L_f$ to a constant independent of $d$.

**Rounding to Make $L_f$ Constant-Size.** Since the encrypted message is only a single bit, we can afford to lose a lot of precision in the above decryption process. In particular, the scheme is still correct if we round down the digest $\mathbf{B}_f$ to a much smaller, depth-independent, modulus $p \ll q$, and change $c_0$ to use the rounded digest (while keeping $\mathbf{c}_1^\intercal$ unchanged):

$$c_0' = \mathbf{s}^\intercal \lfloor \mathbf{B}_f\mathbf{G}^{-1}(\mathbf{a}) \rceil_p + \mu\lfloor p/2 \rceil + e \qquad \text{over } \mathbb{Z}_p.$$

During decryption, one now computes, over $\mathbb{Z}_p$,

$$\begin{aligned} c_0' - \lfloor \mathbf{c}_f^\intercal\mathbf{G}^{-1}(\mathbf{a}) \rceil_p &= c_0' - \lfloor \mathbf{s}^\intercal\mathbf{B}_f\mathbf{G}^{-1}(\mathbf{a}) + f(\mathbf{x})\mathbf{s}^\intercal\mathbf{a} + \mathbf{e}_f^\intercal\mathbf{G}^{-1}(\mathbf{a}) \rceil_p \\ &= c_0' - \big(\mathbf{s}^\intercal \lfloor \mathbf{B}_f\mathbf{G}^{-1}(\mathbf{a}) \rceil_p + f(\mathbf{x})\lfloor \mathbf{s}^\intercal\mathbf{a} \rceil_p + \underbrace{\lfloor \mathbf{e}_f^\intercal\mathbf{G}^{-1}(\mathbf{a}) \rceil_p}_{e_f'} + e_s\big) \\ &= \mu\lfloor p/2 \rceil - f(\mathbf{x})\lfloor \mathbf{s}^\intercal\mathbf{a} \rceil_p + (e - e_f' - e_s), \end{aligned} \tag{4}$$

where the rounding error $e_s$ is of magnitude $|e_s| = \Theta(\|\mathbf{s}\|_1)$. As long as the error terms are much smaller than $p/2$, when $f(\mathbf{x}) = 0$, one can still recover $\mu$. We can now recast the above rounded AB-LFE scheme into a secret sharing scheme with $L_f$ of bit-length $\Theta(\log p)$, independent of depth $d$. (Only the larger modulus $q$ will, and thus $\mathbf{e}_f$ itself can, grow with $d$.)

$$\begin{array}{rll} \mathsf{SS.pp}: & \mathbf{a}, \mathbf{B} & \bmod q; \\ L_f: & c_0' = \mathbf{s}^\intercal \lfloor \mathbf{B}_f\mathbf{G}^{-1}(\mathbf{a}) \rceil_p + \mu\lfloor p/2 \rceil + e & \bmod p \ll q; \\ \mathbf{L}^{\mathbf{x}}: & \mathbf{c}_1^\intercal = \mathbf{s}^\intercal\big(\mathbf{B} - (1, \mathbf{x}) \otimes \mathbf{G}\big) + \mathbf{e}_1^\intercal & \bmod q. \end{array}$$

As shown in Eq. (1), to obtain KP-ABE, we will use a pairing-based IPFE to compute $L_f$ in the exponent. Specifically, the IPFE secret key $\mathsf{isk}$ encodes $(\lfloor \mathbf{B}_f \mathbf{G}^{-1}(\mathbf{a}) \rceil_p, \lfloor p/2 \rceil)$, and the IPFE ciphertext $\mathsf{ict}$ encodes $(\mathbf{s}^\intercal, \mu)$. Together, they decrypt to exactly $[\![L_f]\!]_\mathrm{T}$. Since both vectors live in $\mathbb{Z}_p$, the KP-ABE key, consisting of only $\mathsf{isk}$, is of size independent of $d$. Our secret sharing scheme is summarized below. It turns out that arguing security is actually tricky and requires additional modification.

**Our Secret Sharing Scheme for KP-ABE**

$\mathsf{Setup}(1^\lambda)$ : $\mathsf{pp} = \mathsf{LFE.pp} = (\mathbf{a}, \mathbf{B}) = (\mathbf{a}, \mathbf{A}_0, \mathbf{A}_1, \ldots, \mathbf{A}_\ell)$.

$\mathsf{ShareX}(\mathsf{pp})$ : Compute LWE samples

$$\mathbf{L}_0 = \mathbf{s}^\intercal \mathbf{A}_0 + \mathbf{e}_0, \quad \mathbf{L}_i^b = \mathbf{s}^\intercal (\mathbf{A}_i - b\mathbf{G}) + \mathbf{e}_i^\intercal.$$

Output $(\mathbf{L}_0, \{\mathbf{L}_i^b\}, \mathbf{s})$.

$\mathsf{ShareF}(\mathsf{pp}, f, \mu, \mathbf{s})$ : Compute $\mathbf{B}_f = \mathsf{EvalC}(\mathbf{B}, f)$.

Output $L_f = \mathbf{s}^\intercal \lfloor \mathbf{B}_f \mathbf{G}^{-1}(\mathbf{a}) \rceil_p + \mu \lfloor p/2 \rceil$.

$\forall \mathbf{x} \in \{0,1\}^\ell$ : $\mathbf{L}^\mathbf{x} = \left(\mathbf{L}_0, \mathbf{L}_1^{\mathbf{x}[1]}, \ldots, \mathbf{L}_\ell^{\mathbf{x}[\ell]}\right)$

$\mathsf{Recon}(\mathsf{pp}, f, L_f, \mathbf{x}, \mathbf{L}^\mathbf{x})$ : If $f(\mathbf{x}) = 1$, output $\perp$.

Otherwise, compute $\mathbf{c}_f = \mathsf{EvalCX}(\mathbf{L}^\mathbf{x}, f, \mathbf{x})$, and recover $\mu$ from $L_f - \lfloor \mathbf{c}_f^\intercal \mathbf{G}^{-1}(\mathbf{a}) \rceil_p \approx \mu \lfloor p/2 \rceil$.

**Non-Annihilability by Leakage Simulation.** However, using AB-LFE creates a further complication, as its security relies on flooding the $e'_f, e_s$ terms (which may contain information of $\mathbf{s}$ and $\mathbf{x}$) with $e$, in order to prove pseudorandomness of $L_f$. By Eqs. (3, 4), when $f(\mathbf{x}) = 1$ we have

$$\begin{aligned} L_f = {} & \lfloor \mathsf{EvalCX}(\mathbf{L}^\mathbf{x}, f, \mathbf{x})^\intercal \mathbf{G}^{-1}(\mathbf{a}) \rceil_p \\ & - \lfloor \mathbf{s}^\intercal \mathbf{a} + e_a \rceil_p + \mu \lfloor p/2 \rceil + (e - e'_f - e_s). \end{aligned} \tag{5}$$

Observe that in the above, for later convenience, an additional polynomial LWE noise $e_a$ is introduced in the term $\lfloor \mathbf{s}^\intercal \mathbf{a} + e_a \rceil_p$ (which by rounding simply equals to $\lfloor \mathbf{s}^\intercal \mathbf{a} \rceil_p$).

At this point, in order to show that $L_f$ is pseudorandom, given that $\mathbf{x}$ is selected before $\mathsf{Setup}$, one could program the public matrices as $\mathbf{A}_i = \mathbf{A}'_i + x_i \mathbf{G}$ according to $\mathbf{x}$, where $\mathbf{B}' = (\mathbf{A}'_0, .., \mathbf{A}'_\ell)$ are sampled at random. And one would hope to apply LWE to argue that

$$\mathbf{L}_\mathbf{x} = \mathbf{s}^\intercal (\mathbf{B} + (1, \mathbf{x}) \otimes \mathbf{G}) + \mathbf{e}_1^\intercal = \mathbf{s}^\intercal \mathbf{B}' + \mathbf{e}_1^\intercal,$$

and $(\mathbf{s}^\intercal \mathbf{a} + e_a)$ are jointly pseudorandom. However, the noise terms $e'_f$ and $e_s$ may leak information about $\mathbf{e}_2$ and $\mathbf{s}$.

The solution in [36] is noise flooding. By setting $e$ to be super-polynomially larger than $(e'_f + e_s)$, we have $e - e'_f - e_s \approx_s e$. By LWE, we can now switch $\mathbf{L}^{\mathbf{x}}$ and $(\mathbf{s}^\intercal \mathbf{a} + e_a)$ to random and conclude that $L_f$ is pseudorandom.

However, the unique challenge here is that $L_f$ is going to be computed in the exponent of the pairing group, and decryption only recovers $(\mu \lfloor p/2 \rceil + e)$ in the exponent. When $e$ is super-polynomial, we can no longer extract $\mu$ out of the exponent. Our solution is avoiding flooding altogether and remove the noise $e$ from $L_f$. As such, we cannot prove pseudorandomness of $L_f$, but only a weaker security notion that we call non-annihilability (for $L_f$). This notion captures that $L_f$ is still entropic.

*Non-Annihilability.* Non-annihilability requires that no adversary, after seeing $\mathbf{L}^{\mathbf{x}}$ (but not $L_f$) can come up with an affine function $\gamma$ such that $\gamma(L_f) = 0$. As we will see, this security notion, combined with GGM, suffices for our proof.

Towards proving non-annihilability, we want to show that $L_f$ is highly entropic (even without $e$). Our idea is to view the noises $e'_f, e_s$ as leakage of the randomness that generates $\mathbf{L}^{\mathbf{x}}$ and $(\mathbf{s}^\intercal \mathbf{a} + e_a)$ as well as the other information, and simulate $e'_f, e_s$ using leakage simulation [16,25]. Crucially, because $e'_f, e_s$ have polynomial range, the simulation can run in polynomial time. More precisely, the leakage simulation lemma of [16] states that for any joint distribution $(X, Z) \sim \mathcal{D}$ ($Z$ viewed as leakage of randomness for generating $X$), adversary size bound $s$, and error bound $\epsilon$, there is a simulator $h$ simulating $Z$ as $h(X)$ such that $(X, Z)$ and $(X, h(X))$ are $(s, \epsilon)$-indistinguishable. Furthermore, the running time of $h$ is $\mathrm{O}(s\epsilon^{-2}2^{|Z|})$. Suppose for contradiction that there is an adversary $\mathcal{A}$ of size $s = \mathrm{poly}(\lambda)$ winning the non-annihilability game with probability $2\epsilon \geq 1/\mathrm{poly}(\lambda)$. Consider the joint distribution $\mathcal{D}$ of running the game with $\mathcal{A}$, defined in the first line below:

$$
\begin{aligned}
\mathcal{D} &\to \{X = (\mathsf{pp}, \mathbf{x}, \mathbf{L}^{\mathbf{x}}, f, \mu, \gamma, \psi = \mathbf{s}^\intercal \mathbf{a} + e_a), && Z = e'_f + e_s\} \\
&\overset{s,\epsilon}{\approx} \text{Hybrid 1} \to \{X = (\mathsf{pp}, \mathbf{x}, \mathbf{L}^{\mathbf{x}}, f, \mu, \gamma, \psi = \mathbf{s}^\intercal \mathbf{a} + e_a), && Z = h(X)\} \\
&\approx \text{Hybrid 2} \to \{X = (\mathsf{pp}, \mathbf{x}, \mathbf{L}^{\mathbf{x}} \text{ random}, f, \mu, \gamma, \psi \text{ random}), && Z = h(X)\}.
\end{aligned}
$$

Using $(X, Z)$, one can emulate $L_f$ as (cf. Eq. (5) with $e$ removed and $(e'_f + e_s)$ replaced by $Z$)

$$
L_f = \lfloor \mathsf{EvalCX}(\mathbf{L}^{\mathbf{x}}, f, \mathbf{x}) \mathbf{G}^{-1}(\mathbf{a}) \rceil_p - \lfloor \psi \rceil_p + \mu \lfloor p/2 \rceil - Z.
$$

Since $Z = e'_f + e_s$, and $s, \epsilon^{-1}$ are all polynomially bounded, we can simulate $Z$ by $h(X)$ in polynomial time (Hybrid 1). Now, we can apply LWE to switch $\mathbf{L}^{\mathbf{x}}$, $\psi = \mathbf{s}^\intercal \mathbf{a} + e_a$ to random (Hybrid 2). At this point, it seems that $L_f$ is just pseudorandom by the pseudorandomness of $\psi$. However, there is a subtle issue: $Z = h(X)$ depends on $\psi$ contained in $X$, and hence $(\lfloor \psi \rceil_p - Z)$ may not be pseudorandom, and neither may be $L_f$. Despite this dependency, thanks again to $(e'_f + e_s)$, thus $h(X)$, being polynomially bounded, $(-\lfloor \psi \rceil_p + h(X))$ still has almost full entropy (up to a logarithmic loss). Therefore, the probability that $L_f$ is annihilated by an affine function $\gamma$ chosen by $\mathcal{A}$ before $\psi$ is randomly sampled is negligible. This gives a contradiction and concludes the proof of non-annihilability.

