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

Hanjun Li Huijia Lin Ji Luo

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

May 2022

Abstract

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

- 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 construction 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, $|\mathsf{ct}_x| = |x| \,\mathrm{poly}(d, \lambda)$.
- 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, in particular, $|\mathsf{sk}_f| = \mathrm{poly}(\lambda)$ and $|\mathsf{ct}_x| = \mathrm{poly}(|x|, \lambda)$. 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.

Contents

1 Introduction

Attribute-based encryption (ABE) [SW05,GPSW06] is a novel generalization of public-key encryption for enforcing fine-grained access control. In this work, we focus on 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 sk_f is tied to a policy f and a ciphertext $\mathsf{ct}_{\mathbf{x}}$ encrypting a message μ 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 [BSW07], which enables instead the ciphertexts 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 Figures 1 and 2):

- The work of [BGG+14] 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 [ALdP11,YAHK14,Tak14,Att16,ZGT+16,AT20,LL20] 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 [AY20,AWY20] 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 [AY20] or knowledge assumptions [AWY20].

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 computations

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

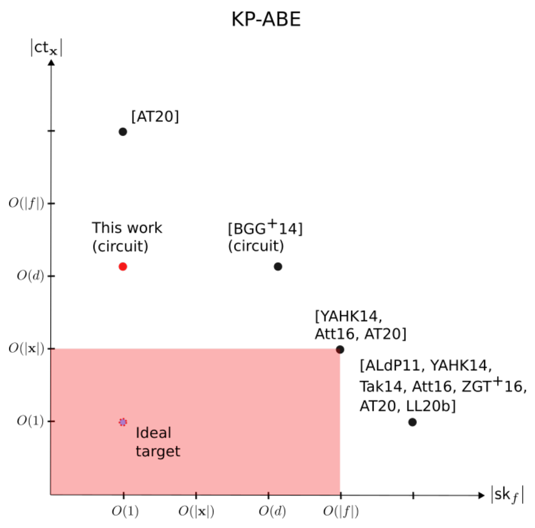

Figure 1. Efficiency comparison for KP-ABE schemes. The pink region highlights succinctness for $|\mathsf{ct}_{\mathbf{x}}|$ and $|\mathsf{sk}_f|$. This work and [BGG+14] are KP-ABE schemes for circuits, while the rest of the schemes are for low-depth computation.

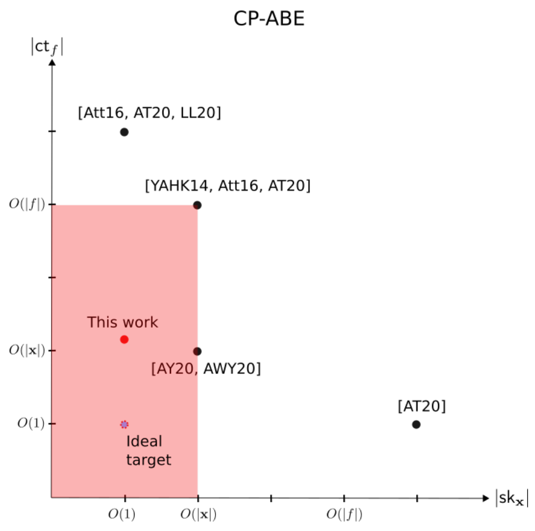

Figure 2. Efficiency comparison for CP-ABE schemes. The pink region highlights succinctness for $|\mathsf{ct}_f|$ and $|\mathsf{sk}_{\mathbf{x}}|$. All the included schemes are CP-ABE for low-depth computation.

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 [GGH+13,GGSW13,KNTY19] 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 $\mathrm{poly}(|f|)$. The only work that manages to obtain ABE for RAM with constant-size keys [GKP+13] 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 [BGG+14]. 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 [AY20].

Theorem (KP-ABE)**.** *Assuming LWE, in the generic (pairing) group model, there is a KP-ABE for circuits (Construction 3) that achieves selective security and has key size* $|\mathsf{sk}_C| = \mathrm{poly}(\lambda)$

(concretely, containing 3 group elements) and ciphertext size $|\mathsf{ct}_{\mathbf{x}}| = |\mathbf{x}| \operatorname{poly}(\lambda, d)$*, 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 (Construction 5) that achieves very selective security, and has constant-size keys* $|\mathsf{sk}_{\mathbf{x}}| = \operatorname{poly}(\lambda)$ *(concretely, containing 2 group elements) and ciphertexts of size* $|\mathsf{ct}_f| = |\mathbf{x}|^2 \operatorname{poly}(\lambda)$ *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 μ 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, $\lambda = \operatorname{poly}(|x|)$ in KP-ABE or $\lambda = \operatorname{poly}(|f|)$ in CP-ABE. As a result, complexity leveraging is not a viable solution when aiming for constant-size keys, as key size $\operatorname{poly}(\lambda)$ would already depends on $|x|$ or $|f|$.

Instead, we show that if our constructions of KP- and CP-ABE assume adaptive LWE instead of plain LWE, then the schemes achieve *adaptive security* and our reduction only incurs polynomial security loss. The adaptive LWE assumption introduced in [QWW18] 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 3) and the CP-ABE scheme (Construction 5) are adaptively secure.*

In the literature, the ABE schemes for circuits based on lattices [GVW12,BGG⁺14] 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 [Tsa19]. Alternatively, there are schemes based on indistinguishability obfuscation or functional encryption for all circuits that are adaptively secure [Wat15, KNTY19], 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 [AY20,AWY20] that combine [BGG⁺14] with pairing groups inherit the selective security of the former (even if assuming adaptive LWE).

2 Technical Overview

High-Level Ideas. Let's focus on our KP-ABE scheme for circuits first. The celebrated constructions of KP-ABE for circuits from LWE [GVW13,BGG+14] have keys of size $\mathrm{poly}(d,\lambda)$ and 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 μ as follows.

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

One can homomorphically evaluate any circuit f on the attribute encoding to obtain $\mathbf{s}^\top(\mathbf{B}_f-f(x)\mathbf{G})+\mathbf{e}_f^\top$. To decrypt, the secret key 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 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 matrixes $\mathbf{A}||\mathbf{B}_f$ 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 matrixes $\mathbf{A}_i$'s (contained in 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) [QWW18], 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 μ as follows.

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

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 μ follows directly from the pseudorandomness of LWE samples $\mathbf{s}^\top((\mathbf{A}_1||\cdots||\mathbf{A}_\ell)-\mathbf{x}\otimes\mathbf{G})+\mathbf{e}'$ and $\mathbf{s}^\top\mathbf{a}+e'''$. 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 [QWW18] to obtain a *nearly linear secret sharing scheme* for circuits. It contains two parts.

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

Note that we round $\mathbf{B}_f\mathbf{r}$ from modulus q of poly(d) length to p of poly(λ) length so that the component L_f in the secret sharing that depends on f and μ 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 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 [AY20,AWY20] to compute $\delta_f \cdot L_f$ in the exponent instead, where δ_f is an independent and random scalar chosen at key generation time. In GGM, the presence of δ_f prevents adversaries from meaningfully "combining" information from multiple L_f for different f.

Comparison with [AY20,AWY20]. Our way of combining lattice-based LSS with pairing-based IPFE differs from that of [AY20,AWY20], 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 [AY20,AWY20] 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 [QWW18]. 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 μ. The proof of non-annihilability combines techniques from AB-LFE and leakage simulation [JP14,CCL18]. The work of [AY20,AWY20] 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 [AY20,AWY20]. 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 [ABDP15] 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). IPFE can be based on a variety of assumptions such as MDDH, LWE, DCR [ABDP15,ALS16].

A *nearly* linear secret sharing scheme enables generating shares $L_f, L_0, \{L_i^b\}$ associated with a policy f and some secret μ, 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 μ if and only if $f(\mathbf{x}) = 0$:

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

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

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 μ, and L_f to be linear in μ and the randomness $\mathbf{r}$ of 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 ict's and the coefficients (of L_f as a function of $\mu, \mathbf{r}$) into 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 μ exactly. We stress again that different from [AY20,AWY20], 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 ict's, and a "selection" vector according to $\mathbf{x}$ in 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 matrixes $\mathbf{B} = (\mathbf{A}_1 || \cdots || \mathbf{A}_{|\mathbf{x}|})$, 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.

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

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 [QWW18]. 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 μ 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.

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

Decryption proceeds by first running $\mathsf{EvalCX}(\mathbf{c}_1, f, \mathbf{x})$ 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 computes $c_0 - \mathbf{c}_f^\intercal\mathbf{r} = \mu\lfloor q/2\rceil + (e - \mathbf{e}_f^\intercal\mathbf{r})$ and round it to recover μ.

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 μ 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 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 μ. We can now recast the above rounded AB-LFE scheme into a secret sharing scheme with L_f of bit-length $\Theta(\log p)$, which could be independent of depth 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 Equation (1), to obtain KP-ABE, we will use a pairing-based IPFE to compute L_f in the exponent. Specifically, the IPFE secret key isk encodes $(\lfloor\mathbf{B}_f\mathbf{G}^{-1}(\mathbf{a})\rceil_p, \lfloor p/2\rceil)$, and the IPFE ciphertext 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 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 μ 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 Equations (3, 4), when $f(\mathbf{x}) = 1$ we have

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

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 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 [QWW18] 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 μ 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 γ 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 [JP14,CCL18]. Crucially, because e'_f, e_s have polynomial range, the simulation can run in polynomial time. More precisely, the leakage simulation lemma of [CCL18] 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 ε, there is a simulator h simulating Z as $h(X)$ such that (X, Z) and $(X, h(X))$ are (s, ε)-indistinguishable. Furthermore, the running time of h is $\mathrm{O}(s\varepsilon^{-2}2^{|Z|})$. Suppose for contradiction that there is an adversary $\mathcal{A}$ of size $s = \mathrm{poly}(\lambda)$ winning the non-annihilability game with probability $2\varepsilon \geq 1/\mathrm{poly}(\lambda)$. Consider the joint distribution $\mathcal{D}$ of running the game with $\mathcal{A}$, defined in the first line below:

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

Using (X, Z), one can emulate L_f as (cf. Equation (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, ε^{-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 ψ. However, there is a subtle issue: $Z = h(X)$ depends on ψ 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 γ chosen by $\mathcal{A}$ before ψ 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 [AY20,AWY20] to "isolate" each ABE secret key in GGM by multiplying it with a fresh random element δ.

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

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

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

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

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 δ_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 δ_j. Since each δ_j is sampled independently at random, by Schwartz–Zippel, with all but negligible probability, γ evaluates to zero only if all γ_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 γ_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 μ encoded in them.

Our KP-ABE Scheme

$\mathsf{Setup}(1^\lambda)$: Output $\mathsf{mpk} = \mathsf{impk}$ for IPFE, 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 μ from
$$[\![\delta L_f]\!]_{\mathrm{T}} - [\![\delta]\!]_{\mathrm{T}} \lfloor \mathbf{c}_f^{\intercal} \mathbf{G}^{-1}(\mathbf{a}) \rceil_p = [\![\delta(\mu \lfloor p/2 \rceil - e_s - e'_f)]\!]_{\mathrm{T}}.$$

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 O(1) group elements, in fact only *three* using the IPFE scheme of [ALS16] 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 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 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 mpk and a ciphertext ct^* from the challenger $\mathcal{C}$. It is then allowed to repeatedly query secret keys 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 μ correctly.

H_0

$$\begin{tikzcd}[column sep=12em]
\mathcal{C} & \mathcal{A} \arrow[l, "\mathbf{x}^*"'] \\
{} \arrow[r, "{\mathsf{mpk}, \mathsf{ct}^* = \mathbf{L}^{\mathbf{x}^*}, \mathsf{ict}(\llbracket \mathbf{s}, \mu \rrbracket_1)}"] & {} \\
{} & {} \arrow[l, "f_j"'] \\
{} \arrow[r, "{\mathsf{sk}_j = (\llbracket \delta_j \rrbracket_2, \mathsf{isk}(\llbracket \delta_j(\lfloor \mathbf{B}_{f_j} \mathbf{G}^{-1}(\mathbf{a}) \rceil_p, \lfloor p/2 \rceil) \rrbracket_2))}"] & {} \arrow[r, phantom, "\text{\} repeat}"] & {}
\end{tikzcd}$$

$$\xRightarrow{\text{IPFE}}$$

H_1

$$\begin{tikzcd}[column sep=12em]
\mathcal{C} & \mathcal{A} \arrow[l, "\mathbf{x}^*"'] \\
{} \arrow[r, "{\mathsf{mpk}, \mathsf{ct}^* = \mathbf{L}^{\mathbf{x}^*}, \widetilde{\mathsf{ict}}(\perp)}"] & {} \\
{} & {} \arrow[l, "f_j"'] \\
{} \arrow[r, "{\mathsf{sk}_j = (\llbracket \delta_j \rrbracket_2, \widetilde{\mathsf{isk}}(\llbracket \delta_j L_{f_j} \rrbracket_2))}"] & {} \arrow[r, phantom, "\text{\} repeat}"] & {}
\end{tikzcd}$$

Note that we can generate the IPFE ciphertext $\mathsf{ict}(\llbracket \mathbf{s}, \mu \rrbracket_1)$ before any IPFE secret keys $\mathsf{isk}(\llbracket \delta(\lfloor \mathbf{B}_{f_j} \mathbf{G}^{-1}(\mathbf{a}) \rceil_p, \lfloor p/2 \rceil) \rrbracket_2)$. Relying on the selective simulation security of IPFE, we can (as summarized in H_1 above) replace $\mathsf{ict}(\llbracket \mathbf{s}, \mu \rrbracket_1)$ with a simulated ciphertext $\widetilde{\mathsf{ict}}(\perp)$, and each $\mathsf{isk}(\llbracket \delta_j(\lfloor \mathbf{B}_{f_j} \mathbf{G}^{-1}(\mathbf{a}) \rceil_p, \lfloor p/2 \rceil) \rrbracket_2)$ with a simulated secret key $\widetilde{\mathsf{isk}}(\llbracket \delta_j L_{f_j} \rrbracket_2)$ using their inner products.

In GGM, we can now argue that $\mathcal{A}$ only learns information about μ 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 μ.

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 Equation (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 EvalCX, in Equation (4) becomes super-polynomial. This again makes the ABE decryption inefficient.

Fortunately, for Boolean formulae, the work of [GV15] develops specialized homomorphic evaluation procedures EvalF, 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 EvalC, EvalCX by EvalF, EvalFX. We summarize our modified secret sharing scheme below (Setup, 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 $L_f = \mathbf{s}^{\mathsf{T}} \mathbf{B}_f \mathbf{G}^{-1}(\mathbf{a}) + \mu \lfloor p/2 \rceil + e$.

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

Otherwise, compute $\mathbf{c}_f = \mathsf{EvalFX}(\mathbf{L}^{\mathbf{x}}, f, \mathbf{x})$,
and find μ from $L_f - \mathbf{c}_f^{\mathsf{T}} \mathbf{G}^{-1}(\mathbf{a}) = \mu \lfloor p/2 \rceil + (e - e'_f)$.