**Multi-Key Security of KP-ABE in GGM.** Our KP-ABE scheme combines an IPFE scheme with the secret sharing scheme described above. As described before, in our KP-ABE scheme, we only compute the $L_f$ part of secret sharing using IPFE, and leave the $\mathbf{L}^{\mathbf{x}}$ part in the clear so that rounding can be performed. To achieve multi-key security, we further employ the idea from [3,4] to "isolate" each ABE secret key in GGM by multiplying it with a fresh random element $\delta$.

$$\left.\begin{array}{lll} \mathsf{kp.sk}: & [\![\delta]\!]_2, & \mathsf{isk}\big([\![\delta(\lfloor \mathbf{B}_f\mathbf{G}^{-1}(\mathbf{a})\rceil_p, \lfloor p/2\rceil)]\!]_2\big) \\ \mathsf{kp.ct}: & \mathbf{L}^{\mathbf{x}}, & \mathsf{ict}([\![(\mathbf{s},\mu)]\!]_1) \end{array}\right\} \text{ decrypt to } [\![\delta L_f]\!]_{\mathrm{T}}.$$

The decryption algorithm first computes IPFE decryption to recover $[\![\delta L_f]\!]_{\mathrm{T}}$. It then computes (homomorphically in the exponent of $g_{\mathrm{T}}$)

$$[\![\delta L_f]\!]_{\mathrm{T}} - [\![\delta]\!]_{\mathrm{T}}\lfloor \mathbf{c}_f^{\mathsf{T}}\mathbf{G}^{-1}(\mathbf{a})\rceil_p = [\![\delta\big(\mu\lfloor p/2\rceil - (e'_f + e_s)\big)]\!]_{\mathrm{T}}.$$

Since the noise $(e'_f + e_s)$ has a polynomial range, the decryption algorithm enumerates all its possible values to recover $\mu$.

Multi-key security, at a high level, relies on the fact that in GGM, an adversary can only learn information about $[\![\delta L_f]\!]_{\mathrm{T}}$ by submitting zero-test queries of affine functions. When the adversary attacks multiple keys, it essentially submits zero-test queries over the terms $\{\delta_j L_{f_j}\}$. Let $\gamma(\{\delta_j L_{f_j}\})$ be any zero-test query submitted by $\mathcal{A}$, we can view it as a degree-1 polynomial over $\delta_j$'s:

$$\gamma(\{\delta_j L_{f_j}\}) = \sum_j \gamma_j(L_{f_j})\delta_j + \gamma_0,$$

where $\gamma_j(L_{f_j})$ is the coefficient of $\delta_j$. Since each $\delta_j$ is sampled independently at random, by Schwartz–Zippel, with all but negligible probability, $\gamma$ evaluates to zero only if all $\gamma_j$'s evaluate to zero. In other words, the adversary is effectively constrained to annihilate each $L_{f_j}$ individually. By the non-annihilability for $L_f$, if $\gamma_j$ is not the zero function, it evaluates to non-zero with overwhelming probability. Hence the adversary learns no information of each $L_{f_j}$ and the message $\mu$ encoded in them.

**Our KP-ABE Scheme**

$\mathsf{Setup}(1^\lambda)$ : Output $\mathsf{mpk} = \mathsf{impk}$ for IPFE, $\mathsf{pp}$ for secret sharing and $\mathsf{msk} = \mathsf{imsk}$ for IPFE.

$\mathsf{KeyGen}(\mathsf{msk}, C)$ : Sample $\delta \xleftarrow{\$} \mathbb{Z}_p$ and compute $\mathbf{B}_f = \mathsf{EvalC}(\mathbf{B}, f)$.
Output $\mathsf{sk} = ([\![\delta]\!]_2, \mathsf{isk}([\![\delta(\lfloor \mathbf{B}_f\mathbf{G}^{-1}(\mathbf{a})\rceil_p, \lfloor p/2\rceil)]\!]_2))$.

$\mathsf{Enc}(\mathsf{mpk}, \mathbf{x}, \mu)$ : Compute $(\mathbf{L}_0, \{\mathbf{L}_i^b\}, \mathbf{s}) \xleftarrow{\$} \mathsf{ShareX}(\mathsf{pp})$.
Output $\mathsf{ct} = (\mathbf{L}^{\mathbf{x}}, \mathsf{ict}([\![\mathbf{s}, \mu]\!]_1))$.

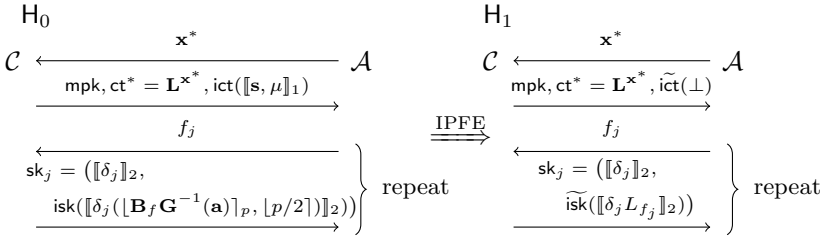
$\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}, C, \mathsf{ct}, \mathbf{x})$ : Run IPFE decryption to recover $[\![\delta L_f]\!]_{\mathrm{T}}$.
Compute $\mathbf{c}_f = \mathsf{EvalCX}(\mathbf{L}^{\mathbf{x}}, f, \mathbf{x})$ and find $\mu$ from
$[\![\delta L_f]\!]_{\mathrm{T}} - [\![\delta]\!]_{\mathrm{T}}\lfloor \mathbf{c}_f^{\mathsf{T}}\mathbf{G}^{-1}(\mathbf{a})\rceil_p = [\![\delta(\mu\lfloor p/2\rceil - e_s - e'_f)]\!]_{\mathrm{T}}$.

**Summary of Our KP-ABE.** Combining the above secret sharing scheme with an IPFE scheme, we obtain a KP-ABE scheme for bounded-depth circuits as summarized above.

We note that our KP-ABE scheme achieves the same asymptotic ciphertext compactness as the BGG$^+$ scheme. Let $d$ be an upper bound on the depth of the policy $f$, then $|\mathsf{ct}| = \mathrm{poly}(\lambda, d)|\mathbf{x}|$. The secret keys of our scheme contains only $\mathrm{O}(1)$ group elements, in fact only *three* using the IPFE scheme of [2] in a group of order $p$. We set $\log p = \mathrm{poly}(\lambda)$ and hence obtain constant size keys.

**Security Sketch for KP-ABE.** Finally, for completeness, we add a security sketch that puts the previous ideas together. We emphasize that we only use GGM in the last argument, when we need to isolate the share $L_{f_j}$ for each $f$.

The selective security game of ABE (summarized in $\mathsf{H}_0$ below) at a high level is as follows: The adversary $\mathcal{A}$ first decides a challenge attribute $\mathbf{x}^*$ before receiving a master public key $\mathsf{mpk}$ and a ciphertext $\mathsf{ct}^*$ from the challenger $\mathcal{C}$. It is then allowed to repeatedly query secret keys $\mathsf{sk}_j$ for functions $f_j$. The adversary wins if every queried function $f_j$ satisfies $f_j(\mathbf{x}^*) \neq 0$, and if it guesses the encrypted bit $\mu$ correctly.




Note that we can generate the IPFE ciphertext $\mathsf{ict}([\![\mathbf{s}, \mu]\!]_1)$ before any IPFE secret keys $\mathsf{isk}([\![\delta(\lfloor \mathbf{B}_{f_j}\mathbf{G}^{-1}(\mathbf{a})\rceil_p, \lfloor p/2\rceil)]\!]_2)$. Relying on the selective simulation security of IPFE, we can (as summarized in $\mathsf{H}_1$ above) replace $\mathsf{ict}([\![\mathbf{s}, \mu]\!]_1)$ with a simulated ciphertext $\widetilde{\mathsf{ict}}(\perp)$, and each $\mathsf{isk}([\![\delta_j(\lfloor \mathbf{B}_{f_j}\mathbf{G}^{-1}(\mathbf{a})\rceil_p, \lfloor p/2\rceil)]\!]_2)$ with a simulated secret key $\widetilde{\mathsf{isk}}([\![\delta_j L_{f_j}]\!]_2)$ using their inner products.

In GGM, we can now argue that $\mathcal{A}$ only learns information about $\mu$ through zero-test queries over $\{\delta_j L_{f_j}\}$. As argued before, by the non-annihilability of $L_f$, the adversary learns no information of $\mu$.

**Building Doubly Succinct CP-ABE.** To build a CP-ABE scheme we need a different secret sharing construction, because the previous rounding solution does not work anymore. As described in Eq. (2), in the CP case, we use IPFE to compute $\mathbf{L}^{\mathbf{x}}$ in the exponent, hence cannot perform rounding on it. Without rounding, the $\mathbf{e}_f$ term, as a result of $\mathsf{EvalCX}$, in Eq. (4) becomes super-polynomial. This again makes the ABE decryption inefficient.

Fortunately, for Boolean formulae, the work of [23] develops specialized homomorphic evaluation procedures $\mathsf{EvalF}, \mathsf{EvalFX}$ that ensure the evaluation noise $\mathbf{e}_f$

has a polynomial range. Therefore, our secret sharing scheme for CP removes the rounding and replaces $\mathsf{EvalC}, \mathsf{EvalCX}$ by $\mathsf{EvalF}, \mathsf{EvalFX}$. We summarize our modified secret sharing scheme below ($\mathsf{Setup}, \mathsf{ShareX}$ are kept the same).

**Modified Secret Sharing Scheme for CP-ABE**

$\mathsf{ShareF}'(\mathsf{pp}, f, \mu, \mathbf{s})$ : Compute $\mathbf{B}_f = \mathsf{EvalF}(\mathbf{B}, f)$.
Output $L_f = \mathbf{s}^\top \mathbf{B}_f \mathbf{G}^{-1}(\mathbf{a}) + \mu\lfloor p/2 \rceil + e$.

$\mathsf{Recon}'(\mathsf{pp}, f, L_f, \mathbf{x}, \mathbf{L}^{\mathbf{x}})$ : If $f(\mathbf{x}) = 1$, output $\perp$.
Otherwise, compute $\mathbf{c}_f = \mathsf{EvalFX}(\mathbf{L}^{\mathbf{x}}, f, \mathbf{x})$,
and find $\mu$ from $L_f - \mathbf{c}_f^\top \mathbf{G}^{-1}(\mathbf{a}) = \mu\lfloor p/2 \rceil + (e - e'_f)$.

As noted before, in our CP-ABE scheme we use IPFE to compute $\mathbf{L}^{\mathbf{x}}$. To achieve double succinctness, we carefully implement a pair of functions $\mathsf{Sel}, \mathsf{Encode}$ using an IPFE with constant-size $\mathsf{isk}$'s, such that $\mathsf{Sel}([\![\mathbf{x}]\!]_2)$ and $\mathsf{Encode}([\![\mathbf{L}_0, \{\mathbf{L}_i^b\}]\!]_1)$ decrypts exactly to $[\![\mathbf{L}^{\mathbf{x}}]\!]_{\mathrm{T}}$. We obtain a CP-ABE scheme for Boolean formulae as summarized below.

**Our CP-ABE Scheme**

$\mathsf{Setup}(1^\lambda)$ : Output $\mathsf{mpk} = \mathsf{impk}$ for IPFE and $\mathsf{pp}$ for secret sharing,
and $\mathsf{msk} = \mathsf{imsk}$ for IPFE.

$\mathsf{KeyGen}(\mathsf{msk}, \mathbf{x})$ : Sample $\delta \xleftarrow{\$} \mathbb{Z}_p$.
Output $\mathsf{sk} = ([\![\delta]\!]_2, \mathsf{Sel}([\![\delta\mathbf{x}]\!]_2))$.

$\mathsf{Enc}(\mathsf{mpk}, f, \mu)$ : Compute $(\mathbf{L}_0, \{\mathbf{L}_i^b\}, \mathbf{s}) \xleftarrow{\$} \mathsf{ShareX}(\mathsf{pp})$
and $L_f \xleftarrow{\$} \mathsf{ShareF}'(\mathsf{pp}, f, \mu, \mathbf{s})$.
Output $\mathsf{ct} = ([\![L_f]\!]_1, \mathsf{Encode}([\![\mathbf{L}_0, \{\mathbf{L}_i^b\}]\!]_1))$.

$\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}, \mathbf{x}, \mathsf{ct}, f)$ : Run IPFE decryption to recover $[\![\delta\mathbf{L}^{\mathbf{x}}]\!]_{\mathrm{T}}$.
Compute $[\![\delta\mathbf{c}_f]\!]_{\mathrm{T}} = \mathsf{EvalFX}([\![\delta\mathbf{L}^{\mathbf{x}}]\!]_{\mathrm{T}}, f, \mathbf{x})$,
and find $\mu$ from
$[\![L_f]\!]_1[\![\delta]\!]_2 - [\![\delta\mathbf{c}_f^\top]\!]_{\mathrm{T}}\mathbf{G}^{-1}(\mathbf{a}) = [\![\delta(\mu\lfloor p/2 \rceil + (e - e'_f))]\!]_{\mathrm{T}}$.

We now describe the $\mathsf{Sel}, \mathsf{Encode}$ functions. Let $\ell = |\mathbf{x}|$ denote the length of $\mathbf{x}$. The $\mathsf{Sel}$ algorithm first computes the "selection vector" for $\mathbf{x}$ as

$$\mathbf{v} = (1, 1 - \mathbf{x}[1], \mathbf{x}[1], \ldots, 1 - \mathbf{x}[i], \mathbf{x}[i], \ldots),$$

and then computes an IPFE secret key $\mathsf{isk}([\![\mathbf{v}]\!]_2)$. The $\mathsf{Encode}$ algorithm places input shares in the matrix

$$\begin{pmatrix} \mathbf{L}_0 & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{L}_1^0 & \mathbf{L}_1^1 & \cdots & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{L}_i^0 & \mathbf{L}_i^1 & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{L}_\ell^0 & \mathbf{L}_\ell^1 \end{pmatrix}$$

and computes one IPFE ciphertext for each row $\mathbf{u}_l$ of the matrix. Our CP-ABE has both succinct keys and ciphertexts: $|\mathsf{sk}| = \mathrm{O}(1)$ and $|\mathsf{ct}| = \mathrm{poly}(\lambda)|\mathbf{x}|^2$.

**Simulation Security for IPFE in GGM.** Similar to the security proof for KP-ABE, our security proof for CP-ABE requires selective simulation security of IPFE.

$$\begin{array}{rll|c|ll} \mathsf{Sel}: & \mathsf{isk}([\![\mathbf{v}]\!]_2) & & & \widetilde{\mathsf{isk}}([\![\mathbf{L}^{\mathbf{x}}]\!]_2) \\ \mathsf{Encode}: \forall l & \mathsf{ict}([\![\mathbf{u}_l]\!]_1) & & \overset{c}{\approx} & \widetilde{\mathsf{ict}}(\bot) \end{array}$$

Note that above we need to simulate *multiple* IPFE ciphertexts and program all their decryption outcome $\mathbf{L}^{\mathbf{x}}$ in each secret key. This is possible using existing IPFE schemes [2,32], but at the cost of having the secret key size proportional to the number $k = |\mathbf{L}^{\mathbf{x}}|$ of ciphertexts to be simulated. However, we aim for constant-size secret keys (independent of $k$). Unfortunately, in the standard model, it is impossible to achieve simulation security for $k$ ciphertexts if the secret key is shorter than $k$ bits by an incompressibility argument [12]. We show that simulation security for unbounded polynomially many ciphertexts can nevertheless be achieved with constant-size secret keys in the GGM. In particular, the IPFE scheme of [1], whose secret key contains a single group element, satisfies it. Roughly speaking, in the GGM, an adversary only learns information about values in the exponent through zero-test queries over the pairings of keys and ciphertexts, which the simulator can answer by translating them into zero-test queries over the inner products. As a side note, we can in fact prove *adaptive* simulation security for the [1] IPFE scheme, though our ABE scheme only relies on *selective* simulation security.

**Achieving Adaptive Security.** Examining the security sketch for KP-ABE, we observe that in our construction, the $\mathsf{ict}([\![\mathbf{s}, \mu]\!]_1)$ component of ciphertext $\mathsf{ct}^*$ doesn't depend on the challenge attribute $\mathbf{x}^*$. This means that even in the adaptive KP-ABE game, where $\mathbf{x}^*$ is decided after some key queries, the $\mathsf{ict}([\![\mathbf{s}, \mu]\!]_1)$ component of $\mathsf{ct}^*$ can be fixed at the beginning of the game, before any key queries. Therefore, we can still rely on selective simulation security of IPFE for the first proof step.

However, when we next need to invoke non-annihilability for $L_f$, we run into a problem: the security for $L_f$ only holds when $\mathbf{x}^*$ is chosen before the LWE public matrix $\mathbf{B}$ is revealed in the public parameter $\mathsf{pp}$ of the secret sharing. To achieve adaptive security, what we need is adaptive non-annihilability property, which allows $\mathbf{x}^*$ to be chosen adaptively dependent on $\mathsf{pp}$. We show that this is implied by the *adaptive* LWE assumption formulated in [36].

In summary, we obtain adaptively secure KP-ABE for circuits and CP-ABE for Boolean formulae both with constant-size keys from GGM and Adaptive LWE.

## 3 Preliminaries

Let $\lambda$ be the security parameter, which runs through $\mathbb{N}$. Except in the definitions, we suppress $\lambda$ for brevity. We write $[a..b]$ for the set $\{a, a+1, \ldots, b\}$ and $[n]$ for $[1..n]$. Vectors and matrices are written in boldface, and are always indexed using $[\cdot]$, i.e., $\mathbf{A}[i,j]$ is the $(i,j)$-entry of $\mathbf{A}$. The infinity norm of a vector and its induced operator norm of a matrix are denoted by $\|\cdot\|_\infty$. We will use the following lemma for various proofs:

**Lemma 1** (Schwartz–Zippel)**.** *Let $P(\mathbf{z})$ be a non-zero polynomial with $Z$ indeterminates of degree at most $d$ over $\mathbb{Z}_p$, then* $\Pr\left[\mathbf{z} \xleftarrow{\$} \mathbb{Z}_p^Z : P(\mathbf{z}) = 0\right] \leq d/p$.

### 3.1 Attribute-Based Encryption

**Definition 1** (ABE [24])**.** *Let $\mathcal{P} = \{\mathcal{P}_\lambda\}_{\lambda\in\mathbb{N}}$ be a sequence of predicate families with $\mathcal{P}_\lambda = \{P : X_P \times Y_P \to \{0,1\}\}$. An* attribute-based encryption *scheme for $\mathcal{P}$ consists of 4 efficient algorithms:*

- $\mathsf{Setup}(1^\lambda, P)$ *takes as input the security parameter $1^\lambda$ and a predicate $P \in \mathcal{P}_\lambda$, and outputs a pair of master public/secret keys* $(\mathsf{mpk}, \mathsf{msk})$.
- $\mathsf{KeyGen}(\mathsf{msk}, y)$ *takes as input the master secret key* $\mathsf{msk}$ *and some $y \in Y_P$, and outputs a secret key* $\mathsf{sk}$.
- $\mathsf{Enc}(\mathsf{mpk}, x, \mu)$ *takes as input the master public key* $\mathsf{mpk}$*, some $x \in X_P$, and a message $\mu \in \{0,1\}$, and it outputs a ciphertext* $\mathsf{ct}$.
- $\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}, y, \mathsf{ct}, x)$ *takes as input the master public key* $\mathsf{mpk}$*, a secret key* $\mathsf{sk}$*, its associated $y$, a ciphertext* $\mathsf{ct}$*, and its associated $x$, and is supposed to recover the message if $P(x,y) = 1$.*

*The scheme is required to be* correct*, i.e., for all $\lambda \in \mathbb{N}$, $P \in \mathcal{P}_\lambda$, $x \in X_P$, $y \in Y_P$, $\mu \in \{0,1\}$ such that $P(x,y) = 1$, it holds that*

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

In KP-ABE, each $y \in Y_P$ describes a function from $X_P$ to $\{0,1\}$, each $x \in X_P$ is an input (bit-string) to the functions, and $P(x,y)$ evaluates $y$ on $x$. When we want to emphasize $x$ (resp. $y$) is a bit-string (resp. circuit), we write $\mathbf{x}$ (resp. $C$) instead.

**Security.** Due to space constraints, we refer the readers to [4,29,31] for the definitions of *adaptive*, *selective*, and *very selective* security for ABE.

**Computation Model.** We will consider KP-ABE for bounded-depth circuits for any polynomial bound. Our CP-ABE for $\mathsf{NC}^1$ can be found in the full version [31].

**Definition 2** (KP-ABE for circuits)**.** *A* KP-ABE for (bounded-depth) circuits *is ABE for* $\mathcal{P}^{\mathsf{Ckt}}$*:*[2]

$$X^{\mathsf{Ckt}}_{\lambda,\ell,d} = \{0,1\}^{\ell}, \quad Y^{\mathsf{Ckt}}_{\lambda,\ell,d} = \left\{\text{Booleancircuit } C : \{0,1\}^{\ell} \to \{0,1\} \text{ of depth } d\right\},$$
$$P^{\mathsf{Ckt}}_{\lambda,\ell,d}(\mathbf{x}, C) = \neg C(\mathbf{x}), \quad \mathcal{P}^{\mathsf{Ckt}}_{\lambda} = \left\{P^{\mathsf{Ckt}}_{\lambda,\ell,d} \middle| \ell, d \in \mathbb{N}, d \le D_\lambda\right\}, \quad \mathcal{P}^{\mathsf{Ckt}} = \left\{\mathcal{P}^{\mathsf{Ckt}}_{\lambda}\right\}_{\lambda\in\mathbb{N}}.$$

*Here,* $D_\lambda$ *is a super-polynomial function (specified by the constructions). As an input to* $\mathsf{Setup}$*, the predicate* $P^{\mathsf{Ckt}}_{\lambda,\ell,d}$ *is represented by* $(1^\ell, 1^d)$*.*

Note that since $\mathsf{Setup}$ takes the unary representation of $\ell, d$, which will be polynomial in $\lambda$, as input, they are bounded by *that* polynomial once the system is set up. However, $d$ can be up to $D_\lambda$, which is super-polynomial in $\lambda$, so one can set up the system for any polynomial depth, i.e., our KP-ABE for circuits supports bounded-depth circuits for arbitrary polynomial depth bound.