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 Sel, Encode using an IPFE with constant-size isk's, such that $\mathsf{Sel}(\llbracket \mathbf{x} \rrbracket_2)$ and $\mathsf{Encode}(\llbracket \mathbf{L}_0, \{\mathbf{L}_i^b\} \rrbracket_1)$ decrypts exactly to $\llbracket \mathbf{L}^{\mathbf{x}} \rrbracket_{\mathrm{T}}$. We obtain a CP-ABE scheme for Boolean formulae as summarized below.

Our CP-ABE Scheme

$$\begin{aligned}
\mathsf{Setup}(1^\lambda) &: \text{Output } \mathsf{mpk} = \mathsf{impk} \text{ for IPFE and } \mathsf{pp} \text{ for secret sharing,} \\
&\qquad \text{and } \mathsf{msk} = \mathsf{imsk} \text{ for IPFE.} \\
\mathsf{KeyGen}(\mathsf{msk}, \mathbf{x}) &: \text{Sample } \delta \xleftarrow{\$} \mathbb{Z}_p. \\
&\quad \text{Output } \mathsf{sk} = ([\![\delta]\!]_2, \mathsf{Sel}([\![\delta \mathbf{x}]\!]_2)). \\
\mathsf{Enc}(\mathsf{mpk}, f, \mu) &: \text{Compute } (\mathbf{L}_0, \{\mathbf{L}_i^b\}, \mathbf{s}) \xleftarrow{\$} \mathsf{ShareX}(\mathsf{pp}) \\
&\qquad \text{and } L_f \xleftarrow{\$} \mathsf{ShareF}'(\mathsf{pp}, f, \mu, \mathbf{s}). \\
&\quad \text{Output } \mathsf{ct} = ([\![L_f]\!]_1, \mathsf{Encode}([\![\mathbf{L}_0, \{\mathbf{L}_i^b\}]\!]_1)). \\
\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}, \mathbf{x}, \mathsf{ct}, f) &: \text{Run IPFE decryption to recover } [\![\delta \mathbf{L}^{\mathbf{x}}]\!]_{\mathrm{T}}. \\
&\quad \text{Compute } [\![\delta \mathbf{c}_f]\!]_{\mathrm{T}} = \mathsf{EvalFX}([\![\delta \mathbf{L}^{\mathbf{x}}]\!]_{\mathrm{T}}, f, \mathbf{x}), \\
&\qquad \text{and find } \mu \text{ from} \\
&\qquad [\![L_f]\!]_1 [\![\delta]\!]_2 - [\![\delta \mathbf{c}_f^{\mathsf{T}}]\!]_{\mathrm{T}} \mathbf{G}^{-1}(\mathbf{a}) = [\![\delta(\mu \lfloor p/2 \rceil + (e - e_f'))]\!]_{\mathrm{T}}.
\end{aligned}$$

We now describe the $\mathsf{Sel}, \mathsf{Encode}$ functions. Let $\ell = |\mathbf{x}|$ denote the length of $\mathbf{x}$. The 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 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: $|\mathsf{sk}| = O(1)$ 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.

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

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 [ALS16, LL20], 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 [BSW11]. 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 [ABDP15], 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 [ABDP15] 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 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 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 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 pp. We show that this is implied by the *adaptive* LWE assumption formulated in [QWW18].

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 λ be the security parameter, which runs through $\mathbb{N}$. Except in the definitions, we suppress λ for brevity. Efficient algorithms are probabilistic polynomial-time (PPT) Turing machines. Efficient adversaries are non-uniform PPT Turing machines, or equivalently families of polynomial-sized circuits. We denote by $\mathsf{H}_0 \approx \mathsf{H}_1$ (resp. $\approx_s$, $\equiv$) computational indistinguishability (resp. statistical indistinguishability, identity) of two distributions or experiments.

We write $[a..b]$ for the set $\{a, a+1, \dots, b\}$ 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 $\|\cdot\|_\infty$.

Schwartz–Zippel Lemma. 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* $\Pr\big[\mathbf{z} \xleftarrow{\$} \mathbb{Z}_p^Z : P(\mathbf{z}) = 0\big] \le d/p$.

3.1 Attribute-Based Encryption

Definition 1 (ABE [GPSW06]). Let $\mathcal{P} = \{\mathcal{P}_\lambda\}_{\lambda \in \mathbb{N}}$ be a sequence of predicate families with $\mathcal{P}_\lambda = \big\{P : X_P \times Y_P \to \{0, 1\}\big\}$. 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^λ 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 msk and some $y \in Y_P$, and outputs a secret key sk.
- $\mathsf{Enc}(\mathsf{mpk}, x, \mu)$ takes as input the master public key mpk, some $x \in X_P$, and a message $\mu \in \{0,1\}$, and it outputs a ciphertext ct.
- $\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}, y, \mathsf{ct}, x)$ takes as input the master public key mpk, a secret key sk, its associated y, a ciphertext 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

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

There are two major variants of ABE, key-policy (KP) and ciphertext-policy (CP). 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. In CP-ABE, the roles of X_P, Y_P are swapped. We refer to the function (resp. input) as the policy (resp. attribute). When we want to emphasize x or y is a function represented by formula or circuit (resp. bit-string), we write f or C (resp. $\mathbf{x}$) instead.

Security. We consider adaptive IND-CPA security of ABE.

Definition 2 (ABE security [LOS+10]). An ABE scheme (Definition 1) is *adaptively secure* if $\mathsf{Exp}^0_{\mathrm{CPA}} \approx \mathsf{Exp}^1_{\mathrm{CPA}}$, where $\mathsf{Exp}^b_{\mathrm{CPA}}(1^\lambda)$ with adversary $\mathcal{A}$ proceeds as follows:

- **Setup.** The challenger launches $\mathcal{A}(1^\lambda)$ and receives from it a predicate $P \in \mathcal{P}_\lambda$. The challenger runs $(\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, P)$ and sends mpk to $\mathcal{A}$.
- **Query I.** The following is repeated for arbitrarily many rounds determined by $\mathcal{A}$: In each round, $\mathcal{A}$ submits some $y_j \in Y_P$ for a secret key. Upon this query, the challenger runs $\mathsf{sk}_j \xleftarrow{\$} \mathsf{KeyGen}(\mathsf{msk}, y_j)$ and sends sk_j to $\mathcal{A}$.
- **Challenge.** $\mathcal{A}$ submits some $x^* \in X_P$ for the challenge ciphertext. The challenger runs $\mathsf{ct} \xleftarrow{\$} \mathsf{Enc}(\mathsf{mpk}, x^*, b)$ and sends ct to $\mathcal{A}$.
- **Query II.** Same as Query I.
- **Guess.** $\mathcal{A}$ outputs a bit b'. The outcome of the experiment is b' if $P(x^*, y_j) = 0$ for all y_j queried in Query I/II. Otherwise, the outcome is set to 0.

Furthermore, an ABE scheme is selectively secure if the same condition above holds except that $\mathcal{A}$ must choose the challenge x^* before it receives mpk. An ABE scheme is very selectively secure if the same condition above holds except that $\mathcal{A}$ must choose both the challenge x^* and all the key queries $\{y_j\}$ before it receives mpk.

Computation Model. In this paper, we construct schemes for both variants: KP-ABE and CP-ABE. Our KP scheme supports bounded-depth circuits for any polynomial bound, and our CP schemes support all polynomial-sized permutation branching programs of width five (5-permutation branching programs, or 5-PBP), which are known to be equivalent to NC^1 [Bar86]. The definition of 5-PBP is standard in the literature, for which we refer the readers to [GV15].

Definition 3 (KP-ABE for circuits)**.** A *KP-ABE for (bounded-depth) circuits* is ABE for $\mathcal{P}^{\mathsf{Ckt}}$:[2]

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

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

Definition 4 (CP-ABE for 5-PBP)**.** A *CP-ABE for 5-PBP* is ABE for $\mathcal{P}^{\mathsf{5PBP}}$:

$$X^{\mathsf{5PBP}}_{\lambda,\ell} = \{\text{5-PBP } f : \{0,1\}^\ell \to \{0,1\} \text{ of length } \le s_\lambda\}, \quad Y^{\mathsf{5PBP}}_{\lambda,\ell} = \{0,1\}^\ell,$$
$$P^{\mathsf{5PBP}}_{\lambda,\ell}(f, \mathbf{x}) = \neg f(\mathbf{x}), \quad \mathcal{P}^{\mathsf{5PBP}}_\lambda = \{P^{\mathsf{5PBP}}_{\lambda,\ell} \,|\, \ell \in \mathbb{N}\}, \quad \mathcal{P}^{\mathsf{5PBP}} = \{\mathcal{P}^{\mathsf{5PBP}}_\lambda\}_{\lambda\in\mathbb{N}}.$$

Here, s_λ is a super-polynomial function (specified by the constructions). As an input to Setup, the predicate $P^{\mathsf{5PBP}}_{\lambda,\ell}$ is represented by 1^ℓ.

It is worth noting that for KP-ABE, since Setup takes the unary representation of ℓ, d, which will be polynomial in λ, as input, they are bounded by *that* polynomial once the system is set up. However, d can be up to D_λ, which is super-polynomial in λ, 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. In contrast, for CP-ABE, each $X^{\mathsf{5PBP}}_{\lambda,\ell}$ contains all 5-PBP of size up to s_λ, again super-polynomial in λ, so the branching program size is not bounded by any polynomial even after the system has been set up, i.e., our CP-ABE for 5-PBP supports unbounded polynomial-size 5-PBP.

Compactness and Succinctness. Since KeyGen, 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 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 5 (ABE efficiency)**.** For KP-ABE for circuits (of depth at most d), it 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 $|C|$.

For CP-ABE for 5-PBP, it has

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

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

- *succinct ciphertexts* if $|\mathsf{ct}| = |\mathbf{x}|\,\mathrm{poly}(\lambda)$ is independent of $|f|$.

The CP-ABE is *doubly succinct* if it has succinct keys and ciphertexts.

We remark that an ideally succinct component should be of length $\mathrm{poly}(\lambda)$. Nevertheless, our versions defined above are still meaningful. For KP-ABE, the circuit size can be much larger than its depth. For CP-ABE, the branching program size can be much larger than its input length.

3.2 Lattice Tools

Homomorphic Evaluation. We use the following abstraction of homomorphic evaluation for ABE over lattices, developed in a series of works [GSW13,BGG+14,GV15] with the syntax in [BV15,BTVW17]. The actual algorithms we use are that for ABE for circuits in [BGG+14] (slightly changed), and that for ABE for 5-PBP in [GV15]. The former handles general polynomial-size circuits, but the noise grows exponentially in the circuit depth. The latter is specialized for 5-PBP, and the noise grows polynomially in the branching program size.

In our version of the algorithms from [BGG+14], instead of using $\mathbf{G}$ as the 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 [BGG+14])**.** EvalC *and* EvalCX *are two efficient deterministic algorithms. Let n, ℓ, 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 ℓ-bit string (row vector), and C a circuit of depth d with input length ℓ. 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

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

Lemma 3 (homomorphic evaluation for 5-PBP [GV15])**.** EvalF *and* EvalFX *are two efficient deterministic algorithms. Let n, ℓ, 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 $n \times (\ell+1)m$, $\mathbf{x}$ an ℓ-bit string (row vector), and f a 5-PBP of length ℓ_{BP} with input length ℓ. The algorithms work as follows:*

- $\mathsf{EvalF}(\mathbf{B}, f)$ *outputs* $\mathbf{H}_f \in \mathbb{Z}^{(\ell+1)m \times m}$*;*
- $\mathsf{EvalFX}(\mathbf{B}, f, \mathbf{x})$ *outputs* $\widehat{\mathbf{H}}_{f,\mathbf{x}} \in \mathbb{Z}^{(\ell+1)m \times m}$.

The outputs satisfy

$$\|\mathbf{H}_f^{\mathsf{T}}\|_\infty, \|\widehat{\mathbf{H}}_{f,\mathbf{x}}^{\mathsf{T}}\|_\infty \leq 3m\ell_{\mathrm{BP}} + 1, \quad \big(\mathbf{B} - (1, \mathbf{x}) \otimes \mathbf{G}\big)\widehat{\mathbf{H}}_{f,\mathbf{x}} = \mathbf{B}\mathbf{H}_f - f(\mathbf{x})\mathbf{G}.$$

Gadget Matrix [MP12]. Let n, q be positive integers and $m = n\lceil \log_2 q \rceil$. The gadget matrix is $\mathbf{G} = \mathbf{g}^{\mathsf{T}} \otimes \mathbf{I}_n$, where $\mathbf{g}^{\mathsf{T}} = (2^0, 2^1, \ldots, 2^{\lceil \log_2 q \rceil - 1})$. 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 adaptive learning with errors (LWE) assumption, a natural variant of LWE first proposed in [QWW18]:

Definition 6 (adaptive LWE, adapted from [QWW18])**.** We suppress the security parameter λ and all the parameters are dependent on λ. Let n be the dimension, q the modulus, χ the error distribution, $m = n\lceil \log_2 q \rceil$, and $\mathbf{G}$ the gadget matrix. The adaptive LWE assumption $\mathrm{ALWE}_{n,q,\chi}$ states that $\mathsf{Exp}^0_{\mathrm{ALWE}} \approx \mathsf{Exp}^1_{\mathrm{ALWE}}$, where $\mathsf{Exp}^b_{\mathrm{ALWE}}(1^n, q, \chi)$ with adversary $\mathcal{A}$ proceeds as follows:

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

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

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

For our KP-ABE, we need a small-secret variant of the adaptive LWE assumption, which we show is implied by the adaptive LWE assumption:

Definition 7 (small-secret adaptive LWE)**.** Using the notations in adaptive LWE (Definition 6), the small-secret adaptive LWE assumption $\mathrm{sALWE}_{n,q,\chi}$ states that $\mathsf{Exp}^0_{\mathrm{sALWE}} \approx \mathsf{Exp}^1_{\mathrm{sALWE}}$, 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,

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

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

Lemma 4 (small-secret adaptive LWE)**.** *The small-secret adaptive LWE assumption* $\mathrm{sALWE}_{n,q,\chi}$ *holds if the adaptive LWE assumption* $\mathrm{ALWE}_{n,q,\chi}$ *holds and* χ *is symmetric (i.e.,* $-\chi \equiv \chi$*).*

Proof (Lemma 4). We follow the ideas in [ACPS09] to transform LWE samples into small-secret LWE samples. Suppose for contradiction that there exists an efficient distinguisher $\mathcal{A}$ for the small secret adaptive LWE experiments $\mathsf{Exp}^0_{\mathrm{sALWE}}$ and $\mathsf{Exp}^1_{\mathrm{sALWE}}$. We construct an efficient distinguisher $\mathcal{B}$ for the adaptive LWE experiments $\mathsf{Exp}^0_{\mathrm{ALWE}}$ and $\mathsf{Exp}_{\mathrm{ALWE}}$ as follows:

1. $\mathcal{B}$ launches $\mathcal{A}$ with fresh randomness $r_{\mathcal{A}}$. It receives from $\mathcal{A}$ two lengths ℓ, m' in unary, and forwards $1^\ell, 1^{n^2+m'}$ to the adaptive LWE experiment.