**Compactness and Succinctness.** Since $\mathsf{KeyGen}, \mathsf{Enc}$ run in polynomial time, the lengths of key and ciphertext could grow polynomially in $|y|, |x|$, respectively. Moreover, the input length is an argument passed into $\mathsf{Setup}$, so both keys and ciphertexts could have polynomial size dependency on it. We are interested in ABE schemes with short keys and ciphertexts:

**Definition 3** (ABE efficiency)**.** *A KP-ABE for circuits (of depth at most d) has*

- succinct keys *if* $|\mathsf{sk}| = \mathrm{poly}(\lambda, d)$ *is independent of* $|C|, |\mathbf{x}|$*;*
- compact ciphertexts *if* $|\mathsf{ct}| = |\mathbf{x}|\,\mathrm{poly}(\lambda, d)$ *is independent of* $|C|$*.*

We remark that an ideally succinct component should be of length $\mathrm{poly}(\lambda)$. Nevertheless, our version defined above is still meaningful as the circuit size can be much larger than its depth.

### 3.2 Lattice Tools

**Homomorphic Evaluation.** We use the following abstraction of homomorphic evaluation for ABE over lattices, developed in a series of works [11,19,23] with the syntax in [13,14]. The actual algorithm we use is a slightly changed version of that for ABE for circuits in [11]. In our version, instead of using $\mathbf{G}$ as the

[2] When working with lattices, it is more convenient to indicate authorization of decryption by zero, thus the negation of $C(\mathbf{x})$.

gadget matrix, we consider $\mathbf{QG}$ for any invertible $\mathbf{Q}$. Note that $\mathbf{G}^{-1}(\mathbf{Q}^{-1} \times \cdot)$ is a right inverse of $\mathbf{QG}$ with binary output. We replace any invocation of $\mathbf{G}^{-1}(\cdot)$ in the original algorithms by $\mathbf{G}^{-1}(\mathbf{Q}^{-1} \times \cdot)$ to obtain the following:

**Lemma 2** (homomorphic evaluation for circuits, adapted from [11])**.** $\mathsf{EvalC}$ *and* $\mathsf{EvalCX}$ *are two efficient deterministic algorithms. Let* $n, \ell, q$ *be positive integers,* $m = n\lceil \log_2 q \rceil$*,* $\mathbf{G}$ *the gadget matrix,* $\mathbf{B}$ *a matrix over* $\mathbb{Z}_q$ *of shape* $n \times (\ell+1)m$*,* $\mathbf{Q}$ *an invertible matrix over* $\mathbb{Z}_q$ *of shape* $n \times n$*,* $\mathbf{x}$ *an* $\ell$*-bit string (row vector), and* $C$ *a circuit of depth* $d$ *with input length* $\ell$*. The algorithms work as follows:*

- $\mathsf{EvalC}(\mathbf{B}, \mathbf{Q}, C)$ *outputs* $\mathbf{H}_C \in \mathbb{Z}^{(\ell+1)m \times m}$*;*
- $\mathsf{EvalCX}(\mathbf{B}, \mathbf{Q}, C, \mathbf{x})$ *outputs* $\widehat{\mathbf{H}}_{C,\mathbf{x}} \in \mathbb{Z}^{(\ell+1)m \times m}$*.*

*The outputs satisfy*

$$\|\mathbf{H}_C^\intercal\|_\infty, \|\widehat{\mathbf{H}}_{C,\mathbf{x}}^\intercal\|_\infty \leq (m+1)^d, \quad \big(\mathbf{B} - (1, \mathbf{x}) \otimes \mathbf{QG}\big)\widehat{\mathbf{H}}_{C,\mathbf{x}} = \mathbf{B}\mathbf{H}_C - C(\mathbf{x})\mathbf{QG}.$$

**Gadget Matrix** [33]**.** Let $n, q$ be positive integers and $m = n\lceil \log_2 q \rceil$. The gadget matrix is $\mathbf{G} = \mathbf{g}^\intercal \otimes \mathbf{I}_n$, where $\mathbf{g}^\intercal = (2^0, 2^1, \ldots, 2^{\lceil \log_2 q \rceil - 1})$. There exists an efficiently computable function $\mathbf{G}^{-1} : \mathbb{Z}_q^n \to \{0,1\}^m$ such that $\mathbf{G} \cdot \mathbf{G}^{-1}(\mathbf{u}) = \mathbf{u}$ for all $\mathbf{u} \in \mathbb{Z}_q^n$.

**Assumption.** We rely on the following assumption, a small-secret version of adaptive learning with errors (LWE), which itself is a natural variant of LWE first proposed in [36]:

**Definition 4** (small-secret adaptive LWE)**.** *We suppress the security parameter* $\lambda$ *and all the parameters are dependent on* $\lambda$*. Let* $n$ *be the dimension,* $q$ *the modulus,* $\chi$ *the error distribution,* $m = n\lceil \log_2 q \rceil$*, and* $\mathbf{G}$ *the gadget matrix. The small-secret adaptive LWE assumption* $\mathrm{sALWE}_{n,q,\chi}$ *statesthat* $\mathsf{Exp}^0_{\mathrm{sALWE}} \approx \mathsf{Exp}^1_{\mathrm{sALWE}}$*, where* $\mathsf{Exp}^b_{\mathrm{sALWE}}(1^n, q, \chi)$ *with adversary* $\mathcal{A}$ *proceeds as follows:*

- **Setup.** The challenger launches $\mathcal{A}$ and receives $(1^\ell, 1^{m'})$ from it. The challenger samples $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m'}$, $\mathbf{B} \xleftarrow{\$} \mathbb{Z}_q^{n \times (\ell+1)m}$, and a uniformly random invertible $\mathbf{Q} \in \mathbb{Z}_q^{n \times n}$. It sends $\mathbf{A}, \mathbf{B}, \mathbf{Q}$ to $\mathcal{A}$.
- **Challenge.** $\mathcal{A}$ submits $\mathbf{x} \in \{0,1\}^\ell$. Depending on $b$,

$$\begin{aligned} &\text{if } b = 0: && \boxed{\mathbf{s} \xleftarrow{\$} \chi^n},\ \mathbf{e} \xleftarrow{\$} \chi^{m'},\ \mathbf{f} \xleftarrow{\$} \chi^{(\ell+1)m}, \\ & && \mathbf{c}^\intercal = \mathbf{s}^\intercal\mathbf{A} + \mathbf{e}^\intercal, \qquad \mathbf{d}^\intercal = \mathbf{s}^\intercal\big(\mathbf{B} - (1, \mathbf{x}) \otimes \mathbf{QG}\big) + \mathbf{f}^\intercal; \\ &\text{if } b = 1: && \mathbf{c}^\intercal \xleftarrow{\$} \mathbb{Z}_q^{m'}, \qquad \mathbf{d}^\intercal \xleftarrow{\$} \mathbb{Z}_q^{(\ell+1)m}. \end{aligned}$$

  The challenger sends $\mathbf{c}, \mathbf{d}$ to $\mathcal{A}$.
- **Guess** $\mathcal{A}$ outputs a bit, which is the outcome of the experiment.

**Lemma 3** (small-secret adaptive LWE)**.** *The small-secret adaptive LWE assumption holds if the adaptive LWE assumption [36] holds for the same parameters.*

The proof of Lemma 3 can be found in the full version [31].

**Parameter Settings.** We rely on the hardness of small-secret LWE with subexponential modulus-to-noise ratio. For some $0 < \delta < \frac{1}{2}$, the small-secret LWE assumption is assumed to be hard when the dimension is $n = \mathrm{poly}(\lambda)$, the *prime* modulus is $q = \mathrm{O}(2^{n^\delta})$, and the error distribution $\chi$ is the discrete Gaussian over $\mathbb{Z}$ of width $\overline{B}/\lambda$ truncated within $[-\overline{B}..\overline{B}]$ for $\overline{B} = \mathrm{poly}(\lambda)$.[3] Hereafter we default to these parameters.

### 3.3 Pairing Groups and Generic Asymmetric Pairing Group Model

We construct our ABE using pairing groups and prove its security in the generic pairing group model.

**Pairing Groups.** Throughout the paper, we use a sequence of pairing groups

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

where $G_{\lambda,1}$ (resp. $G_{\lambda,2}, G_{\lambda,\mathrm{T}}$) is a cyclic group generated by $g_{\lambda,1}$ (resp. $g_{\lambda,2}$, $g_{\lambda,\mathrm{T}}$) of prime order $p_\lambda = 2^{\lambda^{\Theta(1)}}$ and $e_\lambda : G_{\lambda,1} \times G_{\lambda,2} \to G_{\lambda,\mathrm{T}}$ is the pairing operation, satisfying $e_\lambda(g_{\lambda,1}^a, g_{\lambda,2}^b) = g_{\lambda,\mathrm{T}}^{ab}$ for all integers $a, b$. We require the group operations as well as the pairing operation to be efficiently computable.

For a fixed security parameter $\lambda$, we denote $g_{\lambda,i}^x$ by $[\![x]\!]_i$ for $i \in \{1, 2, \mathrm{T}\}$. The notation extends to matrices, $[\![\mathbf{A}]\!]_i = g_{\lambda,i}^{\mathbf{A}}$, where exponentiation is done component-wise. With these notations, the group operations are written additively and the pairing operation multiplicatively. For example, $[\![\mathbf{A}]\!]_1 - \mathbf{B}[\![\mathbf{C}]\!]_1\mathbf{D} = [\![\mathbf{A} - \mathbf{BCD}]\!]_1$ and $[\![\mathbf{X}]\!]_2[\![\mathbf{Y}]\!]_1 = [\![\mathbf{XY}]\!]_\mathrm{T}$.

**Generic Asymmetric Pairing Group.** The security of our ABE scheme holds in the generic asymmetric pairing group model (GGM), where the pairing groups can only be accessed via (non-unique) handles representing group elements and oracles for operating the handles. Due to space constraints, we refer the readers to the full version [31] for the formal definition of the version we use in this work.

### 3.4 Inner-Product Functional Encryption

Inner-product functional encryption schemes enable generating keys and ciphertexts tied to vectors. Decryption reveals the inner product and nothing more about the plaintext vector. In this work, we consider IPFE schemes based on pairing, where keys and ciphertexts are encoded in the two source groups and decryption recovers inner products encoded in the target group.

**Definition 5** (group-based IPFE)**.** *Let $\mathcal{G}$ be a sequence of pairing groups of order $\{p_\lambda\}_{\lambda\in\mathbb{N}}$. An* inner-product functional encryption (IPFE) *scheme based on $\mathcal{G}$ consists of 4 efficient algorithms:*

[3] This truncation only introduces an exponentially small statistical error.

- $\mathsf{Setup}(1^\lambda, 1^N)$ *takes as input the security parameter* $1^\lambda$ *and the vector dimension* $1^N$*. It outputs a pair of master public/secret keys* $(\mathsf{impk}, \mathsf{imsk})$.
- $\mathsf{KeyGen}(\mathsf{imsk}, [\![\mathbf{v}]\!]_2)$ *takes as input the master secret key and a vector (encoded in* $G_2$*), and outputs a secret key* $\mathsf{isk}$.
- $\mathsf{Enc}(\mathsf{impk}, [\![\mathbf{u}]\!]_1)$ *takes as input the master public key and a vector (encoded in* $G_1$*), and outputs a ciphertext* $\mathsf{ict}$.
- $\mathsf{Dec}(\mathsf{isk}, [\![\mathbf{v}]\!]_2, \mathsf{ict})$ *takes a secret key, the vector in the secret key, and a ciphertext as input, and is supposed to compute the inner product encoded in* $G_\mathrm{T}$.

*The scheme is required to be* correct*, meaning that for all* $\lambda, N \in \mathbb{N}, \mathbf{u}, \mathbf{v} \in \mathbb{Z}_{p_\lambda}^N$,

$$\Pr\left[\begin{array}{r}(\mathsf{impk}, \mathsf{imsk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, 1^N) \\ \mathsf{isk} \xleftarrow{\$} \mathsf{KeyGen}(\mathsf{imsk}, [\![\mathbf{v}]\!]_2) : \mathsf{Dec}(\mathsf{isk}, [\![\mathbf{v}]\!]_2, \mathsf{ict}) = [\![\mathbf{u}^\mathsf{T}\mathbf{v}]\!]_\mathrm{T} \\ \mathsf{ict} \xleftarrow{\$} \mathsf{Enc}(\mathsf{impk}, [\![\mathbf{u}]\!]_1)\end{array}\right] = 1.$$

**Definition 6** (key-succinct IPFE)**.** *An IPFE scheme (Definition 5) is* (key-)succinct *if the length of* $\mathsf{isk}$ *is a fixed polynomial in* $\lambda$*, independent of* $N$.

**Security.** Our basic security notion is selective simulation:

**Definition 7** (selective simulation [32,41])**.** *A* simulator *for an IPFE scheme (Definition 5) consists of 3 efficient algorithms:*

- $\widetilde{\mathsf{Setup}}(1^\lambda, 1^N)$ *takes the same input as* $\mathsf{Setup}$*, and outputs simulated keys* $(\widetilde{\mathsf{impk}}, \widetilde{\mathsf{imsk}})$.
- $\widetilde{\mathsf{KeyGen}}(\widetilde{\mathsf{imsk}}, [\![\mathbf{v}]\!]_2, [\![z_i]\!]_2)$ *takes as input the simulated master secret key, a vector encoded in* $G_2$*, and an inner product encoded in* $G_2$*. It outputs a simulated key* $\widetilde{\mathsf{isk}}$.
- $\widetilde{\mathsf{Enc}}(\widetilde{\mathsf{imsk}})$ *takes as input the simulated master secret key. It outputs a simulated ciphertext* $\widetilde{\mathsf{ict}}$.

*The IPFE scheme is* selectively simulation-secure *if there exists a simulator such that* $\mathsf{Exp}_\mathrm{real} \approx \mathsf{Exp}_\mathrm{sim}$*, where* $\mathsf{Exp}_\mathrm{real}(1^\lambda)$ *or* $\mathsf{Exp}_\mathrm{sim}(1^\lambda)$ *with* $\mathcal{A}$ *proceeds as follows:*

- **Challenge.** The challenger launches $\mathcal{A}(1^\lambda)$ and receives from it the vector dimension $1^N$ and the challenge vector $\mathbf{u} \in \mathbb{Z}_p^N$.
- **Setup.** The challenger runs

$$\begin{array}{lll}\text{in } \mathsf{Exp}_\mathrm{real}\text{:} & (\mathsf{impk}, \mathsf{imsk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, 1^N), & \mathsf{ict} \xleftarrow{\$} \mathsf{Enc}(\mathsf{impk}, [\![\mathbf{u}]\!]_1); \\ \text{in } \mathsf{Exp}_\mathrm{sim}\text{:} & (\mathsf{impk}, \widetilde{\mathsf{imsk}}) \xleftarrow{\$} \widetilde{\mathsf{Setup}}(1^\lambda, 1^N), & \mathsf{ict} \xleftarrow{\$} \widetilde{\mathsf{Enc}}(\widetilde{\mathsf{imsk}});\end{array}$$

  and sends $\mathsf{impk}, \mathsf{ict}$ to $\mathcal{A}$.

– **Query.** The following is repeated for arbitrarily many rounds determined by $\mathcal{A}$: In each round, $\mathcal{A}$ submits a vector $[\![\mathbf{v}_j]\!]_2$ encoded in $G_2$. Upon receiving the query, the challenger runs

$$\text{in } \mathsf{Exp}_{\text{real}}: \quad \mathsf{isk}_j \xleftarrow{\$} \mathsf{KeyGen}(\mathsf{imsk}, [\![\mathbf{v}_j]\!]_2);$$
$$\text{in } \mathsf{Exp}_{\text{sim}}: \quad \mathsf{isk}_j \xleftarrow{\$} \widetilde{\mathsf{KeyGen}}(\widetilde{\mathsf{imsk}}, [\![\mathbf{v}_j]\!]_2, \mathbf{u}^\intercal [\![\mathbf{v}_j]\!]_2);$$

and sends $\mathsf{isk}_j$ to $\mathcal{A}$.
– **Guess** $\mathcal{A}$ outputs a bit $b$, which is the output of the experiment.

**Lemma 4** ([2,41]). *Assuming the MDDH assumption (true in GGM), there exists a succinct selectively simulation-secure IPFE scheme. Its components have sizes*

$$|\mathsf{impk}| = k(k+1+N)|G_1|, \qquad |\mathsf{imsk}| = (k+1)N\log_2 p,$$
$$|\mathsf{isk}| = (k+1)|G_2|, \qquad |\mathsf{ict}| = (k+1+N)|G_1|,$$

*where $p$ is the modulus, $k$ is the MDDH parameter (can be 1 in GGM), $N$ is the dimension, and $|G_i|$ is the bit-length of an element in $G_i$.*

## 4 Computational Secret Sharing with Adaptive Security

Secret sharing schemes have been used extensively to construct ABE schemes. The seminal work of [24] and a long line of follow-up works ( [5,28–30,35] to name a few) used *linear* secret sharing schemes to construct ABE schemes in pairing groups. Policies with polynomial-sized shares and information-theoretic security are in NC [8,9,15,26,34].

The works of [3,4] introduced the notion of *nearly linear* secret sharing with *computational* security. The relaxations enabled greater expressiveness and better efficiency. Assuming LWE, such a scheme exists for all polynomial-sized circuits [3,4,11,23] and the shares are *succinct*, i.e., they only grow with the circuit depth, but not the circuit size. However, the scheme is only *selectively* secure. Furthermore, due to technical reasons, when combined with pairing to obtain ABE, it only applies to Boolean formulae (equivalent to 5-PBP).

This work follows the blueprint of [3,4] for the notions of secret sharing schemes, but departs from them in three important aspects. First, we consider a different security notion, *adaptive non-annihilability*, which is incomparable[4] to selective pseudorandomness considered in [3,4] and enables us to prove adaptive security of ABE. Second, we further relax the linearity requirement so that it could apply to KP-ABE for polynomial-sized circuits. Third, we refine the syntax to separate encodings of input and function.

[4] It is stronger in that it is adaptive, but weaker in that the shares are not necessarily pseudorandom.

**Definition 8** (secret sharing). *Let* $\mathcal{F} = \{\mathcal{F}_{\lambda,\ell,\mathsf{param}}\}_{\lambda,\ell\in\mathbb{N},\mathsf{param}}$ *be an ensemble of Boolean function families such that for all* $\lambda, \ell \in \mathbb{N}$ *and* $\mathsf{param}$*, every* $f \in \mathcal{F}_{\lambda,\ell,\mathsf{param}}$ *is a function mapping* $\{0,1\}^\ell$ *to* $\{0,1\}$*. A* secret sharing scheme *for* $\mathcal{F}$ *consists of 4 efficient algorithms:*

- $\mathsf{Setup}(1^\lambda, 1^\ell, \mathsf{param})$ *takes the security parameter* $1^\lambda$*, the input length* $1^\ell$*, and additional parameters* $\mathsf{param}$ *as input. It outputs some public parameter* $\mathsf{pp}$*.*
- $\mathsf{ShareX}(\mathsf{pp})$ *takes the public parameter* $\mathsf{pp}$ *as input. It outputs* $1 + 2\ell$ *shares,* $L_0, \{L_i^b\}_{i\in[\ell]}^{b\in\{0,1\}}$*, and some shared randomness* $r$*. For* $x \in \{0,1\}^\ell$*, we denote by* $L^{\mathbf{x}}$ *the set of shares* $L_0, \{L_i^{\mathbf{x}[i]}\}_{i\in[\ell]}$*.*
- $\mathsf{ShareF}(\mathsf{pp}, f, \mu, r)$ *takes the public parameter* $\mathsf{pp}$*, a function* $f \in \mathcal{F}_{\lambda,\ell,\mathsf{param}}$*, a secret* $\mu \in \{0,1\}$*, and the shared randomness* $r$ *(output by* $\mathsf{ShareX}$*) as input. It outputs a share* $L_f$*.*
- $\mathsf{Recon}(\mathsf{pp}, f, \mathbf{x}, L_f, L^{\mathbf{x}})$ *takes the public parameter* $\mathsf{pp}$*, the Boolean function* $f \in \mathcal{F}_{\lambda,\ell,\mathsf{param}}$*, the input* $\mathbf{x} \in \{0,1\}^\ell$ *to* $f$*, and the shares* $L_f, L^{\mathbf{x}}$ *as input. It is supposed to recover the secret* $\mu$ *if* $f(\mathbf{x}) = 0$*.* [5]

*The scheme is required to be* correct*, i.e., for all* $\lambda, \ell \in \mathbb{N}$*,* $\mathsf{param}$*,* $\mathbf{x} \in \{0,1\}^\ell$*,* $f \in \mathcal{F}_{\lambda,\ell,\mathsf{param}}$*,* $\mu \in \{0,1\}$ *such that* $f(\mathbf{x}) = 0$*, it holds that*

$$\Pr\left[\begin{array}{c} \mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda, 1^\ell, \mathsf{param}) \\ (L_0, \{L_i^b\}_{i\in[\ell]}^{b\in\{0,1\}}, r) \xleftarrow{\$} \mathsf{ShareX}(\mathsf{pp}) \\ L_f \xleftarrow{\$} \mathsf{ShareF}(\mathsf{pp}, f, \mu, r) \end{array} : \mathsf{Recon}(\mathsf{pp}, f, \mathbf{x}, L_f, L^{\mathbf{x}}) = \mu \right] = 1.$$

**Definition 9** (succinct shares). *A secret sharing scheme is* succinct *if the size of each share output by* $\mathsf{ShareX}, \mathsf{ShareF}$ *is a fixed polynomial in* $\lambda$*, independent of the length of* $\mathbf{x}$ *or the description size of* $f$*, i.e.,* $|L_f|, |L_0|, |L_i^b|$ *are all* $\mathrm{poly}(\lambda, |\mathsf{param}|)$*, where* $i \in [\ell]$*,* $b \in \{0,1\}$*.* [6]

While correctness (Definition 8) and succinctness (Definition 9) are defined similarly to that of [3], our linearity and security notions are different.

### 4.1 Secret Sharing for Bounded-Depth Circuits from Adaptive LWE

In our KP-ABE construction, we need a secret sharing scheme with two linearity properties. The first is a relaxation of the nearly linear reconstruction requirement in [3]. requirement on reconstruction. Our relaxed version (Definition 10) only stipulates it to be linear in $\mathbf{L}_f$ (and possibly non-linear in $L^{\mathbf{x}}$).

**Definition 10** (weakly nearly linear reconstruction). *A secret sharing scheme (Definition 8) is* weakly nearly linear *if it satisfies the following requirements:*

- *Let* $\{p_\lambda\}_{\lambda\in\mathbb{N}}$ *be a sequence of prime numbers.* $\mathbf{L}_f = L_f$ *is a vector over* $\mathbb{Z}_{p_\lambda}$*.*

[5] We use $f(\mathbf{x}) = 0$ to express authorization.

[6] There are $2|\mathbf{x}| + 2$ shares, so the total share size is linear in the length of $\mathbf{x}$.