2. $\mathcal{B}$ receives back $\mathbf{A}, \mathbf{B}$ which have shapes $n \times (n^2+m')$ and $n \times (\ell+1)m$. It goes through the first n^2 columns of $\mathbf{A}$, to accumulate n columns that form an invertible matrix. If $\mathcal{B}$ fails, then abort. Otherwise, it sets $\mathbf{Q}^{-1}$ to be this invertible matrix, and $\mathbf{A}'$ to be the last m' columns of $\mathbf{A}$. It sends $(\mathbf{QA}', \mathbf{QB}, \mathbf{Q})$ to $\mathcal{A}$.

3. $\mathcal{B}$ receives an input $\mathbf{x} \in \{0,1\}^\ell$ from $\mathcal{A}$, and forwards $\mathbf{x}$ to the adaptive LWE experiment.

4. $\mathcal{B}$ receives back two vectors $\mathbf{c}, \mathbf{d}$. It defines $\mathbf{q}$ to be the entries of $\mathbf{c}$ corresponding to the columns that form $\mathbf{Q}^{-1}$, and $\mathbf{c}'$ to be the last m' entries of $\mathbf{c}$. It then sets

$$\begin{aligned}\widetilde{\mathbf{c}}^{\mathsf{T}} &= \mathbf{q}^{\mathsf{T}}\mathbf{QA}' - \mathbf{c}'^{\mathsf{T}},\\ \widetilde{\mathbf{d}}^{\mathsf{T}} &= \mathbf{q}^{\mathsf{T}}\mathbf{QB} - (1, \mathbf{x}) \otimes \mathbf{q}^{\mathsf{T}}\mathbf{QG} - \mathbf{d}^{\mathsf{T}},\end{aligned}$$

and sends $\widetilde{\mathbf{c}}, \widetilde{\mathbf{d}}$ to $\mathcal{A}$.

5. $\mathcal{B}$ receives a bit from $\mathcal{A}$, and outputs the same bit.

By construction $\mathcal{B}$ is efficient, and if $\mathcal{B}$ doesn't abort in step 2, then $\mathcal{B}$ emulates $\mathsf{Exp}^1_{\mathrm{sALWE}}$ for $\mathcal{A}$ in $\mathsf{Exp}^1_{\mathrm{ALWE}}$. It remains to show that $\mathcal{B}$ doesn't abort in step 2 and that $\mathcal{B}$ emulates $\mathsf{Exp}^0_{\mathrm{sALWE}}$ in $\mathsf{Exp}^0_{\mathrm{ALWE}}$.

□

Parameter Settings. We rely on the hardness of adaptive LWE with subexponential modulus-to-noise ratio. For some $0 < \delta < \frac{1}{2}$, the adaptive LWE assumption is assumed to be hard when the dimension is $n = \mathrm{poly}(\lambda)$, the *prime* modulus is $q = O(2^{n^\delta})$, and the error distribution χ 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 $e_\lambda(g^a_{\lambda,1}, g^b_{\lambda,2}) = g^{ab}_{\lambda,\mathrm{T}}$ 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 λ, we denote $g^x_{\lambda,i}$ by $[\![x]\!]_i$ for $i \in \{1, 2, \mathrm{T}\}$. The notation extends to matrices, $[\![\mathbf{A}]\!]_i = g^{\mathbf{A}}_{\lambda,i}$, 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 $[\![\mathbf{X}]\!]_2[\![\mathbf{Y}]\!]_1 = [\![\mathbf{XY}]\!]_{\mathrm{T}}$.

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

Generic Asymmetric Pairing Group. We will prove the security of our ABE scheme 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. There are several different yet equivalent definitions of the model. We use a minimalist definition simplified from [Ps16], which suffices for security proofs.

Definition 8 (generic asymmetric pairing group [Ps16]). Let $p = \{p_\lambda\}_{\lambda \in \mathbb{N}}$ be a sequence of natural numbers. The *generic asymmetric pairing group* of order p consists of four stateful oracles. The state consists of three lists Vals_i ($i \in \{1, 2, \mathrm{T}\}$) storing the encoded values, which are initially singleton lists of 1 (representing the generators). The oracles are as follows:

- The encoding oracle $\mathsf{GrpEnc}_i(v)$, where $i \in \{1, 2, \mathrm{T}\}$, takes $v \in \mathbb{Z}_p$ as input. It appends v to Vals_i and outputs the current length of Vals_i as a handle for v.
- The zero-test oracle $\mathsf{GrpZT}(\gamma)$ takes as input a linear function γ. It evaluates
$$\gamma\left(\{\mathsf{Vals}_1[j_1]\mathsf{Vals}_2[j_2]\}_{j_1 \in [J_1], j_2 \in [J_2]}, \{\mathsf{Vals}_\mathrm{T}[j_\mathrm{T}]\}_{j_\mathrm{T} \in [j_\mathrm{T}]}\right),$$
where J_i is the current length of Vals_i for $i \in \{1, 2, \mathrm{T}\}$. The oracle outputs whether the evaluation yields 0 or not.

The security experiment can call all the four oracles, yet the adversary is only allowed to call GrpZT.

We remark two major differences between our formulation and that in [Ps16]. First, instead of random bit-strings, we use sequential indices as the handles, which can be used to position the coefficients of linear functions passed into GrpZT. Second, for zero-test queries, the function is linear over all values that can be computed in the target group, which simplifies the interface.

As noted in [Ps16], the definition does not explicitly allow algebraic operations over the encodings, yet this is without loss of generality. The security experiment provides the oracles and maintains their states, so it *knows* which value is encoded with each handle, and can operate over the values and call GrpEnc_i on the result. The adversary can replace all (allowed) algebraic operations by keeping track of the linear function over the values encoded by the security experiment, and zero-test the appropriate linear functions.

Brace Notation. For brevity, we write $\{\!\{v\}\!\}_i$ for a newly created handle encoding the value v in group G_i, so "returning $\{\!\{3\}\!\}_2$" means calling $h \leftarrow \mathsf{GrpEnc}_2(3)$ and returning h. This notation naturally extends to matrices, where a handle is created for each entry in the matrix. Since they represent handles and our definition does not explicitly allow algebraic operations over the encodings, we do not use operations with brace notations (unlike bracket notations).

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 9 (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:

- $\mathsf{Setup}(1^\lambda, 1^N)$ takes as input the security parameter 1^λ 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 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 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_T.

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

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

Definition 10 (key-succinct IPFE)**.** An IPFE scheme (Definition 9) is *(key-)succinct* if the length of isk is a fixed polynomial in λ, independent of N.

Security. Our basic security notion is selective simulation:

Definition 11 (selective simulation [Wee17, LL20])**.** A *simulator* for an IPFE scheme (Definition 9) consists of 3 efficient algorithms:

- $\widetilde{\mathsf{Setup}}(1^\lambda, 1^N)$ takes the same input as 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 $\mathsf{Exp}_\mathsf{real} \approx \mathsf{Exp}_\mathsf{sim}$, where $\mathsf{Exp}_\mathsf{real}(1^\lambda)$ or $\mathsf{Exp}_\mathsf{sim}(1^\lambda)$ 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

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

  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

$$\begin{aligned} &\text{in } \mathsf{Exp}_{\mathrm{real}}: && \mathsf{isk}_j \xleftarrow{\$} \mathsf{KeyGen}(\mathsf{imsk}, [\![\mathbf{v}_j]\!]_2); \\ &\text{in } \mathsf{Exp}_{\mathrm{sim}}: && \mathsf{isk}_j \xleftarrow{\$} \widetilde{\mathsf{KeyGen}}(\widetilde{\mathsf{imsk}}, [\![\mathbf{v}_j]\!]_2, \mathbf{u}^\top [\![\mathbf{v}_j]\!]_2); \end{aligned}$$

  and sends isk_j to $\mathcal{A}$.

- **Guess.** $\mathcal{A}$ outputs a bit b, which is the output of the experiment.

Lemma 5 ([ALS16,Wee17])**.** *Assuming the MDDH assumption (true in GGM), there exists a succinct selectively simulation-secure IPFE scheme. Its components have sizes*

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

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 [GPSW06] and a long line of follow-up works ([LOS+10,OT10,LW12,Att16, KW19] to name a few) used *linear* secret sharing schemes to construct ABE schemes in pairing groups. Its security notion is information-theoretic. The share size of such a scheme is equal to the smallest monotone span program [KW93] computing the policy. It is also known that functions computed by polynomial-sized span programs are in NC [Bei96,Ber84,BDHM92,KW93,Mul87].

The works of [AY20,AWY20] 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 [BGG+14, GV15,AY20,AWY20] 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 [AY20,AWY20] 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 [AY20,AWY20] 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. This separation helps proving adaptive security for our CP-ABE scheme, in which we need to simulate input encodings before the function is known. Our secret sharing schemes for both variants achieve succinct share sizes.

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

Definition 12 (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 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^λ, the input length 1^ℓ, and additional parameters param as input. It outputs some public parameter pp.
- $\mathsf{ShareX}(\mathsf{pp})$ takes the public parameter 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 pp, a Boolean function $f \in \mathcal{F}_{\lambda,\ell,\mathsf{param}}$, a secret $\mu \in \{0,1\}$, and the shared randomness r (output by 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 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 μ if $f(\mathbf{x}) = 0$.[5]

The scheme is required to be *correct*, i.e., for all $\lambda, \ell \in \mathbb{N}$, 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

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

Definition 13 (succinct shares)**.** A secret sharing scheme is *succinct* if the size of each share output by ShareX, ShareF is a fixed polynomial in λ, 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 12) and succinctness (Definition 13) are defined similarly to that of [AWY20], our linearity and security notions are different. Furthermore, our secret sharing schemes for KP-ABE and CP-ABE require slightly different linearity and security properties, so we introduce these definitions separately with their respective constructions.

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 [AWY20]. requirement on reconstruction. Our relaxed version (Definition 14) only stipulates it to be linear in $\mathbf{L}_f$ (and possibly non-linear in $L^{\mathbf{x}}$).

Definition 14 (weakly nearly linear reconstruction)**.** A secret sharing scheme (Definition 12) 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 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 γ 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

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

The second is an additional linearity requirement on ShareF.

Definition 15 (linear function sharing)**.** Let $\{p_\lambda\}_{\lambda\in\mathbb{N}}$ be a sequence of prime numbers. A secret sharing scheme (Definition 12) 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 FindCoef, we let Recon call FindCoef to obtain γ, B and output the unique $\mu \in \{0,1\}$ satisfying $\gamma(\mathbf{L}_f) - \mu\lfloor p/2 \rfloor \in [-B..B]$. The constructed Recon is efficient and correct. Since Recon is implied by FindCoef, we will only specify FindCoef and omit Recon when constructing (weakly) nearly linear secret sharing schemes.

Security. We consider a different security notion from [AWY20], called *non-annihilability*. Unlike [AWY20], which fixes the choice of policy f before Setup is run, we allow the adversary to adaptively choose f after seeing the public parameters 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 16 (non-annihilability for $\mathbf{L}_f$)**.** Let $\{p_\lambda\}_{\lambda\in\mathbb{N}}$ be a sequence of prime numbers. A secret sharing scheme (Definition 12) is *adaptively non-annihilable* for $\mathbf{L}_f$ if the output $\mathbf{L}_f$ of ShareF is a vector over $\mathbb{Z}_{p_\lambda}$ and all efficient adversary wins $\mathsf{Exp}_{\text{ANN-f}}$ with negligible probability, where in $\mathsf{Exp}^{\mathcal{A}}_{\text{ANN-f}}(1^\lambda)$, 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^ℓ and the additional parameter param. The challenger sets up the system by running $\mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda, 1^\ell, \mathsf{param})$, and sends 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 $\left(L_0, \{L_i^b\}_{i\in[\ell]}^{b\in\{0,1\}}, r\right) \xleftarrow{\$} \mathsf{ShareX}(\mathsf{pp})$ 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 γ. 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)* γ 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 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 [BGG+14, QWW18].

Construction 1 (secret sharing for circuits)**.** All variables x_λ are indexed by λ. For simplicity of notations, we suppress λ in subscripts. Let n be the LWE dimension, $p = 2^{\omega(\log \lambda)}$ a fixed prime modulus, (the LWE parameters will be chosen during 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 3):

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

where $d \le \frac{p^{\delta/4} - \log_2 p}{(1+\delta^{-1})\,\Theta(1)}$. Let $(\mathsf{EvalC}, \mathsf{EvalCX})$ be the algorithms in Lemma 2. The scheme works as follows:

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

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

  and picks χ 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 δ 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 $q/\overline{B} \ge (m+1)^{d+1}$ and $4((n+1)\overline{B}+3)+1 < p$.*

- $\mathsf{ShareX}(\mathsf{pp})$ takes the public parameter 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 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

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

  It outputs $\mathbf{L}_f$.

  Note: *The scheme indeed has linear function sharing (Definition 15) because* 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* ShareX *contains m elements in $\mathbb{Z}_q$. Note that m is a fixed polynomial in λ, d and is independent of the description size of f and the input length ℓ.*

- $\mathsf{FindCoef}(\mathsf{pp}, f, \mathbf{x}, \mathbf{L}^{\mathbf{x}})$ takes as input the public parameter 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 λ, d. We show that* FindCoef *is correct, i.e., if $f(\mathbf{x}) = 0$, then $4B + 1 \le p$ and $\gamma(\mathbf{L}_f) = \mu\lfloor p/2 \rceil + e$ for some $e \in [-B, B]$. First, by the choice of $n, \overline{B}$, we have*

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

Next, by construction we have

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

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

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

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

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

Finally, we have

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

By the definition of e_f, e_s, ε, and the setting of q, we have

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

We remark that by the choice of parameters, the above construction is a weakly nearly linear and succinct secret sharing scheme for the family of bounded-depth circuits (see Definition 3).

Efficiency. In the above construction, the public parameters 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 pp is $|\mathrm{pp}| = \mathrm{poly}(\lambda, d)\cdot \ell$. 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 and prove (in Section 4.2) non-annihilability security for $\mathbf{L}_f$ of the scheme.

Proposition 6. *Assuming the small-secret adaptive LWE assumption, Construction 1 is* non-annihilable for $\mathbf{L}_f$. *Specifically, for all efficient adversary $\mathcal{A}$ that wins the non-annihilability for $\mathbf{L}_f$ game with probability $\varepsilon_{\mathcal{A}}$, and for all $k \in \mathbb{N}$, there exists an efficient $\mathcal{B}$ that distinguishes the small-secret adaptive LWE experiments with probability $\varepsilon_{\mathrm{sALWE}}$, such that*

$$\varepsilon_{\mathcal{A}} \le (\lambda^{-k} + \varepsilon_{\mathrm{sALWE}} + (2B'+1)/p), \tag{6}$$

where $B' = (n+1)\overline{B} + 1$ is a polynomial.

Remark 1. In Construction 1, if we only assume plain (non-adaptive) LWE (which implies small-secret LWE [ACPS09]), then the resultant secret sharing scheme is selective non-annihilable for $\mathbf{L}_f$. The advantage relation in Proposition 6 holds, except $\varepsilon_{\mathcal{A}}$ is the advantage in the selective game and $\varepsilon_{\mathrm{sALWE}}$ is replaced by $\varepsilon_{\mathrm{sLWE}}$.

4.2 Proof of Proposition 6

We will need the following lemma in our proof:

Lemma 7 (leakage simulation [CCL18]). *Let $s, t, t' \in \mathbb{N}$ and $\varepsilon > 0$. For every distribution (X, Z) over $\{0,1\}^t \times \{0,1\}^{t'}$, there exists a simulator h of circuit size $\tilde{O}(2^{t'} s\varepsilon^{-2})$ such that for all circuit D of size at most s with binary output,*

$$\left|\Pr[(x,z) \xleftarrow{\$} (X,Z) : D(x,z) = 1] - \Pr[(x,z) \xleftarrow{\$} (X,Z) : D(x,h(x)) = 1]\right| \le \varepsilon.$$

Proof (Proposition 6). Fix an an efficient adversary $\mathcal{A}$ and constant $k \in \mathbb{N}$, we construct an adversary $\mathcal{B}$ that distinguishes the small-secret adaptive LWE experiments. We bound the advantage $\varepsilon_{\mathcal{A}}$ of $\mathcal{A}$ in the non-annihilability game as in (6) from the proposition statement.

Recall that in the security game, the adversary first sees the public parameter, and chooses some $\mathbf{x}$. The adversary next sees $L^{\mathbf{x}}$, and chooses f, μ such that $f(\mathbf{x}) = 1$. It finally, without seeing anything else, outputs some non-zero affine function γ. We denote components in the non-annihilability experiment by the following:

$$X_0 = r_{\mathcal{A}}, \quad X_1 = \mathrm{pp} = (\mathbf{a}, \mathbf{B}, \mathbf{Q}), \qquad X_2 = \left(\mathbf{L}_0, \mathbf{L}_1^{\mathbf{x}[1]}, \ldots, \mathbf{L}_\ell^{\mathbf{x}[\ell]}\right), \quad X_3 = \mathbf{s}^{\intercal}\mathbf{Q}\mathbf{a} + e,$$
$$X = (X_0, X_1, X_2, X_3), \qquad Z = -(\lfloor e_f \rceil_p + e_s + \varepsilon_1 + \varepsilon_2),$$
$$B' = (n+1)\overline{B} + 5.$$

Here, $r_{\mathcal{A}}$ is the randomness used by $\mathcal{A}$, e is an LWE noise sampled as $e \leftarrow \chi$, e_f is a shorthand defined as $e_f = (\mathbf{e}_0^\intercal, \mathbf{e}_1^\intercal, \ldots, \mathbf{e}_\ell^\intercal)\widehat{\mathbf{H}}_{f,\mathbf{x}}\mathbf{G}^{-1}(\mathbf{a})$, and $\varepsilon_1, \varepsilon_2, e_s$ are rounding errors defined as follows:

$$\begin{aligned}
&\lfloor \mathbf{s}^\intercal \mathbf{B}\mathbf{H}_f\mathbf{G}^{-1}(\mathbf{a}) - \mathbf{s}^\intercal\mathbf{Q}\mathbf{a} + e_f \rceil_p \\
&\;= \lfloor \mathbf{s}^\intercal \mathbf{B}\mathbf{H}_f\mathbf{G}^{-1}(\mathbf{a})\rceil_p - \lfloor \mathbf{s}^\intercal\mathbf{Q}\mathbf{a}\rceil_p + \lfloor e_f \rceil_p + \varepsilon_1, && \text{where } |\varepsilon_1| \le 4 \\
&\lfloor \mathbf{s}^\intercal\mathbf{Q}\mathbf{a} + e\rceil_p = \lfloor \mathbf{s}^\intercal\mathbf{Q}\mathbf{a}\rceil_p + \underbrace{\lfloor e \rceil_p}_{=0} + \varepsilon_2, && \text{where } |\varepsilon_2| \le 1 \\
&\lfloor \mathbf{s}^\intercal \mathbf{B}\mathbf{H}_f\mathbf{G}^{-1}(\mathbf{a})\rceil_p = \mathbf{s}^\intercal \lfloor \mathbf{B}\mathbf{H}_f\mathbf{G}^{-1}(\mathbf{a})\rceil_p + e_s, && \text{where } |e_s| \le n \cdot \|\mathbf{s}\|_\infty \le n\overline{B}.
\end{aligned}$$

We consider four hybrids, and let A_i denote the probabilities that $\mathcal{A}$ wins in H_i for $i \in [4]$.

- H_1 is the non-annihilability experiment.
- H_2 proceeds identically to H_1, except for two differences. First, if $Z > B'$, we immediately claim $\mathcal{A}$ to have lost. Second, we equivalently compute $\mathbf{L}_f$ as

  $$\mathbf{L}_f = \lfloor X_2 \widehat{\mathbf{H}}_{f,\mathbf{x}}\mathbf{G}^{-1}(\mathbf{a})\rceil_p + \lfloor X_3 \rceil_p + Z + \mu\lfloor p/2 \rceil.$$

  By the following claim, we have $A_1 = A_2$.

  Claim 8. $|Z| \le B'$ always holds in H_1, and H_1 is identical to H_2.

- H_3 proceeds identically to H_2, except that we simulate Z by $h(X)$, where h is an efficient algorithm guaranteed by the leakage simulation lemma.

  Claim 9. There exists some efficient algorithm h such that $|A_2 - A_3| \le \lambda^{-k}$ when h is used in H_3.

- H_4 proceeds identically to H_3, except that we replace X_2, X_3 by random. By the fact that they are LWE samples, we can construct from $\mathcal{A}$ an adversary $\mathcal{B}$ against the small-secret adaptive LWE assumption.

  Claim 10. There exists an efficient $\mathcal{B}$ that distinguishes the small-secret adaptive LWE experiments with advantage $\varepsilon_{\mathrm{sALWE}}$ such that $|A_3 - A_4| \le \varepsilon_{\mathrm{sALWE}}$.

  Next, using Schwartz–Zippel lemma (of degree-1) we show that $|A_4|$ is negligible.

  Claim 11. In H_4, the adversary $\mathcal{A}$ wins with negligible probability. Specifically, $|A_4| \le \frac{2B'+1}{p}$.

By a hybrid argument, we conclude that $\varepsilon_{\mathcal{A}} = A_1$ is bounded as in Equation (6) from the proposition statement. □

We now prove the claims.

Proof (Claim 8). First, we show that $|Z| \le B_{\mathcal{A}}$ always holds. By the definition of $e_f, e_s, \varepsilon_1, \varepsilon_2$, and the setting of q, we have

$$|Z| \le \left|\lfloor e_f \rceil_p\right| + |e_s| + |\varepsilon_1| + |\varepsilon_2| \le \left\lceil \frac{(m+1)^{(d+1)}}{q/p}\overline{B}\right\rceil + n\overline{B} + 5 \le B_{\mathcal{A}}.$$

Second, we show that $\mathbf{L}_f$ (in H_1) is indeed equal to $\lfloor X_2 \widehat{\mathbf{H}}_{f\mathbf{x}} \mathbf{G}^{-1}(\mathbf{a}) \rceil_p + \lfloor X_3 \rceil_p + Z + \mu \lfloor p/2 \rceil$ (in H_2). Expanding the term $\lfloor X_2 \widehat{\mathbf{H}}_{f\mathbf{x}} \mathbf{G}^{-1}(\mathbf{a}) \rceil_p$, we have

$$\begin{aligned}
\lfloor X_2 \widehat{\mathbf{H}}_{f,\mathbf{x}} \mathbf{G}^{-1}(\mathbf{a}) \rceil_p &= \lfloor \mathbf{L}^{\mathbf{x}} \widehat{\mathbf{H}}_{f,\mathbf{x}} \mathbf{G}^{-1}(\mathbf{a}) \rceil_p \\
&= \lfloor \underbrace{\left(\mathbf{s}^\intercal (\mathbf{B} + (1, \mathbf{x}) \otimes \mathbf{QG}) + (\mathbf{e}_0^\intercal, \mathbf{e}_1^\intercal, \ldots, \mathbf{e}_\ell^\intercal)\right)}_{\mathbf{L}^{\mathbf{x}}} \widehat{\mathbf{H}}_{f,\mathbf{x}} \mathbf{G}^{-1}(\mathbf{a}) \rceil_p \\
\text{(Lemma 2)} &= \lfloor \mathbf{s}^\intercal (\mathbf{BH}_f - \underbrace{f(\mathbf{x})}_{=1} \, \mathbf{QG}) \mathbf{G}^{-1}(\mathbf{a}) + \underbrace{(\mathbf{e}_0^\intercal, \mathbf{e}_1^\intercal, \ldots, \mathbf{e}_\ell^\intercal) \widehat{\mathbf{H}}_{f,\mathbf{x}} \mathbf{G}^{-1}(\mathbf{a})}_{=e_f} \rceil_p \\
&= \lfloor \mathbf{s}^\intercal \mathbf{BH}_f \mathbf{G}^{-1}(\mathbf{a}) - \mathbf{s}^\intercal \mathbf{Qa} + e_f \rceil_p \\
\text{(def. of } \varepsilon_1) &= \lfloor \mathbf{s}^\intercal \mathbf{BH}_f \mathbf{G}^{-1}(\mathbf{a}) \rceil_p - \lfloor \mathbf{s}^\intercal \mathbf{Qa} \rceil_P + \lfloor e_f \rceil_p + \varepsilon_1 \\
\text{(def. of } \varepsilon_2) &= \lfloor \mathbf{s}^\intercal \mathbf{BH}_f \mathbf{G}^{-1}(\mathbf{a}) \rceil_p - \lfloor \mathbf{s}^\intercal \mathbf{Qa} + e \rceil_P + \lfloor e_f \rceil_p + \varepsilon_1 + \varepsilon_2 \\
\text{(def. of } X_3, Z) &= \lfloor \mathbf{s}^\intercal \mathbf{BH}_f \mathbf{G}^{-1}(\mathbf{a}) \rceil_p - \lfloor X_3 \rceil_P - Z - e_s.
\end{aligned}$$

Using the above, we get

$$\begin{aligned}
& \lfloor X_2 \widehat{\mathbf{H}}_{f,\mathbf{x}} \mathbf{G}^{-1}(\mathbf{a}) \rceil_p + \lfloor X_3 \rceil_p + Z + \mu \lfloor p/2 \rceil \\
&= \lfloor \mathbf{s}^\intercal \mathbf{BH}_f \mathbf{G}^{-1}(\mathbf{a}) \rceil_p - e_s + \mu \lfloor p/2 \rceil \\
\text{(def. of } e_s) &= \mathbf{s}^\intercal \lfloor \mathbf{BH}_f \mathbf{G}^{-1}(\mathbf{a}) \rceil_p + \mu \lfloor p/2 \rceil = \mathbf{L}_f
\end{aligned}$$

This shows that H_1 and H_2 are identical. □

Proof (Claim 9). We need to show that for some efficient h, replacing Z by $h(X)$ only affects the winning probability by at most $\varepsilon = \lambda^{-k}$. Let (X, Z) be the components in the experiment as defined earlier. Note that by Claim 8, we have $|Z| \leq B'$ and we can let Z be a bit-string of length $\lceil \log_2(2B' + 1) \rceil$. We construct the (deterministic) algorithm $\mathcal{B}(X, Z)$ that (re-)performs the non-annihilability experiment:

1. $\mathcal{B}$ checks whether $|Z| \leq B'$. If not, it outputs 0 and terminates.
2. $\mathcal{B}$ re-runs the experiment with $X_0 = r_{\mathcal{A}}$ and $X_1 = \mathsf{pp} = (\mathbf{a}, \mathbf{B}, \mathbf{Q})$. It obtains $1^\ell, 1^d, f, \mu$, $\mathbf{x}$ as well as the non-zero affine function γ. from $\mathcal{A}$.
3. $\mathcal{B}$ checks whether $f(\mathbf{x}) = 1$. If this does not hold, it outputs 0 and terminates. It also checks whether γ is the zero function. If so, it outputs 0 and terminates.
4. $\mathcal{B}$ sets

$$\mathbf{L}_f = \lfloor X_2 \widehat{\mathbf{H}}_{f,\mathbf{x}} \mathbf{G}^{-1}(\mathbf{a}) \rceil_p + \lfloor X_3 \rceil_p + Z + \mu \lfloor p/2 \rceil,$$

   and checks whether $\gamma(\mathbf{L}_f) = 0$. It outputs 1 if this holds, and 0 otherwise.

Now let (X, Z) be jointly sampled by running the experiment on $\mathcal{A}$ and taking the corresponding components from the execution, then $\mathcal{B}(X, Z)$ outputs whether $\mathcal{A}$ won, and

$$\Pr[\mathcal{B}(X, Z) \to 1] = \Pr[\mathcal{A} \text{ wins}] = \varepsilon_{\mathcal{A}} = A_1 = A_2.$$

Since $\mathcal{A}$ has polynomial circuit size $s_{\mathcal{A}}$ and the experiment itself can be performed efficiently, the circuit size of $\mathcal{B}$ is also bounded by a polynomial $s_{\mathcal{B}}$. We apply the leakage simulation lemma (Lemma 7) to the joint distribution (X, Z) against distinguishers of size at most $s_{\mathcal{B}}$ with error $\varepsilon = \lambda^{-k}$. This gives us a simulator h with complexity $\tilde{O}(2^{\log_2(2B'+1)} s_{\mathcal{B}} \varepsilon^{-2}) = \tilde{O}(\lambda^k B' s_{\mathcal{B}})$, which is polynomial in λ.

Note that if we use h in H_3, the probability that $\mathcal{A}$ wins in H_3 is exactly $\Pr[\mathcal{B}(X, h(X))]$, and by the guarantee of leakage simulation lemma,

$$|A_2 - A_3| = \left|\Pr[\mathcal{B}(X,Z) \to 1] - \Pr[\mathcal{B}(X,h(X)) \to 1]\right| \leq \lambda^{-k}.$$

This finishes the proof of the claim. □

Proof (Claim 10). Recall that in both H_3 and H_4, the adversary $\mathcal{A}$ first chooses the input length ℓ and a polynomial upperbound d on circuit depths, and receives the public parameter $\mathsf{pp} = (\mathbf{a}, \mathbf{B}, \mathbf{Q})$ (LWE public matrices). $\mathcal{A}$ next chooses $\mathbf{x}$, and receives $X_2 = \mathbf{L}^{\mathbf{x}}$. The adversary $\mathcal{A}$ finally chooses f, μ, γ. The experiments test for the winning condition, where γ is evaluated on different distributions in H_3 and H_4.

We construct a small-secret adaptive LWE distinguisher $\mathcal{B}$ (with advantage $\varepsilon_{\mathrm{sALWE}}$) as follows:

1. $\mathcal{B}$ launches $\mathcal{A}(1^\lambda)$ with fresh randomness $r_{\mathcal{A}}$. It receives from $\mathcal{A}$ the input length 1^ℓ and circuit depth upperbound 1^d.