– *There is an efficient coefficient-finding algorithm* $\mathsf{FindCoef}(\mathsf{pp}, f, \mathbf{x}, L^{\mathbf{x}})$*, taking as input the public parameter* $\mathsf{pp}$*, a Boolean function* $f \in \mathcal{F}_{\lambda,\ell,\mathsf{param}}$*, an input* $\mathbf{x} \in \{0,1\}^{\ell}$ *to* $f$*, and the shares* $L^{\mathbf{x}}$*. It outputs an affine function* $\gamma$ *and a noise bound* $1^B$*. For all* $\lambda, \ell \in \mathbb{N}, \mathsf{param}, \mathbf{x} \in \{0,1\}^{\ell}, f \in \mathcal{F}_{\lambda,\ell,\mathsf{param}}, \mu \in \{0,1\}$ *such that* $f(\mathbf{x}) = 0$*, it holds that*

$$\Pr\left[\begin{array}{rl} \mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^{\lambda}, 1^{\ell}, \mathsf{param}) & \\ (L_0, \{L_i^b\}_{i\in[\ell]}^{b\in\{0,1\}}, r) \xleftarrow{\$} \mathsf{ShareX}(\mathsf{pp}) & 4B+1 < p_{\lambda} \text{ and} \\ \mathbf{L}_f \leftarrow \mathsf{ShareF}(\mathsf{pp}, f, \mu, r) & : \exists e \in [-B..B] \text{s.t.} \\ (\gamma, 1^B) \xleftarrow{\$} \mathsf{FindCoef}(\mathsf{pp}, f, \mathbf{x}, L^{\mathbf{x}}) & \gamma(\mathbf{L}_f) = \mu\lfloor p/2 \rceil + e \end{array}\right] = 1.$$

The second is an additional linearity requirement on $\mathsf{ShareF}$.

**Definition 11** (linear function sharing)**.** *Let* $\{p_{\lambda}\}_{\lambda\in\mathbb{N}}$ *be a sequence of primes. A secret sharing scheme (Definition 8) has* linear function sharing *if* $\mathbf{r} = r$ *is a vector over* $\mathbb{Z}_{p_{\lambda}}$ *and* $\mathsf{ShareF}(\mathsf{pp}, f, \mu, \mathbf{r})$ *is deterministic and linear in* $(\mu, \mathbf{r})$*.*

A (weakly) nearly linear scheme is by definition correct. Given $\mathsf{FindCoef}$, we let $\mathsf{Recon}$ call $\mathsf{FindCoef}$ to obtain $\gamma, B$ and output the unique $\mu \in \{0,1\}$ satisfying $\gamma(\mathbf{L}_f) - \mu\lfloor p/2 \rceil \in [-B..B]$. The constructed $\mathsf{Recon}$ is efficient and correct. Since $\mathsf{Recon}$ is implied by $\mathsf{FindCoef}$, we will only specify $\mathsf{FindCoef}$ and omit $\mathsf{Recon}$ when constructing (weakly) nearly linear secret sharing schemes.

**Security.** We consider a new security notion called *non-annihilability*. Unlike [3], which fixes the choice of policy $f$ before $\mathsf{Setup}$ is run, we allow the adversary to adaptively choose $f$ after seeing the public parameters $\mathsf{pp}$ and the input shares $L^{\mathbf{x}}$. Another difference is that instead of requiring all shares $(L_f, L^{\mathbf{x}})$ to look random, we only require that efficient adversaries cannot find a non-trivial affine function (potentially dependent on $L^{\mathbf{x}}$) that evaluates to zero on $\mathbf{L}_f$. This notion suffices for the security proofs of our KP-ABE scheme.

**Definition 12** (non-annihilability for $\mathbf{L}_f$)**.** *Let* $\{p_{\lambda}\}_{\lambda\in\mathbb{N}}$ *be a sequence of prime numbers. A secret sharing scheme (Definition 8) is* adaptively non-annihilable *for* $\mathbf{L}_f$ *if the output* $\mathbf{L}_f$ *of* $\mathsf{ShareF}$ *is a vector over* $\mathbb{Z}_{p_{\lambda}}$ *and all efficient adversary wins* $\mathsf{Exp}_{\mathrm{ANN-f}}$ *with negligible probability, where in* $\mathsf{Exp}^{\mathcal{A}}_{\mathrm{ANN-f}}(1^{\lambda})$*, the adversary* $\mathcal{A}$ *interacts with the challenger as follows:*

– **Setup.** *The challenger launches* $\mathcal{A}(1^{\lambda})$ *and receives from it the input length* $1^{\ell}$ *and the additional parameter* $\mathsf{param}$*. The challenger sets up the system by running* $\mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^{\lambda}, 1^{\ell}, \mathsf{param})$*, and sends* $\mathsf{pp}$ *to* $\mathcal{A}$*.*
– **Share.** $\mathcal{A}$ *first submits an input* $\mathbf{x} \in \{0,1\}^{\ell}$*. Upon receiving it, the challenger creates the input shares by running* $(L_0, \{L_i^b\}_{i\in[\ell]}^{b\in\{0,1\}}, r) \xleftarrow{\$} \mathsf{ShareX}(\mathsf{pp})$ *and sends* $L^{\mathbf{x}}$ *to* $\mathcal{A}$*.*
– **Challenge.** $\mathcal{A}$ *outputs a Boolean function* $f \in \mathcal{F}_{\lambda,\ell}$*, a message bit* $\mu \in \{0,1\}$*, and an affine function* $\gamma$*. Upon receiving them, the challenger runs*

$\mathbf{L}_f \xleftarrow{\$} \mathsf{ShareF}(\mathsf{pp}, f, \mu, r)$ *and determines the outcome of the experiment.* $\mathcal{A}$ *wins if* i) $f(\mathbf{x}) = 1$*;* ii) $\gamma$ *is not the zero function; and* iii) $\gamma(\mathbf{L}_f) = 0$*. Otherwise,* $\mathcal{A}$ *loses.*

*Furthermore, a secret sharing scheme is* selectively non-annihilable *if it satisfies the above conditions, with the change that the adversary must choose the input* $\mathbf{x}$ *before receiving* $\mathsf{pp}$*.*

We now construct a succinct secret sharing scheme, satisfying the above linearity and adaptive annihilability for bounded-depth circuits from small-secret adaptive LWE. Our construction is based on the attribute-based laconic function evaluation scheme [11,36].

**Construction 1** (secret sharing for circuits)**.** Let $n$ be the LWE dimension, $p = 2^{\omega(\log \lambda)}$ a fixed prime modulus, (the LWE parameters will be chosen during $\mathsf{Setup}$). We construct a weakly nearly linear and succinct secret sharing scheme, with linear function sharing, for the family of bounded-depth circuits (see Definition 2):

$$\mathsf{Ckt}_{\lambda,\ell,d} = \left\{ \text{Boolean circuit } C : \{0,1\}^{\ell} \to \{0,1\} \text{ of depth at most } d \right\},$$

where $d \leq \frac{p^{\delta/4} - \log_2 p}{(1+\delta^{-1})\Theta(1)}$. Let $(\mathsf{EvalC}, \mathsf{EvalCX})$ be the algorithms in Lemma 2.

- $\mathsf{Setup}(1^{\lambda}, 1^{\ell}, 1^{d})$ takes the input length $\ell$ in unary as input. It sets

$$n = \overline{B} = \big((d+1)(\delta^{-1}+1) + \log_2 p + \mathrm{O}(d)\big)^{2/\delta}, \quad q = 2^{n^{\delta}}, \quad m = n\lceil \log_2 q \rceil,$$

  and picks $\chi$ to be $\overline{B}$-bounded. It next samples and sets

$$\mathbf{a} \xleftarrow{\$} \mathbb{Z}_q^n, \quad \mathbf{A}_0, \mathbf{A}_1, \ldots, \mathbf{A}_\ell \xleftarrow{\$} \mathbb{Z}_q^{n \times m}, \quad \mathbf{B} = (\mathbf{A}_0, \mathbf{A}_1, \ldots, \mathbf{A}_\ell).$$

  It finally samples a random invertible matrix $\mathbf{Q} \in \mathbb{Z}_q^{n \times n}$, and outputs $\mathsf{pp} = (n, q, m, \overline{B}, \chi, \mathbf{a}, \mathbf{B}, \mathbf{Q})$.
  Note: *Recall that* $\delta$ *is a constant depending on the underlying adaptive LWE assumption. The choice of* $n, \overline{B}, q$ *are subject to the requirement of the underlying adaptive LWE assumption as well as correctness and efficiency of the scheme. They satisfy* $q/\overline{B} \geq (m+1)^{d+1}$ *and* $4((n+1)\overline{B}+3)+1 < p$*.*
- $\mathsf{ShareX}(\mathsf{pp})$ takes the public parameter $\mathsf{pp}$ as input. It samples and sets

$$\mathbf{s} \xleftarrow{\$} \chi^n, \quad \mathbf{e}_0, \mathbf{e}_1, \ldots, \mathbf{e}_\ell \xleftarrow{\$} \chi^m,$$
$$\mathbf{L}_0 = \mathbf{s}^{\intercal}(\mathbf{A}_0 - \mathbf{Q}\mathbf{G}) + \mathbf{e}_0^{\intercal}, \quad \{\mathbf{L}_i^b = \mathbf{s}^{\intercal}(\mathbf{A}_i - b\mathbf{Q}\mathbf{G}) + \mathbf{e}_i^{\intercal}\}_{i \in [\ell]}^{b \in \{0,1\}},$$

  and outputs $(\mathbf{L}_0, \{\mathbf{L}_i^b\}_{i \in [\ell]}^{b \in \{0,1\}}, \mathbf{s})$.
- $\mathsf{ShareF}(\mathsf{pp}, f, \mu, \mathbf{s})$ takes as input the public parameter $\mathsf{pp}$, some $f \in \mathsf{Ckt}_{\lambda,\ell,d}$, a secret bit $\mu \in \{0,1\}$, and the shared randomness $\mathbf{s}$. It runs $\mathbf{H}_f \leftarrow \mathsf{EvalC}(\mathbf{B}, \mathbf{Q}, f)$, and sets and outputs

$$\mathbf{L}_f = \mathbf{s}^{\intercal}\lfloor \mathbf{B}\mathbf{H}_f\mathbf{G}^{-1}(\mathbf{a}) \rceil_p + \mu\lfloor p/2 \rceil, \quad \text{where } \lfloor x \rceil_p = \lfloor px/q \rceil.$$

Note: *The scheme indeed has linear function sharing (Definition 11) because* $\mathsf{ShareF}$ *is a deterministic linear function over* $\mu, \mathbf{s}$ *with coefficients* $\lfloor p/2 \rceil$, $\lfloor \mathbf{B}\mathbf{H}_f\mathbf{G}^{-1}(\mathbf{a}) \rceil_p$. *The scheme is also succinct as* $\mathbf{L}_f$ *contains* 1 *element in* $\mathbb{Z}_p$, *and each share output by* $\mathsf{ShareX}$ *contains* $m$ *elements in* $\mathbb{Z}_q$. *Note that* $m$ *is a fixed polynomial in* $\lambda, d$ *and is independent of the description size of* $f$ *and the input length* $\ell$.

– $\mathsf{FindCoef}(\mathsf{pp}, f, \mathbf{x}, \mathbf{L}^{\mathbf{x}})$ takes as input the public parameter $\mathsf{pp}$, some $\mathbf{x} \in \{0,1\}^{\ell}$, some $f \in \mathsf{Ckt}_{\lambda,\ell,d}$, and the shares $\mathbf{L}^{\mathbf{x}}$. If $f(\mathbf{x}) = 1$, it outputs $\perp$ and terminates. Otherwise, it runs $\widehat{\mathbf{H}}_{f,\mathbf{x}} \leftarrow \mathsf{EvalCX}(\mathbf{B}, \mathbf{Q}, f, \mathbf{x})$, and defines

$$\gamma(\mathbf{L}_f) = \mathbf{L}_f - \lfloor \mathbf{L}^{\mathbf{x}} \widehat{\mathbf{H}}_{f,\mathbf{x}} \mathbf{G}^{-1}(\mathbf{a}) \rceil_p, \quad B = (n+1)\overline{B} + 3,$$

The algorithm outputs $(\gamma, 1^B)$.