2. $\mathcal{B}$ sets $q = 2^{n^\delta}$ as described in Setup, and then sends $(1^\ell, 1^1)$ (i.e., $m' = 1$) to the small-secret adaptive LWE experiment with modulus q. It receives back $\mathbf{A}, \mathbf{B}, \mathbf{Q}$, which have shapes $n \times 1, n \times (\ell+1)m$ and $n \times n$. The distinguisher $\mathcal{B}$ sets $\mathbf{a} = \mathbf{A}$ and sends $\mathsf{pp} = (\mathbf{a}, \mathbf{B}, \mathbf{Q})$ to $\mathcal{A}$.

3. $\mathcal{B}$ waits for $\mathcal{A}$ to submit an attribute $\mathbf{x} \in \{0,1\}^\ell$. Upon receiving $\mathbf{x}$, it forwards $\mathbf{x}$ to the small-secret adaptive LWE experiment, and receives back $\mathbf{c} \in \mathbb{Z}_p^1$ and $\mathbf{d} \in \mathbb{Z}_p^{(\ell+1)m}$. It parses, sets and computes

$$X_0 = r_{\mathcal{A}}, \quad X_1 = \mathsf{pp}, \quad X_2 = (\mathbf{L}_0, \mathbf{L}_1^{\mathbf{x}[1]}, \ldots, \mathbf{L}_\ell^{\mathbf{x}[\ell]}) \leftarrow \mathbf{d}^\intercal, \quad X_3 = \mathbf{c},$$

   and sends $L^{\mathbf{x}}$ to $\mathcal{A}$.

4. $\mathcal{B}$ waits for $\mathcal{A}$ to submit a policy $f \in \mathsf{Ckt}_{\lambda,\ell,d}$, a bit $\mu \in \{0,1\}$, and an affine function γ.

5. If $f(\mathbf{x}) \neq 0$ or γ is the zero function, the distinguisher $\mathcal{B}$ aborts by outputting 0.

6. Otherwise, it computes and sets

$$Z \xleftarrow{\$} h(X), \ \widehat{\mathbf{H}}_{f,\mathbf{x}} \leftarrow \mathsf{EvalCX}(\mathbf{B}, \mathbf{Q}, f, \mathbf{x}),$$
$$\mathbf{L}_f \leftarrow \lfloor X_2 \widehat{\mathbf{H}}_{f,\mathbf{x}} \mathbf{G}^{-1}(\mathbf{a}) \rceil_p + \lfloor X_3 \rceil_p + Z + \mu \lfloor p/2 \rfloor.$$

   If $|Z| > B_{\mathcal{A}}$, the distinguisher $\mathcal{B}$ outputs 0 and terminates.

7. Otherwise, it proceeds to evaluate $\gamma(\mathbf{L}_f)$. It outputs 1 if the result is zero, and 0 otherwise.

Clearly $\mathcal{B}$ is efficient. In $\mathsf{Exp}^0_{\text{ALWE}}$, the distinguisher $\mathcal{B}$ emulates H_3 for $\mathcal{A}$ and outputs whether $\mathcal{A}$ won. In $\mathsf{Exp}^1_{\text{ALWE}}$, it does so for H_4. Therefore,

$$|A_3 - A_4| = \left|\Pr[\mathcal{B} \to 1 \text{ in } \mathsf{Exp}^0_{\text{ALWE}}] - \Pr[\mathcal{B} \to 1 \text{ in } \mathsf{Exp}^1_{\text{ALWE}}]\right| \le \varepsilon_{\text{sALWE}}.$$

□

Proof (Claim 11). In H_4, we need to show that the non-zero affine function γ annihilates $\mathbf{L}_f$ with negligible probability. Recall the components in H_4:

$$\begin{aligned} &X_0 = r_{\mathcal{A}}, \quad X_1 = \mathsf{pp}, && X_2 = (\mathbf{L}_0, \mathbf{L}_1^{\mathbf{x}[1]}, \dots, \mathbf{L}_\ell^{\mathbf{x}[\ell]}) = \mathbf{d}^\intercal, \quad X_3 = \mathbf{c}, \\ &Z \xleftarrow{\$} h(X), && \mathbf{L}_f \leftarrow \boxed{\lfloor X_2 \widehat{\mathbf{H}}_{f,\mathbf{x}} \mathbf{G}^{-1}(\mathbf{a}) \rceil_p} + Z \boxed{+ \lfloor X_3 \rceil_p + \mu \lfloor p/2 \rfloor} = X_3' - Z, \end{aligned}$$

where X_3' collects the boxed terms. Note that by definition, μ and γ are determined by X_0, X_1, X_2. In H_4, the values in X_3 are uniformly random and independent of X_0, X_1, X_2 (thus μ, γ). This implies that X_3' is also uniformly random and independent of γ, so $\mathbf{L}_f = X_3' - Z$ is a uniformly random value perturbed by $Z = h(X)$.

The following probabilities are taken in H_4 and are implicitly intersected with the requirement that $f(\mathbf{x}) = 0$, γ is not the zero function, and $|Z| \le B'$. We apply union bound over all possible values in $[-B'..B']$ to obtain

$$\begin{aligned} A_4 &= \Pr\left[\gamma(\mathbf{L}_f) = 0\right] \\ &= \Pr\left[\gamma(X_3' - Z) = 0\right] \\ &\le \Pr\left[\exists z \in [-B'..B'] \text{ s.t. } \gamma(X_3' - z) = 0\right] \\ &\le \sum_{z=-B'}^{B'} \Pr\left[\gamma(X_3' - z) = 0\right] \le \frac{2B'+1}{p}, \end{aligned}$$

where the third inequality follows from Schwartz–Zippel lemma for degree 1. Since B' is polynomial and p is super-polynomial, A_4 is negligible. □

4.3 Secret Sharing for Boolean Formulae from Adaptive LWE

In our CP-ABE constructions, we need a secret sharing scheme with the following linearity property, defined similarly to that of [AWY20].

Definition 17 (nearly linear reconstruction)**.** A secret sharing scheme (Definition 12) is *nearly linear* if it satisfies the following requirements:

- Let $\{p_\lambda\}_{\lambda \in \mathbb{N}}$ be a sequence of prime numbers. $\mathbf{L}_0 = L_0, \{\mathbf{L}_i^b = L_i^b\}_{i \in [\ell]}^{b \in \{0,1\}}, \mathbf{L}_f = L_f$ are vectors over $\mathbb{Z}_{p_\lambda}$.
- There is an efficient coefficient-finding algorithm $\mathsf{FindCoef}(\mathsf{pp}, f, \mathbf{x})$, taking as input the public parameter pp, a Boolean function $f \in \mathcal{F}_{\lambda,\ell}$, and an input $\mathbf{x} \in \{0,1\}^\ell$ to f. It outputs an affine function γ and a noise bound 1^B. For all $\lambda, \ell \in \mathbb{N}$, 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

$$\Pr\left[\begin{array}{r} \mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda, 1^\ell, \mathsf{param}) \\ (\mathbf{L}_0, \{\mathbf{L}_i^b\}_{i \in [\ell]}^{b \in \{0,1\}}, r) \xleftarrow{\$} \mathsf{ShareX}(\mathsf{pp}) \\ \mathbf{L}_f \xleftarrow{\$} \mathsf{ShareF}(\mathsf{pp}, f, \mu, r) \\ (\gamma, 1^B) \xleftarrow{\$} \mathsf{FindCoef}(\mathsf{pp}, f, \mathbf{x}) \end{array} : \begin{array}{l} 4B + 1 < p_\lambda \text{ and} \\ \exists e \in [-B..B] \text{ s.t.} \\ \gamma(\mathbf{L}_f, \mathbf{L}^{\mathbf{x}}) = \mu \lfloor p/2 \rceil + e \end{array}\right] = 1.$$

Security. In addition to the non-annihilability for $\mathbf{L}_f$ security (Definition 16), our CP-ABE constructions need a secret sharing scheme with the following property:

Definition 18 (non-annihilability for $\mathbf{L}^{\mathbf{x}}$)**.** Let $\{p_\lambda\}_{\lambda\in\mathbb{N}}$ be a sequence of prime numbers. A secret sharing scheme (Definition 12) is *adaptively non-annihilable* for $\mathbf{L}^{\mathbf{x}}$ if the output $\mathbf{L}_0, \{\mathbf{L}_i^b\}_i^b$ of ShareX are vectors over $\mathbb{Z}_{p_\lambda}$ and all efficient adversary wins $\mathsf{Exp}_{\text{ANN-x}}$ only negligible probability, where in $\mathsf{Exp}^{\mathcal{A}}_{\text{ANN-x}}(1^\lambda)$, 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^ℓ and the additional parameter param. The challenger sets up the system by running $\mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda, 1^\ell, \mathsf{param})$, and sends pp to $\mathcal{A}$.
- **Challenge.** $\mathcal{A}$ submits an input $\mathbf{x} \in \{0,1\}^\ell$ and an affine function γ. Upon receiving them, the challenger runs $\big(\mathbf{L}_0, \{\mathbf{L}_i^b\}_{i\in[\ell]}^{b\in\{0,1\}}, r\big) \xleftarrow{\$} \mathsf{ShareX}(\mathsf{pp})$ and determines the outcome of the experiment. $\mathcal{A}$ *wins* if *i)* γ is not the zero function; and *ii)* $\gamma(\mathbf{L}^{\mathbf{x}}) = 0$. Otherwise, $\mathcal{A}$ loses.

A secret sharing scheme is *selectively non-annihilable* for $\mathbf{L}^{\mathbf{x}}$ if the above conditions hold except that $\mathcal{A}$ must choose $\mathbf{x}$ before receiving pp.

We now construct a succinct secret sharing scheme, satisfying the above linearity and security definitions, for 5-PBP from adaptive LWE. Our scheme is based on the attribute-based laconic function evaluation scheme in [QWW18] and we employ the techniques in [BGG+14,GV15] to control LWE noises.

Construction 2 (secret sharing for branching programs)**.** The construction is described for a fixed value of λ, and λ is suppressed for brevity. Let n be the LWE dimension, p the LWE prime modulus, $m = n\lceil\log_2 p\rceil$ and χ the error distribution that is $\overline{B}$-bounded. We set $n, \overline{B} = \lambda^{\Theta(1)}$ and $p = 2^{\omega(\log\lambda)}$. We construct a nearly linear and succinct secret sharing scheme for the family of length-5 permutation branching programs (see Definition 4):

$$\mathsf{5PBP}_\ell = \{\, \text{5-PBP } f : \{0,1\}^\ell \to \{0,1\} \text{ of length at most } s \,\},$$

where $s = (p - 4m\overline{B} - 4\overline{B} - 2)/12m^2\overline{B} = 2^{\omega(\log\lambda)}$. Let $(\mathsf{EvalF}, \mathsf{EvalFX})$ be the algorithms in Lemma 3. The scheme works as follows:

- $\mathsf{Setup}(1^\ell)$ takes the input length as input. It samples and sets

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

  and outputs $\mathsf{pp} = (\mathbf{a}, \mathbf{B})$.

- $\mathsf{ShareX}(\mathsf{pp})$ takes as input the public parameter pp. It samples and sets

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

  and outputs $(\mathbf{L}_0, \{\mathbf{L}_i^b\}_{i\in[\ell]}^{b\in\{0,1\}}, \mathbf{s})$.

  Note: *We remark that an input* $\mathbf{x}$ *could be thought as being prepended with* $\mathbf{x}[0] = 1$ *(see also Lemma 3), and* $\mathbf{L}_0$ *encodes this bit of input under* $\mathbf{A}_0$*. This extra bit enables operating with constants in* f*.*

- $\mathsf{ShareF}(\mathsf{pp}, f, \mu, \mathbf{s})$ takes as input the public parameter pp, some $f \in \mathsf{5PBP}_\ell$, a secret bit $\mu \in \{0,1\}$, and the shared randomness $\mathbf{s}$. It runs $\mathbf{H}_f \leftarrow \mathsf{EvalF}(\mathbf{B}, f)$, and samples and sets

  $$\bar{e} \xleftarrow{\$} \chi, \quad \mathbf{L}_f = \mathbf{s}^\top \mathbf{B}\mathbf{H}_f \mathbf{G}^{-1}(\mathbf{a}) + \bar{e} + \mu\lfloor p/2 \rceil$$

  where $\mathbf{G}$ is the gadget matrix. The algorithm outputs $\mathbf{L}_f$.

  Note: *The scheme is succinct as $\mathbf{L}_f$ contains 1 element in $\mathbb{Z}_p$, and each share output by* ShareX *contains m elements in $\mathbb{Z}_p$. Since, m is a fixed polynomial in λ and is independent of the description size of f and the input length ℓ.*

- $\mathsf{FindCoef}(\mathsf{pp}, f, \mathbf{x})$ takes as input the public parameter pp, some $\mathbf{x} \in \{0,1\}^\ell$, and some $f \in \mathsf{5PBP}_\ell$. If $f(\mathbf{x}) = 1$, it outputs $\perp$ and terminates. Otherwise, it runs $\widehat{\mathbf{H}}_{f,\mathbf{x}} \leftarrow \mathsf{EvalFX}(\mathbf{B}, f, \mathbf{x})$ and defines

  $$\gamma(\mathbf{L}_f, \mathbf{L}^\mathbf{x}) = \mathbf{L}_f - \mathbf{L}^\mathbf{x}\widehat{\mathbf{H}}_{f,\mathbf{x}}\mathbf{G}^{-1}(\mathbf{a}),$$
  $$B = (3m\ell_{\mathrm{BP}} + 1)m\overline{B} + \overline{B},$$

  where ℓ_{BP} is the length of f. The algorithm outputs $(\gamma, 1^B)$.

  Note: *The procedure is indeed efficient since B is polynomial in λ and ℓ_{BP} (the length of f). We show that* FindCoef *is correct. By the definition of* $\mathsf{5PBP}_\ell$*, we have $\ell_{\mathrm{BP}} \le s$ and thus*

  $$4B + 1 \le 4\big((3ms + 1)m\overline{B} + \overline{B}\big) + 1 = p - 1 < p.$$

  By the property of EvalF, EvalFX *(Lemma 3) and the definition of $\mathbf{L}_f$ and $\mathbf{L}^\mathbf{x}$,*

  $$\begin{aligned}&\mathbf{L}_f - \mathbf{L}^\mathbf{x}\widehat{\mathbf{H}}_{f,\mathbf{x}}\mathbf{G}^{-1}(\mathbf{a})\\ &= \mathbf{s}^\top\mathbf{B}\mathbf{H}_f\mathbf{G}^{-1}(\mathbf{a}) \boxed{+\bar{e}} + \mu\lfloor p/2\rceil\\ &\quad - \mathbf{s}^\top\big(\mathbf{B} - (1,\mathbf{x})\otimes\mathbf{G}\big)\widehat{\mathbf{H}}_{f,\mathbf{x}}\mathbf{G}^{-1}(\mathbf{a}) \boxed{-(\mathbf{e}_0^\top, \mathbf{e}_1^\top, \ldots, \mathbf{e}_\ell^\top)\widehat{\mathbf{H}}_{f,\mathbf{x}}\mathbf{G}^{-1}(\mathbf{a})}\\ &= \mathbf{s}^\top\mathbf{B}\mathbf{H}_f\mathbf{G}^{-1}(\mathbf{a}) - \mathbf{s}^\top(\mathbf{B}\mathbf{H}_f - \underbrace{f(\mathbf{x})}_{=0}\mathbf{G})\mathbf{G}^{-1}(\mathbf{a}) + \mu\lfloor p/2\rceil + e' = \mu\lceil p/2\rceil + e',\end{aligned}$$

  where e' collects the boxed terms. Since $\mathbf{G}^{-1}(\mathbf{a}) \in \{0,1\}^m$, we have

  $$|e'| \le |\bar{e}| + m\cdot\|\widehat{\mathbf{H}}_{f,\mathbf{x}}^\top\|_\infty \cdot \|(\mathbf{e}_0^\top, \mathbf{e}_1^\top, \ldots, \mathbf{e}_\ell^\top)^\top\|_\infty \le \overline{B} + m(3m\ell_{\mathrm{BP}} + 1)\overline{B} = B,$$

  where ℓ_{BP} is the length of f.

We remark that by the choice of parameters, the above construction is a nearly linear and succinct secret sharing scheme for the family of length-5 permutation branching programs (Definition 4).

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

We next state and prove (in Section 4.4) non-annihilability security for both $\mathbf{L}^\mathbf{x}$ and $\mathbf{L}_f$ of the scheme.

Proposition 12. *Assuming the adaptive LWE assumption, Construction 2 is* non-annihilable *for* $\mathbf{L}^{\mathbf{x}}$*. Specifically, for all efficient adversary* $\mathcal{A}$ *that wins the non-annihilability for* $\mathbf{L}^{\mathbf{x}}$ *game with probability* $\varepsilon_{\mathcal{A}}$*, there exists an efficient* $\mathcal{B}$ *that distinguishes the adaptive LWE experiments with probability* $\varepsilon_{\mathrm{ALWE}}$*, such that*

$$\varepsilon_{\mathcal{A}} \le \varepsilon_{\mathrm{ALWE}}.$$

Proposition 13. *Assuming the adaptive LWE assumption, Construction 2 is* non-annihilable *for* $\mathbf{L}_f$*. Specifically, for all efficient adversary* $\mathcal{A}$ *that wins the non-annihilability for* $\mathbf{L}_f$ *game with probability* $\varepsilon_{\mathcal{A}}$*, and for all* $k \in \mathbb{N}$*, there exists an efficient* $\mathcal{B}$ *that distinguishes the adaptive LWE experiments with probability* $\varepsilon_{\mathrm{ALWE}}$*, such that*

$$\varepsilon_{\mathcal{A}} \le (\lambda^{-k} + \varepsilon_{\mathrm{ALWE}} + \frac{2B_{\mathcal{A}}+1}{p}), \tag{7}$$

where $B_{\mathcal{A}} = (3m|\mathcal{A}|+1)m\overline{B}$ *is a polynomial depending on the running time of* $\mathcal{A}$*.*

Remark 2. In Construction 2, if we only assume plain (non-adaptive) LWE, then the resultant secret sharing scheme achieves selective non-annihilability for $\mathbf{L}^{\mathbf{x}}$ and $\mathbf{L}_f$. The proofs of the corresponding versions of Propositions 12 and 13 are different and program $\mathbf{x}$ into the public matrix $\mathbf{B}$, as done in [BGG⁺14,GV15].

4.4 Proof of Proposition 13

Note that Proposition 12 follows directly from the adaptive LWE assumption. We focus on showing Proposition 13.

Proof (Proposition 13). The proof is similar to that of Proposition 6, except we will apply the ALWE assumption, instead of the sALWE assumption.

Fix an an efficient adversary $\mathcal{A}$, (with circuit size $s_{\mathcal{A}}$), and constant $k \in \mathbb{N}$, we construct an adversary $\mathcal{B}$ that distinguishes the adaptive LWE experiments. We bound the advantage $\varepsilon_{\mathcal{A}}$ of $\mathcal{A}$ in the non-annihilability game as in Equation (7) from the proposition statement.

Recall that in the security game, the adversary first sees the public parameter, and chooses some $\mathbf{x}$. The adversary next sees $L^{\mathbf{x}}$, and chooses f, μ such that $f(\mathbf{x}) = 1$. It finally, without seeing anything else, outputs some non-zero affine function γ. We denote components in the non-annihilability experiment by the following:

$$X_0 = r_{\mathcal{A}}, \quad X_1 = \mathsf{pp} = (\mathbf{a}, \mathbf{B}), \qquad X_2 = \left(\mathbf{L}_0, \mathbf{L}_1^{\mathbf{x}[1]}, \ldots, \mathbf{L}_\ell^{\mathbf{x}[\ell]}\right), \quad X_3 = \mathbf{s}^\intercal \mathbf{a} + \overline{e},$$
$$X = (X_0, X_1, X_2, X_3), \qquad Z = (\mathbf{e}_0^\intercal, \mathbf{e}_1^\intercal, \ldots, \mathbf{e}_\ell^\intercal)\widehat{\mathbf{H}}_{f,\mathbf{x}}\mathbf{G}^{-1}(\mathbf{a}),$$
$$B_{\mathcal{A}} = (3ms_{\mathcal{A}}+1)m\overline{B}.$$

Here, $r_{\mathcal{A}}$ is the randomness used by $\mathcal{A}$. As for $B_{\mathcal{A}}$, we will later show that it is an upper bound of $|Z|$. We consider four hybrids, and let A_i denote the probabilities that $\mathcal{A}$ wins in H_i for $i \in [4]$.

- H_1 is the non-annihilability experiment.
- H_2 proceeds identically to H_1, except for two differences. First, if $Z > B_{\mathcal{A}}$, we immediately claim $\mathcal{A}$ to have lost. Second, when performing zero-tests, we replace $\mathbf{L}_f$ by (equivalently computed)

$$X_2\widehat{\mathbf{H}}_{f,\mathbf{x}}\mathbf{G}^{-1}(\mathbf{a}) - Z + X_3 + \mu\lfloor p/2\rceil.$$

  By the following claim, we have $A_1 = A_2$.

Claim 14. *$|Z| \le B_{\mathcal{A}}$ always holds in H_1, and H_1 is identical to H_2.*

- H_3 proceeds identically to H_2, except that we simulate Z (used to compute $\mathbf{L}_f$, which in turn is used in performing zero-tests) by $h(X)$, where h is an efficient algorithm guaranteed by the leakage simulation lemma.

  Claim 15. *There exists some efficient algorithm h such that $|A_2 - A_3| \le \lambda^{-k}$ when h is used in H_3.*

- H_4 proceeds identically to H_3, except that we replace X_2, X_3 by random. By the fact that they are LWE samples, we can construct from $\mathcal{A}$ an adversary $\mathcal{B}$ against the small-secret adaptive LWE assumption.

  Claim 16. *There exists an efficient $\mathcal{B}$ that distinguishes the adaptive LWE experiments with advantage $\varepsilon_{\mathrm{ALWE}}$ such that $|A_3 - A_4| \le \varepsilon_{\mathrm{ALWE}}$.*

  Next, using Schwartz–Zippel lemma (of degree-1) we show that $|A_4|$ is negligible.

  Claim 17. *In H_4, the adversary $\mathcal{A}$ wins with negligible probability. Specifically, $|A_4| \le \frac{2B_{\mathcal{A}}+1}{p}$.*

By a hybrid argument, we conclude that $\varepsilon_{\mathcal{A}} = A_1$ is bounded as in Equation (7) from the proposition statement. □

Proof (Claim 14). We need to show that $|Z| \le B_{\mathcal{A}}$ always holds in H_1, and that the expression used to compute $\mathbf{L}_f$ in H_2 is always equal to the value in H_1. They follow by the property of $\mathsf{EvalF}, \mathsf{EvalFX}$ (Lemma 3).

Let ℓ_{BP} be the length of f, and $s_{\mathcal{A}}$ be the circuit size of $\mathcal{A}$, then $\ell_{\mathrm{BP}} \le s_{\mathcal{A}}$ since f is output by $\mathcal{A}$. We have

$$\|\widehat{\mathbf{H}}_{f,\mathbf{x}}\|_\infty \le 3m\ell_{\mathrm{BP}} + 1 \le 3ms_{\mathcal{A}} + 1.$$

Since $\mathbf{G}^{-1}(\mathbf{a}) \in \{0,1\}^m$ and $\|\mathbf{e}_i\|_\infty \le \overline{B}$ for $i = 0, 1, \ldots, \ell$, it follows that

$$|Z| = |(\mathbf{e}_0^\intercal, \mathbf{e}_1^\intercal, \ldots, \mathbf{e}_\ell^\intercal)\widehat{\mathbf{H}}_{f,\mathbf{x}}\mathbf{G}^{-1}(\mathbf{a})| \le m \cdot \|\widehat{\mathbf{H}}_{f,\mathbf{x}}^\intercal\|_\infty \|(\mathbf{e}_0^\intercal, \mathbf{e}_1^\intercal, \ldots, \mathbf{e}_\ell^\intercal)^\intercal\|_\infty \le m \cdot (3ms_{\mathcal{A}} + 1)\overline{B} = B_{\mathcal{A}}.$$

For the second half of the claim, when $f(\mathbf{x}) = 1$, we can rewrite $\mathbf{L}_f$ as follows:

$$\begin{aligned}
\mathbf{L}_f &= \mathbf{s}^\intercal \mathbf{B}\mathbf{H}_f \mathbf{G}^{-1}(\mathbf{a}) + \overline{e} + \mu\lfloor p/2 \rceil \\
\text{(Lemma 3)} \quad &= \mathbf{s}^\intercal \Big((\mathbf{B} - (1,\mathbf{x}) \otimes \mathbf{G})\widehat{\mathbf{H}}_{f,\mathbf{x}} + f(\mathbf{x})\mathbf{G} \Big) \mathbf{G}^{-1}(\mathbf{a}) + \overline{e} + \mu\lfloor p/2 \rceil, \\
&= \mathbf{s}^\intercal (\mathbf{B} - (1,\mathbf{x}) \otimes \mathbf{G})\widehat{\mathbf{H}}_{f,\mathbf{x}}\mathbf{G}^{-1}(\mathbf{a}) + \mathbf{s}^\intercal \mathbf{a} + \overline{e} + \mu\lfloor p/2 \rceil, \\
&= \big(\mathbf{s}^\intercal (\mathbf{B} - (1,\mathbf{x}) \otimes \mathbf{G}) \underline{+ (\mathbf{e}_0^\intercal, \mathbf{e}_1^\intercal, \ldots, \mathbf{e}_\ell^\intercal)}\big)\widehat{\mathbf{H}}_{f,\mathbf{x}}\mathbf{G}^{-1}(\mathbf{a}) \\
&\quad \underline{- (\mathbf{e}_0^\intercal, \mathbf{e}_1^\intercal, \ldots, \mathbf{e}_\ell^\intercal)\widehat{\mathbf{H}}_{f,\mathbf{x}}\mathbf{G}^{-1}}(\mathbf{a}) + (\mathbf{s}^\intercal \mathbf{a} + \overline{e}) + \mu\lfloor p/2 \rceil \\
&= X_2 \widehat{\mathbf{H}}_{f,\mathbf{x}}\mathbf{G}^{-1}(\mathbf{a}) - Z + X_3 + \mu\lfloor p/2 \rceil.
\end{aligned}$$

This shows that H_1 and H_2 are identical. □

Proof (Claim 15). We need to show that for some efficient h, replacing Z by $h(X)$ only affects the winning probability by at most $\varepsilon = \lambda^{-k}$. Let (X, Z) be the components in the experiment as defined earlier. Note that by Claim 14, we have $|Z| \le B_{\mathcal{A}}$ and we can let Z be a bit-string of length $\lceil \log_2(2B_{\mathcal{A}} + 1) \rceil$. We construct the (deterministic) algorithm $\mathcal{B}(X, Z)$ that (re-)performs the non-annihilability experiment:

1. $\mathcal{B}$ checks whether $|Z| \le B_{\mathcal{A}}$. If not, it outputs 0 and terminates.

2. $\mathcal{B}$ re-runs the experiment with $X_0 = r_{\mathcal{A}}$ and $X_1 = \mathsf{pp} = (\mathbf{a}, \mathbf{B})$. It obtains $1^\ell, f, \mu, \mathbf{x}$ as well as the non-zero affine function γ. from $\mathcal{A}$.

3. $\mathcal{B}$ checks whether $f(\mathbf{x}) = 1$. If this does not hold, it outputs 0 and terminates. It also checks whether γ is the zero function. If so, it outputs 0 and terminates.

4. $\mathcal{B}$ sets

$$\mathbf{L}_f = X_2 \widehat{\mathbf{H}}_{f,\mathbf{x}} \mathbf{G}^{-1}(\mathbf{a}) - Z + X_3 + \mu \lfloor p/2 \rceil,$$

   and checks whether $\gamma(\mathbf{L}_f) = 0$. It outputs 1 if this holds, and 0 otherwise.

Now let (X, Z) be jointly sampled by running the experiment on $\mathcal{A}$ and taking the corresponding components from the execution, then $\mathcal{B}(X, Z)$ outputs whether $\mathcal{A}$ won, and

$$\Pr[\mathcal{B}(X, Z) \to 1] = \Pr[\mathcal{A} \text{ wins}] = \varepsilon_{\mathcal{A}} = A_1 = A_2.$$

Since $\mathcal{A}$ has polynomial circuit size $s_{\mathcal{A}}$ and the experiment itself can be performed efficiently, the circuit size of $\mathcal{B}$ is also bounded by a polynomial $s_{\mathcal{B}}$. We apply the leakage simulation lemma (Lemma 7) to the joint distribution (X, Z) against distinguishers of size at most $s_{\mathcal{B}}$ with error $\varepsilon = \lambda^{-k}$. This gives us a simulator h with complexity $\tilde{\mathrm{O}}(2^{|Z|} s_{\mathcal{B}} \varepsilon^{-2}) = \tilde{\mathrm{O}}(\lambda^k B_{\mathcal{A}} s_{\mathcal{B}})$, which is polynomial in λ.

Note that if we use h in H_3, the probability that $\mathcal{A}$ wins in H_3 is exactly $\Pr[\mathcal{B}(X, h(X))]$, and by the guarantee of leakage simulation lemma,

$$|A_2 - A_3| = \big|\Pr[\mathcal{B}(X, Z) \to 1] - \Pr[\mathcal{B}(X, h(X)) \to 1]\big| \le \lambda^{-k}.$$

This finishes the proof of the claim. □

Proof (Claim 16). Recall that in both H_3 and H_4, the adversary $\mathcal{A}$ first chooses the input length ℓ, and receives the public parameter $\mathsf{pp} = (\mathbf{a}, \mathbf{B})$ (LWE public matrices). $\mathcal{A}$ next chooses $\mathbf{x}$, and receives $\mathbf{L}^{\mathbf{x}}$. The adversary $\mathcal{A}$ finally chooses f, μ, γ. The experiments test for the winning condition, where γ is evaluated on different distributions in H_3 and H_4.