Note: *The procedure is indeed efficient since* $n, \overline{B}$ *are polynomials in* $\lambda, d$. *We show that* $\mathsf{FindCoef}$ *is correct, i.e., if* $f(\mathbf{x}) = 0$, then $4B + 1 \leq p$ *and* $\gamma(\mathbf{L}_f) = \mu\lfloor p/2 \rceil + e$ *for some* $e \in [-B, B]$. *First, by the choice of* $n, \overline{B}$,

$$4B + 1 = 4((n+1)\overline{B} + 3) + 1 \leq p.$$

*Next, by construction we have*

$$\begin{aligned}
\gamma(\mathbf{L}_f) &= \mathbf{L}_f - \lfloor \mathbf{L}^{\mathbf{x}} \widehat{\mathbf{H}}_{f,\mathbf{x}} \mathbf{G}^{-1}(\mathbf{a}) \rceil_p \\
&= \underbrace{\mathbf{s}^{\intercal} \lfloor \mathbf{B}\mathbf{H}_f\mathbf{G}^{-1}(\mathbf{a}) \rceil_p + \mu\lfloor p/2 \rceil}_{\mathbf{L}_f} \\
&\quad - \lfloor \underbrace{\big(\mathbf{s}^{\intercal}(\mathbf{B} - (1, \mathbf{x}) \otimes \mathbf{Q}\mathbf{G}) + (\mathbf{e}_0^{\intercal}, \mathbf{e}_1^{\intercal}, \ldots, \mathbf{e}_{\ell}^{\intercal})\big)}_{\mathbf{L}^{\mathbf{x}}} \widehat{\mathbf{H}}_{f,\mathbf{x}} \mathbf{G}^{-1}(\mathbf{a}) \rceil_p \\
(\text{Lemma 2}) &= \mathbf{s}^{\intercal} \lfloor \mathbf{B}\mathbf{H}_f\mathbf{G}^{-1}(\mathbf{a}) \rceil_p + \mu\lfloor p/2 \rceil \\
&\quad - \lfloor \mathbf{s}^{\intercal}(\mathbf{B}\mathbf{H}_f - \underbrace{f(\mathbf{x})}_{=0} \mathbf{Q}\mathbf{G})\mathbf{G}^{-1}(\mathbf{a}) + \underbrace{(\mathbf{e}_0^{\intercal}, \mathbf{e}_1^{\intercal}, \ldots, \mathbf{e}_{\ell}^{\intercal}) \widehat{\mathbf{H}}_{f,\mathbf{x}} \mathbf{G}^{-1}(\mathbf{a})}_{=e_f} \rceil_p \\
&= \mathbf{s}^{\intercal} \lfloor \mathbf{B}\mathbf{H}_f\mathbf{G}^{-1}(\mathbf{a}) \rceil_p + \mu\lfloor p/2 \rceil - \lfloor \mathbf{s}^{\intercal}\mathbf{B}\mathbf{H}_f\mathbf{G}^{-1}(\mathbf{a}) + e_f \rceil_p
\end{aligned}$$

*Since* $\mathbf{G}^{-1}(\mathbf{a}) \in \{0,1\}^m$, *by the definition of* $\mathsf{EvalCX}$ *(Lemma 2), we have*

$$|e_f| \leq m \cdot \|\widehat{\mathbf{H}}_{f,\mathbf{x}}^{\intercal}\|_{\infty} \cdot \|(\mathbf{e}_0^{\intercal}, \mathbf{e}_1^{\intercal}, \ldots, \mathbf{e}_{\ell}^{\intercal})^{\intercal}\|_{\infty} \leq (m+1)^{(d+1)}\overline{B}$$

*Note that we can break a rounded sum into a sum of individually rounded terms, at the expense of some rounding errors:*

$$\begin{aligned}
&\lfloor \mathbf{s}^{\intercal}\mathbf{B}\mathbf{H}_f\mathbf{G}^{-1}(\mathbf{a}) + e_f \rceil = \lfloor \mathbf{s}^{\intercal}\mathbf{B}\mathbf{H}_f\mathbf{G}^{-1}(\mathbf{a}) \rceil_p + \lfloor e_f \rceil_p + \epsilon, \text{ where } |\epsilon| \leq 3, \\
&\lfloor \mathbf{s}^{\intercal}\mathbf{B}\mathbf{H}_f\mathbf{G}^{-1}(\mathbf{a}) \rceil_p = \mathbf{s}^{\intercal} \lfloor \mathbf{B}\mathbf{H}_f\mathbf{G}^{-1}(\mathbf{a}) \rceil_p + e_s, \text{ where } |e_s| \leq n \cdot \|\mathbf{s}\|_{\infty} \leq n\overline{B}.
\end{aligned}$$

*Finally, we have*

$$\begin{aligned}
\gamma(\mathbf{L}_f) &= \mu\lfloor p/2 \rceil + \mathbf{s}^{\intercal} \lfloor \mathbf{B}\mathbf{H}_f\mathbf{G}^{-1}(\mathbf{a}) \rceil_p - \lfloor \mathbf{s}^{\intercal}\mathbf{B}\mathbf{H}_f\mathbf{G}^{-1}(\mathbf{a}) + e_f \rceil_p \\
&= \mu\lfloor p/2 \rceil + \mathbf{s}^{\intercal} \lfloor \mathbf{B}\mathbf{H}_f\mathbf{G}^{-1}(\mathbf{a}) \rceil_p - \lfloor \mathbf{s}^{\intercal}\mathbf{B}\mathbf{H}_f\mathbf{G}^{-1}(\mathbf{a}) \rceil_p - \lfloor e_f \rceil_p - \epsilon \\
&= \mu\lfloor p/2 \rceil \underbrace{- e_s - \lfloor e_f \rceil_p - \epsilon}_{=e}.
\end{aligned}$$

*By the definition of* $e_f, e_s, \epsilon$*, and the setting of* $q$*, we have*

$$|e| \leq |e_s| + |\lfloor e_f \rceil_p| + |\epsilon| \leq \left\lceil \frac{(m+1)^{(d+1)}}{q/p} \overline{B} \right\rceil + n\overline{B} + 3 \leq B.$$

**Efficiency.** In the above construction, the public parameters $\mathsf{pp}$ mainly consists of three matrices $\mathbf{a} \in \mathbb{Z}_q^n, \mathbf{B} \in \mathbb{Z}_q^{n \times (m\ell)}, \mathbf{Q} \in \mathbb{Z}_q^{n \times n}$, where $n = \mathrm{poly}(\lambda, d), q = 2^{n^\delta}$, and $m = n\lceil \log q \rceil = \mathrm{poly}(\lambda, d)$. Therefore, the bit length of $\mathsf{pp}$ is $|\mathsf{pp}| = \mathrm{poly}(\lambda, d) \cdot \ell$. The shares $\mathbf{L}_0$ and $\{\mathbf{L}_i^b\}$ are $2\ell + 1$ vectors in $\mathbb{Z}_q^m$. Therefore $|\mathbf{L}_0| = |\mathbf{L}_i^b| = \mathrm{poly}(\lambda, d)$. Finally, $\mathbf{L}_f$ is a single element in $\mathbb{Z}_p$, where $p = 2^{\omega(\log \lambda)}$. Therefore, $|\mathbf{L}_f| = \mathrm{poly}(\lambda)$.

We next state non-annihilability security for $\mathbf{L}_f$ of the scheme. The proof can be found in the full version [31].

**Proposition 5.** *Assuming the small-secret adaptive LWE assumption, Construction 1 is* non-annihilable for $\mathbf{L}_f$.

## 5 KP-ABE for Bounded-Depth Circuits

In this section, we combine a succinct and weakly nearly linear secret sharing scheme that has linear function sharing, with a succinct and selectively simulation-secure IPFE scheme to obtain a compact and adaptively secure KP-ABE scheme.

**Construction 2** (KP-ABE)**.** All variables $x_\lambda$ are indexed by $\lambda$. For simplicity of notations, we suppress $\lambda$ in subscripts. Our construction uses the following two ingredients:

- A group based IPFE scheme $(\mathsf{IPFE.Setup}, \mathsf{IPFE.KeyGen}, \mathsf{IPFE.Enc}, \mathsf{IPFE.Dec})$ with modulus $p$ given by Lemma 4.
- A secret sharing scheme $(\mathsf{SS.Setup}, \mathsf{SS.ShareX}, \mathsf{SS.ShareF}, \mathsf{SS.FindCoef})$ for bounded-depth circuits as in Construction 1. Recall that the scheme has three properties. First, the shares are succinct: $\mathbf{L}_0$ and $\mathbf{L}_i^b$ are vectors in $\mathbb{Z}_q$ of length $m = \mathrm{poly}(\lambda, d)$, and $\mathbf{L}_C$ is a single element in $\mathbb{Z}_p$. Second, the scheme has weakly nearly linear reconstruction: the algorithm $\mathsf{SS.FindCoef}$ outputs an affine function $\gamma$ over $\mathbf{L}_C$ that approximately evaluates to $\mu\lfloor p/2 \rceil$. Third, the scheme has linear function sharing: $\mathsf{SS.ShareF}_{\mathsf{SS.pp},C}(\cdot,\cdot)$ is a deterministic linear function over $\mathbb{Z}_p$.

Our KP-ABE for circuits (see Definition 2) works as follows:

- $\mathsf{Setup}(1^\lambda, P)$ takes as input the security parameter $\lambda$ in unary, and a predicate $P \in \mathsf{Ckt}$. Let $\ell, d$ be the attribute length and depth for $P$. The algorithm runs and sets

$$\begin{gathered} \mathsf{SS.pp} \xleftarrow{\$} \mathsf{SS.Setup}(1^\lambda, 1^\ell, 1^d), \\ (\mathsf{impk}, \mathsf{imsk}) \xleftarrow{\$} \mathsf{IPFE.Setup}(1^\lambda, 1^N) \text{ fordimension } N = n + 1, \\ \mathsf{mpk} = (\mathsf{SS.pp}, \mathsf{impk}), \quad \mathsf{msk} = \mathsf{imsk}. \end{gathered}$$

It outputs $\mathsf{mpk}, \mathsf{msk}$.

- $\mathsf{KeyGen}(\mathsf{msk}, C)$ takes as input the master secret key $\mathsf{msk}$ and a policy $C \in \mathsf{Ckt}_{\ell,d}$. Since the secret sharing scheme has linear function sharing (Definition 11), the $\mathsf{SS.ShareF}_{\mathsf{SS.pp},C}(\cdot,\cdot)$ function is a deterministic linear function with coefficients $\mathbf{c} = (c_\mu, \mathbf{c_r})$. The $\mathsf{KeyGen}$ algorithm samples $\delta \xleftarrow{\$} \mathbb{Z}_p \backslash \{0\}$, runs

$$\mathsf{isk} \xleftarrow{\$} \mathsf{IPFE.KeyGen}(\mathsf{imsk}, [\![\delta \mathbf{c}]\!]_2),$$

  and outputs $\mathsf{sk} = ([\![\delta]\!]_2, \mathsf{isk})$ as the secret key for $C$.
- $\mathsf{Enc}(\mathsf{mpk}, \mathbf{x}, \mu)$ takes as input the master public key $\mathsf{mpk}$, an attribute $\mathbf{x} \in \{0,1\}^\ell$, and a message $\mu \in \{0,1\}$. The algorithm runs

$$\left(\mathbf{L}_0, \{\mathbf{L}_i^b\}_{i\in[\ell]}^{b\in\{0,1\}}, \mathbf{r}\right) \xleftarrow{\$} \mathsf{SS.ShareX}(\mathsf{SS.pp}), \quad \mathsf{ict} \xleftarrow{\$} \mathsf{IPFEEnc}(\mathsf{impk}, [\![(\mu, \mathbf{r})]\!]_1),$$

  and outputs $\mathsf{ct} = (\mathbf{L}^{\mathbf{x}}, \mathsf{ict})$.
- $\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}, C, \mathsf{ct}, \mathbf{x})$ takes as input the master public key $\mathsf{mpk}$, a secret key $\mathsf{sk}$, its associated policy $C$, a ciphertext $\mathsf{ct}$, and its associated attribute $\mathbf{x}$. If $P(\mathbf{x}, C) = 0$, the algorithm outputs $\perp$ and terminates. Otherwise, it parses $\mathsf{sk} = ([\![\delta]\!]_2, \mathsf{isk})$, and computes the coefficients $\mathbf{c} = (c_\mu, \mathbf{c_r})$ for $\mathsf{ShareF}_{\mathsf{SS.pp},C}(\cdot,\cdot)$ as in $\mathsf{KeyGen}$. The algorithm next parses $\mathsf{ct}$ into $\mathbf{L}^{\mathbf{x}}, \mathsf{ict}$, and runs

$$\Lambda_C \xleftarrow{\$} \mathsf{IPFE.Dec}(\mathsf{isk}, [\![\delta]\!]_2 \mathbf{c}, \mathsf{ict}), \quad (\gamma, 1^B) \xleftarrow{\$} \mathsf{SS.FindCoef}(\mathsf{SS.pp}, C, \mathbf{x}, \mathbf{L}^{\mathbf{x}}).$$