We construct an adaptive LWE distinguisher $\mathcal{B}$ (with advantage $\varepsilon_{\mathrm{ALWE}}$) as follows:

1. $\mathcal{B}$ launches $\mathcal{A}(1^\lambda)$ with fresh randomness $r_{\mathcal{A}}$. It receives from $\mathcal{A}$ the input length 1^ℓ.

2. $\mathcal{B}$ sends $(1^\ell, 1^1)$ to the adaptive LWE experiment (i.e., $m' = 1$). It receives back $\mathbf{A}, \mathbf{B}$, which have shapes $n \times 1$ and $n \times (\ell + 1)m$. The distinguisher $\mathcal{B}$ sets $\mathbf{a} = \mathbf{A}$ and sends $\mathsf{pp} = (\mathbf{a}, \mathbf{B})$ to $\mathcal{A}$.

3. $\mathcal{B}$ waits for $\mathcal{A}$ to submit an attribute $\mathbf{x} \in \{0,1\}^\ell$. Upon receiving $\mathbf{x}$, it forwards $\mathbf{x}$ to the adaptive LWE experiment, and receives back $\mathbf{c} \in \mathbb{Z}_p^1$ and $\mathbf{d} \in \mathbb{Z}_p^{(\ell+1)m}$. It parses, sets and computes
$$X_0 = r_{\mathcal{A}}, \quad X_1 = \mathsf{pp}, \quad X_2 = (\mathbf{L}_0, \mathbf{L}_1^{\mathbf{x}[1]}, \dots, \mathbf{L}_\ell^{\mathbf{x}[\ell]}) \leftarrow \mathbf{d}^\intercal, \quad X_3 = \mathbf{c},$$
and sends $L^{\mathbf{x}}$ to $\mathcal{A}$.
4. $\mathcal{B}$ waits for $\mathcal{A}$ to submit a policy $f \in \mathsf{5PBP}_{\lambda,\ell}$, a bit $\mu \in \{0,1\}$, and an affine function γ.
5. If $f(\mathbf{x}) \neq 0$ or γ is the zero function, the distinguisher $\mathcal{B}$ aborts by outputting 0.
6. Otherwise, it computes and sets
$$Z \xleftarrow{\$} h(X), \ \widehat{\mathbf{H}}_{f,\mathbf{x}} \leftarrow \mathsf{EvalFX}(\mathbf{B}, f, \mathbf{x}), \ \mathbf{L}_f \leftarrow X_2 \widehat{\mathbf{H}}_{f,\mathbf{x}} \mathbf{G}^{-1}(\mathbf{a}) - Z + X_3 + \mu \lfloor p/2 \rfloor.$$
If $|Z| > B_{\mathcal{A}}$, the distinguisher $\mathcal{B}$ outputs 0 and terminates.
7. Otherwise, it proceeds to evaluate $\gamma(\mathbf{L}_f)$. It outputs 1 if the result is zero, and 0 otherwise.

Clearly $\mathcal{B}$ is efficient. In $\mathsf{Exp}^0_{\mathrm{ALWE}}$, the distinguisher $\mathcal{B}$ emulates H_3 for $\mathcal{A}$ and outputs whether $\mathcal{A}$ won. In $\mathsf{Exp}^1_{\mathrm{ALWE}}$, it does so for H_4. Therefore,
$$|A_3 - A_4| = \left|\Pr[\mathcal{B} \to 1 \text{ in } \mathsf{Exp}^0_{\mathrm{ALWE}}] - \Pr[\mathcal{B} \to 1 \text{ in } \mathsf{Exp}^1_{\mathrm{ALWE}}]\right| \leq \varepsilon_{\mathrm{ALWE}}.$$
□

Proof (Claim 17). In H_4, we need to show that the non-zero affine function γ annihilates $\mathbf{L}_f$ with negligible probability. Recall the components in H_4:
$$X_0 = r_{\mathcal{A}}, \quad X_1 = \mathsf{pp}, \qquad X_2 = (\mathbf{L}_0, \mathbf{L}_1^{\mathbf{x}[1]}, \dots, \mathbf{L}_\ell^{\mathbf{x}[\ell]}) \leftarrow \mathbf{d}^\intercal, \quad X_3 = \bar{\mathbf{c}},$$
$$Z \xleftarrow{\$} h(X), \qquad \mathbf{L}_f \leftarrow \boxed{X_2 \widehat{\mathbf{H}}_{f,\mathbf{x}} \mathbf{G}^{-1}(\mathbf{a})} - Z \boxed{+ X_3 + \mu \lfloor p/2 \rfloor} = X_3' - Z,$$
where X_3' collects the boxed terms. Note that by definition, μ and γ are determined by X_0, X_1, X_2. In H_4, the values in X_3 are uniformly random and independent of X_0, X_1, X_2 (thus μ, γ). This implies that X_3' is also uniformly random and independent of γ, so $\mathbf{L}_f = X_3' - Z$ is a uniformly random value perturbed by $Z = h(X)$.

The following probabilities are taken in H_4 and are implicitly intersected with the requirement that $f(\mathbf{x}) = 0$, γ is not the zero function, and $|Z| \leq B_{\mathcal{A}}$. We apply union bound over all possible values in $[-B_{\mathcal{A}}..B_{\mathcal{A}}]$ to obtain
$$\begin{aligned} A_4 &= \Pr\left[\gamma(\mathbf{L}_f) = 0\right] \\ &= \Pr\left[\gamma(X_3' - Z) = 0\right] \\ &\leq \Pr\left[\exists z \in [-B_{\mathcal{A}}..B_{\mathcal{A}}] \text{ s.t. } \gamma(X_3' - z) = 0\right] \\ &\leq \sum_{z=-B_{\mathcal{A}}}^{B_{\mathcal{A}}} \Pr\left[\gamma(X_3' - z) = 0\right] \leq \frac{2B_{\mathcal{A}} + 1}{p}, \end{aligned}$$
where the second inequality follows from Schwartz–Zippel lemma for degree 1. Since $B_{\mathcal{A}}$ is polynomial and p is super-polynomial, A_4 is negligible. □

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 3 (KP-ABE)**.** All variables x_λ are indexed by λ. For simplicity of notations, we suppress λ 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 5.
- 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 γ 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 $\mathbb{Z}_p$.

We construct a KP-ABE scheme for the predicate family Ckt defined as follows.

$$P^{\mathsf{Ckt}}_{\ell,d}(\mathbf{x}, C) = \neg C(\mathbf{x}) \quad \text{for } \mathbf{x} \in \{0,1\}^\ell, C \in \mathsf{Ckt}_{\ell,d},$$
$$\mathsf{Ckt} = \{P^{\mathsf{Ckt}}_{\ell,d} | \ell, d \in \mathbb{N}\}.$$

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

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

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

- $\mathsf{KeyGen}(\mathsf{msk}, C)$ takes as input the master secret key msk and a policy $C \in \mathsf{Ckt}_{\ell,d}$. Since the secret sharing scheme has linear function sharing (Definition 15), the $\mathsf{SS.ShareF}_{\mathsf{SS.pp},C}(\cdot,\cdot)$ function is a deterministic linear function with coefficients $\mathbf{c} = (c_\mu, \mathbf{c}_\mathbf{r})$. The KeyGen algorithm samples $\delta \xleftarrow{\$} \mathbb{Z}_p \setminus \{0\}$, 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 mpk, an attribute $\mathbf{x} \in \{0,1\}^\ell$, and a message $\mu \in \{0,1\}$. The algorithm runs

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

  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 mpk, a secret key sk, its associated policy C, a ciphertext 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 KeyGen. The algorithm next parses ct into $\mathbf{L}^{\mathbf{x}}$, 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 γ homomorphically in the exponent of G_{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 5, 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 SS.pp and impk, hence has bit length
  $|\mathsf{mpk}| = |\mathsf{SS.pp}| + |\mathsf{impk}| = \mathrm{poly}(\lambda, d) \cdot \ell$.
- The master secret key consists of imsk, hence has bit length
  $|\mathsf{msk}| = |\mathsf{imsk}| = \mathrm{poly}(\lambda, d)$.
- A secret key consists of a single 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 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 and prove (in Section 5.1) adaptive IND-CPA security of the scheme.

Proposition 18. *Suppose in Construction 3, 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. Specifically, for all efficient adversary $\mathcal{A}$ that distinguishes the adaptive ABE experiments with advantage $\varepsilon_{\mathcal{A}}$, there exist an efficient $\mathcal{B}_1$ that distinguishes the selective simulation security experiments of IPFE with advantage $\varepsilon_{\mathsf{IPFE}}$, and an efficient $\mathcal{B}_2$ that wins the non-annihilable game with advantage $\varepsilon_{\text{ANN-f}}$, such that:*

$$\varepsilon_{\mathcal{A}} \le \varepsilon_{\mathsf{IPFE}} + Q/q + TQ\varepsilon_{\text{ANN-f}}, \tag{8}$$

where T, Q are polynomial upper bounds on the number of zero-test queries and key queries from $\mathcal{A}$.

Remark 3. In Construction 3, if the secret sharing scheme is only selectively non-annihilable, then the resultant KP-ABE achieves selective security. The proof mostly remains unchanged, and the advantage relation in Proposition 18 holds except $\varepsilon_{\mathcal{A}}, \varepsilon_{\text{ANN-f}}$ are advantages in the selective games.

5.1 Proof of Proposition 18

Proof (Proposition 18). Fix an an efficient adversary $\mathcal{A}$, we construct adversaries $\mathcal{B}_1, \mathcal{B}_2$ against the selective simulation security of IPFE, and non-annihilability for $\mathbf{L}_f$. We bound the advantage $\varepsilon_{\mathcal{A}}$ of $\mathcal{A}$ in the non-annihilability game as in Equation (8) from the proposition statement.

Recall that in the adaptive security game of KP-ABE, the adversary first repeatedly queries policies $\{C_i\}$ and receives secret keys $\{\mathsf{sk}_i\}$ for them. It next adaptively chooses an attribute $\mathbf{x}$ as its challenge, and receives a ciphertext ct for it. After the challenge, it again repeatedly chooses policies $\{C_i\}$ and receives secret keys $\{\mathsf{sk}_i\}$ for them. In the end, it outputs a bit b to distinguish whether the encrypted message μ in ct is 0 or 1.

Recall that the selectively simulation-security of IPFE guarantees a simulator ($\mathsf{IPFE}.\widetilde{\mathsf{Setup}}$, $\mathsf{IPFE}.\widetilde{\mathsf{KeyGen}}$, $\mathsf{IPFE}.\widetilde{\mathsf{Enc}}$). We consider the following hybrids, and let A_i^μ be distinguishing advantages of $\mathcal{A}$ in H_i^μ for $i \in [3]$.

- H_1^μ is the adaptive security experiment $\mathsf{Exp}_{\mathrm{CPA}}^\mu$. Specifically, the challenger first runs

  $$(\mathsf{impk}, \mathsf{imsk}) \xleftarrow{\$} \mathsf{IPFE.Setup}(1^\lambda, 1^{(n+1)}),$$

  and then computes each queried secret key for C_i as follows.

  $$\mathsf{isk}_i \xleftarrow{\$} \mathsf{IPFE.KeyGen}(\mathsf{imsk}, [\![\delta_i \mathbf{c}_i]\!]_2), \qquad \mathsf{sk}_i = ([\![\delta_i]\!]_2, \mathsf{isk}_i)$$

  where δ_i is a fresh random non-zero element, and $\mathbf{c}_i$ are coefficients for $\mathsf{SS.ShareF}_{\mathsf{SS.pp},C_i}(\cdot,\cdot)$. The ciphertext for $\mathbf{x}$ is computed by first generating shares and randomness $(\mathbf{L}_0, \{\mathbf{L}_i^b\}, \mathbf{r})$ from running $\mathsf{SS.ShareX}$, and then set

  $$\mathsf{ict} \xleftarrow{\$} \mathsf{IPFE.Enc}(\mathsf{impk}, [\![(\mu, \mathbf{r})]\!]_1) \quad \mathsf{ct} = (\mathbf{L}^{\mathbf{x}}, \mathsf{ict}).$$

- H_2^μ proceeds identically to H_1 except that we replace the IPFE scheme with the simulator. Specifically, the challenger first runs

  $$(\mathsf{impk}, \widetilde{\mathsf{imsk}}) \xleftarrow{\$} \mathsf{IPFE}\widetilde{\mathsf{Setup}}(1^\lambda, 1^{(n+1)}),$$

  and then generates shares and randomness $(\mathbf{L}_0, \{\mathbf{L}_i^b\}, \mathbf{r})$ from running $\mathsf{SS.ShareX}$. Before receiving any queries or the challenge, it simulates the $\widetilde{\mathsf{ict}}$ by

  $$\widetilde{\mathsf{ict}} \xleftarrow{\$} \widetilde{\mathsf{Enc}}(\widetilde{\mathsf{imsk}}).$$

  The challenger computes each queried secret key for C_i by computing

  $$L_{C_i} = \mathsf{SS.ShareF}(\mathsf{SS.pp}, C_i, \mu, \mathbf{r}), \quad \widetilde{\mathsf{isk}}_i \xleftarrow{\$} \widetilde{\mathsf{KeyGen}}(\widetilde{\mathsf{imsk}}, [\![\delta_i \mathbf{c}_i]\!]_2, [\![\delta_i L_{C_i}]\!]_2),$$

  and setting $\mathsf{sk}_i = ([\![\delta_i]\!]_2, \widetilde{\mathsf{isk}}_i)$. Upon receiving the challenge $\mathbf{x}$, it answers with $\mathsf{ct} = (\mathbf{L}^{\mathbf{x}}, \widetilde{\mathsf{ict}})$.

  Claim 19. There exists an efficient $\mathcal{B}_1$ that distinguishes the selective simulation experiments with advantage $\varepsilon_{\mathsf{IPFE}}$ such that $|A_1^\mu - A_2^\mu| \le \varepsilon_{\mathsf{IPFE}}$.