  The algorithm applies the affine function $\gamma$ homomorphically in the exponent of $G_{\mathrm{T}}$ to compute $\gamma(\Lambda_C)$. It then finds and outputs the unique $\mu' \in \{0,1\}$ (as the decrypted message) such that $\gamma(\Lambda_C) = [\![\mu' \lfloor p/2 \rceil + e]\!]_1 [\![\delta]\!]_2$, for some $e \in [-B..B]$, by enumerating over all possible $e$. Note: *We show that the scheme is correct. By the correctness of IPFE and by linear function sharing of the secret sharing scheme, we have*

$$\Lambda_C = [\![\delta(c_\mu \cdot \mu + \mathbf{c_r} \cdot \mathbf{r})]\!]_{\mathrm{T}} = [\![\delta \mathsf{SS.ShareF}_{\mathsf{SS.pp},C}(\mu, \mathbf{r})]\!]_{\mathrm{T}} = [\![\delta \mathbf{L}_C]\!]_{\mathrm{T}}.$$

  *Therefore,* $\gamma(\Lambda_C) = [\![\delta\gamma(\mathbf{L}_C)]\!]_{\mathrm{T}} = [\![\gamma(\mathbf{L}_C)]\!]_1 [\![\delta]\!]_2$. *By the correctness of the weakly nearly linear secret sharing scheme, the decryption algorithm outputs the correct bit* $\mu' = \mu$.

**Efficiency.** By Lemma 4, for MDDH dimension $k = \mathrm{poly}(\lambda)$ and input vector length $N = n + 1$, the IPFE components have bit lengths $|\mathsf{impk}|, |\mathsf{imsk}|, |\mathsf{ict}| = \mathrm{poly}(\lambda, d), |\mathsf{isk}| = \mathrm{poly}(\lambda)$. Also recall that the secret sharing components have bit lengths $|\mathsf{SS.pp}| = \mathrm{poly}(\lambda, d) \cdot \ell, |\mathbf{L}_0| = |\mathbf{L}_i^b| = \mathrm{poly}(\lambda, d), |\mathbf{L}_C| = \mathrm{poly}(\lambda)$. In the above construction,

- the master public key consists of $\mathsf{SS.pp}$ and $\mathsf{impk}$, hence has bit length $|\mathsf{mpk}| = |\mathsf{SS.pp}| + |\mathsf{impk}| = \mathrm{poly}(\lambda, d) \cdot \ell$.

- The master secret key consists of $\mathsf{imsk}$, hence has bit length $|\mathsf{msk}| = |\mathsf{imsk}| = \mathrm{poly}(\lambda, d)$.
- A secret key consists of a single $\mathsf{isk}$, and $[\![\delta]\!]_2$ in $G_2$, hence has bit length $|\mathsf{sk}| = |\mathsf{isk}| + |G_2| = \mathrm{poly}(\lambda)$.
- A ciphertext consists of a single $\mathsf{ict}$, and $\ell + 1$ shares, hence has bit length $|\mathsf{ct}| = |\mathsf{ict}| + (\ell + 1)|\mathbf{L}_0| = \mathrm{poly}(\lambda, d) \cdot \ell$.

We now state adaptive IND-CPA security of the scheme. The proof can be found in the full version [31].

**Proposition 6.** *Suppose in Construction 2, the IPFE scheme is selectively simulation-secure, and the secret sharing scheme is non-annihilable for* $\mathbf{L}_f$*. Then the constructed KP-ABE scheme is adaptively IND-CPA in GGM.*

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

## References

1. Abdalla, M., Bourse, F., De Caro, A., Pointcheval, D.: Simple functional encryption schemes for inner products. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 733–751. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46447-2_33
2. Agrawal, S., Libert, B., Stehlé, D.: Fully secure functional encryption for inner products, from standard assumptions. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9816, pp. 333–362. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53015-3_12
3. Agrawal, S., Wichs, D., Yamada, S.: Optimal broadcast encryption from LWE and pairings in the standard model. In: Pass, R., Pietrzak, K. (eds.) TCC 2020. LNCS, vol. 12550, pp. 149–178. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64375-1_6
4. Agrawal, S., Yamada, S.: Optimal broadcast encryption from pairings and LWE. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020. LNCS, vol. 12105, pp. 13–43. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45721-1_2
5. Attrapadung, N.: Dual system encryption framework in prime-order groups via computational pair encodings. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10032, pp. 591–623. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53890-6_20
6. Attrapadung, N., Libert, B., de Panafieu, E.: Expressive key-policy attribute-based encryption with constant-size ciphertexts. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 90–108. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19379-8_6

7. Attrapadung, N., Tomida, J.: Unbounded dynamic predicate compositions in ABE from standard assumptions. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020. LNCS, vol. 12493, pp. 405–436. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64840-4_14
8. Beimel, A.: Secure schemes for secret sharing and key distribution. Ph.D. thesis, Technion-Israel Institute of Technology (1996)
9. Berkowitz, S.J.: On computing the determinant in small parallel time using a small number of processors. Inf. Process. Lett. **18**(3), 147–150 (1984). https://doi.org/10.1016/0020-0190(84)90018-8
10. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: 2007 IEEE Symposium on Security and Privacy, pp. 321–334. IEEE Computer Society Press (2007). https://doi.org/10.1109/SP.2007.11
11. Boneh, D., et al.: Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 533–556. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55220-5_30
12. Boneh, D., Sahai, A., Waters, B.: Functional encryption: definitions and challenges. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 253–273. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19571-6_16
13. Brakerski, Z., Tsabary, R., Vaikuntanathan, V., Wee, H.: Private constrained PRFs (and more) from LWE. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017. LNCS, vol. 10677, pp. 264–302. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70500-2_10
14. Brakerski, Z., Vaikuntanathan, V.: Constrained key-homomorphic PRFs from standard lattice assumptions. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015. LNCS, vol. 9015, pp. 1–30. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46497-7_1
15. Buntrock, G., Damm, C., Hertrampf, U., Meinel, C.: Structure and importance of logspace-MOD class. Math. Syst. Theory **25**(3), 223–237 (1992). https://doi.org/10.1007/BF01374526
16. Chen, Y.-H., Chung, K.-M., Liao, J.-J.: On the complexity of simulating auxiliary input. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10822, pp. 371–390. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78372-7_12
17. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: 54th FOCS, pp. 40–49. IEEE Computer Society Press (2013). https://doi.org/10.1109/FOCS.2013.13
18. Garg, S., Gentry, C., Sahai, A., Waters, B.: Witness encryption and its applications. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) 45th ACM STOC, pp. 467–476. ACM Press (2013). https://doi.org/10.1145/2488608.2488667
19. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: conceptually-simpler, asymptotically-faster, attribute-based. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 75–92. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_5
20. Goldwasser, S., Kalai, Y.T., Popa, R.A., Vaikuntanathan, V., Zeldovich, N.: How to run turing machines on encrypted data. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8043, pp. 536–553. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40084-1_30

21. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Functional encryption with bounded collusions via multi-party computation. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 162–179. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_11
22. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Attribute-based encryption for circuits. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) 45th ACM STOC, pp. 545–554. ACM Press (2013). https://doi.org/10.1145/2488608.2488677
23. Gorbunov, S., Vinayagamurthy, D.: Riding on asymmetry: efficient ABE for branching programs. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015. LNCS, vol. 9452, pp. 550–574. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48797-6_23
24. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Juels, A., Wright, R.N., De Capitani di Vimercati, S. (eds.) ACM CCS 2006, pp. 89–98. ACM Press (2006). https://doi.org/10.1145/1180405.1180418. available as Cryptology ePrint Archive Report 2006/309
25. Jetchev, D., Pietrzak, K.: How to fake auxiliary input. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 566–590. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54242-8_24
26. Karchmer, M., Wigderson, A.: On span programs. In: Proceedings of Structures in Complexity Theory, pp. 102–111 (1993)
27. Kitagawa, F., Nishimaki, R., Tanaka, K., Yamakawa, T.: Adaptively secure and succinct functional encryption: improving security and efficiency, simultaneously. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. LNCS, vol. 11694, pp. 521–551. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26954-8_17
28. Kowalczyk, L., Wee, H.: Compact adaptively secure abe for $\mathsf{NC}^1$ from $k$-Lin. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019. LNCS, vol. 11476, pp. 3–33. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17653-2_1
29. Lewko, A., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 62–91. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_4
30. Lewko, A., Waters, B.: New proof methods for attribute-based encryption: achieving full security through selective techniques. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 180–198. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_12
31. Li, H., Lin, H., Luo, J.: ABE for circuits with constant-size secret keys and adaptive security. Cryptology ePrint Archive, Report 2022/659 (2022). https://eprint.iacr.org/2022/659
32. Lin, H., Luo, J.: Succinct and adaptively secure ABE for ABP from $k$-Lin. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020. LNCS, vol. 12493, pp. 437–466. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64840-4_15
33. Micciancio, D., Peikert, C.: Trapdoors for lattices: simpler, tighter, faster, smaller. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 700–718. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_41
34. Mulmuley, K.: A fast parallel algorithm to compute the rank of a matrix over an arbitrary field. Combinatorica **7**(1), 101–104 (1987). https://doi.org/10.1007/BF02579205

35. Okamoto, T., Takashima, K.: Fully secure functional encryption with general relations from the decisional linear assumption. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 191–208. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14623-7_11
36. Quach, W., Wee, H., Wichs, D.: Laconic function evaluation and applications. In: Thorup, M. (ed.) 59th FOCS, pp. 859–870. IEEE Computer Society Press (Oct 2018). https://doi.org/10.1109/FOCS.2018.00086
37. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005). https://doi.org/10.1007/11426639_27
38. Takashima, K.: Expressive attribute-based encryption with constant-size ciphertexts from the decisional linear assumption. In: Abdalla, M., De Prisco, R. (eds.) SCN 2014. LNCS, vol. 8642, pp. 298–317. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10879-7_17
39. Tsabary, R.: Fully secure attribute-based encryption for $t$-CNF from LWE. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. LNCS, vol. 11692, pp. 62–85. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26948-7_3
40. Waters, B.: A punctured programming approach to adaptively secure functional encryption. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9216, pp. 678–697. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48000-7_33
41. Wee, H.: Attribute-hiding predicate encryption in bilinear groups, revisited. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017. LNCS, vol. 10677, pp. 206–233. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70500-2_8
42. Yamada, S., Attrapadung, N., Hanaoka, G., Kunihiro, N.: A framework and compact constructions for non-monotonic attribute-based encryption. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 275–292. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54631-0_16
43. Zhang, K., et al.: Practical and efficient attribute-based encryption with constant-size ciphertexts in outsourced verifiable computation. In: Chen, X., Wang, X., Huang, X. (eds.) ASIACCS 16, pp. 269–279. ACM Press (2016)