- In H_3^μ, the challenger does not run the secret sharing algorithms anymore. Instead, it obtains $\mathsf{SS.pp}$, $[\![\delta_i, \delta_i L_{C_i}]\!]_2$ and $\mathbf{L}^{\mathbf{x}}$ from interacting (as $\mathcal{A}'$) with the $\mathsf{Exp}_{\mathrm{KP}}^\mu$ experiment, where $\mathsf{Exp}_{\mathrm{KP}}^\mu(1^\lambda)$ with a machine $\mathcal{A}'$ proceeds as follows:

- **Setup.** Launch $\mathcal{A}'(1^\lambda)$ and receive from it the input length 1^ℓ. First, run $\mathsf{SS.Setup}(1^\lambda, 1^\ell)$ to generate pp, and send it to $\mathcal{A}'$. Next, run $\mathsf{SS.Share}(\mathsf{pp})$ to generate $(\mathbf{L}_0, \{\mathbf{L}_k^b\}_{k\in[\ell]}^{b\in\{0,1\}}, r)$.
- **Query I.** Repeat the following for arbitrarily many rounds determined by $\mathcal{A}'$: In each round, $\mathcal{A}'$ submit some $C_i \in \mathsf{Ckt}_\ell^d$. Upon this query, sample a random element $[\![\delta_i]\!]_2$ encoded in G_2 and run

$$\mathbf{L}_{C_i} \xleftarrow{\$} \mathsf{SS.ShareF}(\mathsf{pp}, C_i, \mu, r).$$

  Send $([\![\delta_i \mathbf{L}_{C_i}]\!]_2, [\![\delta_i]\!]_2)$ to $\mathcal{A}'$.
- **Challenge.** The adversary submits some $\mathbf{x}^* \in \{0,1\}^\ell$. Return $\mathbf{L}^{\mathbf{x}^*}$ to $\mathcal{A}'$.
- **Query II.** Same as Query I.
- **Guess.** The adversary outputs a bit μ'. The outcome of the experiment is μ' if $C_i(\mathbf{x}^*) = 0$ for all C_i queried in Query I/II. Otherwise, the outcome is set to 0.

$|A_2^\mu - A_3^\mu| = 0$ as H_3^μ is the same as H_2^μ by construction. We show the following claim:

Claim 20. In GGM, there exists efficient $\mathcal{B}_2$ that wins the non-annihilability games for $\mathbf{L}_f$ with advantages $\varepsilon_{\text{ANN-f}}$, such that

$$|A_3^\mu| \leq \frac{Q}{q} + TQ\varepsilon_{\text{ANN-f}},$$

where T, Q are polynomial upper bounds on the number of zero-test queries and key queries from $\mathcal{A}$.

□

We now prove the claims.

Proof (Claim 19). We prove this claim by reduction to the selective simulation security of the IPFE scheme (see Definition 11). We construct a distinguisher $\mathcal{B}$ for the selective simulation games $\mathsf{Exp}_{\mathsf{real}}$ and $\mathsf{Exp}_{\mathsf{sim}}$ as follows:

1. $\mathcal{B}$ launches $\mathcal{A}(1^\lambda)$ with fresh randomness $r_\mathcal{A}$, and receives from it a predicate $P \in \mathsf{Ckt}_\lambda$. Let ℓ, d be the attribute length and depth of P.

2. $\mathcal{B}$ runs

$$\mathsf{SS.pp} \xleftarrow{\$} \mathsf{SS.Setup}(1^\lambda, 1^\ell, 1^d), \quad (\mathbf{L}_0, \{\mathbf{L}_i^b\}_i^b, \mathbf{r}) \xleftarrow{\$} \mathsf{SS.ShareX}(\mathsf{SS.pp}),$$

   Let n be the size of r output by $\mathsf{SS.ShareX}$, and let $N = n+1$. $\mathcal{B}$ sends 1^N, and the concatenation of $\mu, \mathbf{r}$ to the challenger.

3. $\mathcal{B}$ receives impk, and an ict from the challenger. It sends $\mathsf{mpk} = (\mathsf{SS.pp}, \mathsf{impk})$ to the adversary $\mathcal{A}$.

4. Upon receiving a query $C_i \in P$ from $\mathcal{A}$, the distinguisher $\mathcal{B}$ samples a random non-zero element δ_i, and computes the coefficient vector $\mathbf{c}_i$ for $\mathsf{SS.ShareF}_{\mathsf{SS.pp}, C_i}(\cdot,\cdot)$. It then sends $[\![\delta_i \mathbf{c}_i]\!]_2$ to the challenger, and receives isk_i back. Finally, $\mathcal{B}$ answers $\mathcal{A}$ with $\mathsf{sk}_i = ([\![\delta_i]\!]_2, \mathsf{isk}_i)$.

5. Upon receiving a challenge $\mathbf{x} \in \{0,1\}^\ell$, the distinguisher $\mathcal{B}$ answers $\mathcal{A}$ with $\mathsf{ct} = (\mathbf{L}^{\mathbf{x}}, \mathsf{ict})$. (The ict was received from the challenger in step 3.)

6. In the end, $\mathcal{B}$ receives an output bit b from $\mathcal{A}$. It outputs 1 if and only if $b = \mu$.

First note that $\mathcal{B}$ is efficient. In $\mathsf{Exp}_{\mathrm{real}}$, the distinguisher $\mathcal{B}$ emulates H_1^μ for $\mathcal{A}$, and outputs whether $\mathcal{A}$'s output matches μ. In $\mathsf{Exp}_{\mathrm{sim}}$, it does so for H_2^μ. Therefore,

$$|A_1^\mu - A_2^\mu| = \left|\Pr[\mathcal{B} \to 1 \text{ in } \mathsf{Exp}_{\mathrm{real}}] - \Pr[\mathcal{B} \to 1 \text{ in } \mathsf{Exp}_{\mathrm{sim}}]\right|$$

□

Proof (Claim 20). Let $\mathcal{A}'$ be the challenger in H_3. We consider the following hybrids, and let $A_i'^\mu$ be distinguishing advantages of $\mathcal{A}'$ in G_i^μ for $i \in [3]$. (Note that $A_1'^\mu = A_3^\mu$.)

- G_1^μ is the experiment $\mathsf{Exp}_{\mathrm{KP}}^\mu$. Specifically, throughout the experiment, the GGM oracle answers zero-test queries from the adversary of the form

$$\gamma(\{\delta_i \mathbf{L}_{C_i}\}_i, \{\delta_i\}_i).$$

- G_2^μ proceeds identically as G_1^μ, except that the challenger views each zero-test query from $\mathcal{A}'$ as a degree-1 polynomial where δ_i are the variables:

$$\gamma(\{\delta_i \mathbf{L}_{C_i}\}_i, \{\delta_i\}_i) = \sum_i \gamma_i(\mathbf{L}_{C_i})\delta_i + \alpha_0,$$

  where α_0 is a constant. The challenger answers the query with zero if and only if γ_i evaluates to zero for all i, and $\alpha_0 = 0$.

  Let Q be the maximum number of zero-test queries from $\mathcal{A}'$. Since δ_i are sampled independently at random, by Schwartz–Zippel lemma of degree-1 we have $|A_1'^\mu - A_2'^\mu| \le \frac{Q}{q}$.

- G_3^μ proceeds identically as G_2^μ, except that the challenger answers zero-test queries with zero if and only all γ_i are the zero function, and $\alpha_0 = 0$.

  Note that in GGM, $\mathcal{A}'$ only gains information through zero-test queries. In G_3, all queries are answered independently of the shares. Hence $\mathcal{A}'$ has zero advantage, i.e., $A_3'^\mu = 0$. It remains to construct efficient $\mathcal{B}_2$ that wins the non-annihilability game for $\mathbf{L}_f$ with advantage $\varepsilon_{\text{ANN-f}}$, such that

$$|A_2'^\mu - A_3'^\mu| \le TQ\varepsilon_{\text{ANN-f}},$$

  where T, Q are polynomial upper bounds on the number of zero-test queries and key queries (for C_i) from $\mathcal{A}'$.

Note that G_2^μ and G_3^μ differs if and only if there exists i, j such that when viewing the j^{th} zero-test query from $\mathcal{A}$ as a polynomial over the δ's, the affine function γ_i is non-zero, but evaluates to zero. Let $E_{i,j}$ denote such an event for i, j, and let E be the union of all $E_{i,j}$. Let T, Q be polynomial upper bounds on the number of zero-test queries and key queries (for C_i) from $\mathcal{A}'$. We construct an efficient adversary $\mathcal{B}_2$ for the non-annihilability game for $\mathbf{L}_f$ as follows:

- $\mathcal{B}_2$ samples $t \xleftarrow{\$} [T]$ and $q \xleftarrow{\$} [Q]$, and then launches $\mathcal{A}'(1^\lambda)$.
- $\mathcal{B}_2$ receives the input length 1^ℓ from $\mathcal{A}'$, and forwards 1^ℓ to the challenger. $\mathcal{B}_2$ next receives $\mathsf{SS.pp}$ from the challenger, and forwards it to $\mathcal{A}'$.
- Upon receiving a query C_i, the adversary $\mathcal{B}_2$ stores C_i, and answers with new group handles in G_2.
- Upon receiving the challenge $\mathbf{x}^*$, the adversary $\mathcal{B}_2$ forwards $\mathbf{x}^*$ to the challenger, and receives $\mathbf{L}^{\mathbf{x}^*}$ back. $\mathcal{B}_2$ forwards $\mathbf{L}^{\mathbf{x}^*}$ to $\mathcal{A}'$.
- Upon receiving the j^{th} zero-test query, if $j < q$ then $\mathcal{B}_2$ answers as in G_3^μ, , by looking only at the coefficients of all γ_i. If $j = q$, then $\mathcal{B}_2$ uses C_i and γ_i in this query, together with b as its output.

Note that if E happens, and $\mathcal{B}_2$ samples t, q exactly equal to the smallest i, j such that $E_{i,j}$ happens, then $\mathcal{B}_2$ wins the non-annihilability game. That is:

$$|A_2'^\mu - A_3'^\mu| \leq \Pr[E] \leq TQ \cdot \Pr[\mathcal{B}_2 \text{ wins in } \mathsf{Exp}_{\mathrm{ANN}}].$$

□

Combining Construction 3, Proposition 18, Construction 1, Proposition 6, Lemma 4, and Lemma 5, we have the following theorem:

Theorem 21 (KP-ABE). *Assume the polynomial hardness of adaptive LWE, with subexponential modulus-to-noise ratio. That is, there exists an arbitrary constant $\varepsilon > 0$ such that $\mathrm{ALWE}_{n,q,\chi}$ with $\alpha = \frac{q}{B} = 2^{n^\varepsilon}$ holds, where χ is the discrete Gaussian over $\mathbb{Z}$ of width B/λ truncated within absolute value of $B = \mathrm{poly}(\lambda)$.*

Let λ be the security parameter, and $p = 2^{\omega(\log \lambda)}$ be an arbitrary prime. In the generic pairing group model with order p, there exists a KP-ABE scheme for circuits that

- *satisfies the adaptive IND-CPA security, and*
- *has constant-size secret keys $|\mathsf{sk}_C| = \mathrm{poly}(\lambda)$ (concretely, containing 3 group elements) and ciphertexts of size $|\mathsf{ct}_\mathbf{x}| = \mathrm{poly}(\lambda, d)\ell$ where ℓ is the attribute length and d is maximum depth of the policy circuits.*

Specifically, for all $\lambda \in \mathbb{N}$, for all efficient adversary $\mathcal{A}$ that distinguishes the adaptive ABE experiments with advantage $\varepsilon_\mathcal{A}$, and for all $k \in \mathbb{N}$, there exist an efficient $\mathcal{B}_1$ that distinguishes the selective simulation security experiments of IPFE with advantage $\varepsilon_{\mathsf{IPFE}}$, and an efficient $\mathcal{B}_2$ that distinguishes the adaptive LWE experiments with advantage $\varepsilon_{\mathrm{ALWE}}$, such that

$$\varepsilon_\mathcal{A} \leq \varepsilon_{\mathsf{IPFE}} + \lambda^{-k} + \mathrm{poly}(|\mathcal{A}|)\varepsilon_{\mathrm{ALWE}} + \frac{\mathrm{poly}(|\mathcal{A}|, \lambda)}{p}$$

6 Doubly Succinct CP-ABE

6.1 Stronger IPFE in GGM

Definition 19 (adaptive simulation in GGM). A *simulator* for an IPFE scheme (Definition 9) in GGM consists of 4 efficient algorithms:

- $\widetilde{\mathsf{Setup}}(1^\lambda, 1^N)$ outputs a simulated master public key $\widetilde{\mathsf{impk}}$ and an initial state st.
- $\widetilde{\mathsf{KeyGen}}(\mathsf{st})$ outputs an updated state st′ and a simulated key $\widetilde{\mathsf{isk}}$.
- $\widetilde{\mathsf{Enc}}(\mathsf{st})$ outputs an updated state st′ and a simulated ciphertext $\widetilde{\mathsf{ict}}$.
- $\widetilde{\mathsf{GrpZT}}^{\mathsf{InPrdZT}(\cdot)}(\mathsf{st}, \gamma)$ takes as input a zero-test query γ, which is an affine function over all possible parings of the group handles created by earlier calls to the other simulation algorithms. It has oracle access to $\mathsf{InPrdZT}(\cdot)$, which performs zero-tests of affine functions of the inner products and the vectors in the keys (described in more details later). The algorithm outputs an updated state st′ and an answer z to the query γ.

The scheme is *adaptively simulation-secure in GGM* if there exists a simulator such that $\mathsf{Exp}_{\mathsf{real}}$ and $\mathsf{Exp}_{\mathsf{sim}}$ are indistinguishable, where the experiments $\mathsf{Exp}_{\mathsf{real}}(1^\lambda), \mathsf{Exp}_{\mathsf{sim}}(1^\lambda)$ with $\mathcal{A}$ proceed as follows:

- **Setup.** The challenger launches $\mathcal{A}(1^\lambda)$ and receives the vector length 1^N. It runs

$$\begin{aligned} &\text{in } \mathsf{Exp}_{\mathsf{real}}: && (\mathsf{impk}, \mathsf{imsk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, 1^N); \\ &\text{in } \mathsf{Exp}_{\mathsf{sim}}: && (\mathsf{impk}, \mathsf{st}) \xleftarrow{\$} \widetilde{\mathsf{Setup}}(1^\lambda, 1^N); \end{aligned}$$

  and sends impk to $\mathcal{A}$. (Since we are considering unbounded simulation, the third argument to Setup does not matter.)

- **Challenge.** The following is repeated for arbitrarily many rounds determined by $\mathcal{A}$: In each round, $\mathcal{A}$ submits a vector $\mathbf{u}_i$ encoded in G_1 or a vector $\mathbf{v}_j$ encoded in G_2. Upon receiving the query, the challenger runs

$$\begin{aligned} &\text{in } \mathsf{Exp}_{\mathsf{real}}: && \mathsf{ict}_i \xleftarrow{\$} \mathsf{Enc}(\mathsf{impk}, [\![\mathbf{u}_i]\!]_1) && \text{or} && \mathsf{isk}_j \xleftarrow{\$} \mathsf{KeyGen}(\mathsf{imsk}, [\![\mathbf{v}_j]\!]_2) \\ &\text{in } \mathsf{Exp}_{\mathsf{sim}}: && (\mathsf{ict}_i, \mathsf{st}') \xleftarrow{\$} \widetilde{\mathsf{Enc}}(1^N, \mathsf{st}) && \text{or} && (\mathsf{isk}_j, \mathsf{st}') \xleftarrow{\$} \widetilde{\mathsf{KeyGen}}(\mathsf{st}) \end{aligned}$$

  and sends ict_i or isk_j to $\mathcal{A}$. Additionally, in $\mathsf{Exp}_{\mathsf{sim}}$ the challenger updates the state $\mathsf{st} \leftarrow \mathsf{st}'$, and maintains a database of the key vectors and the inner products $D = (\{[\![\mathbf{v}_j]\!]_2\}_{j\in[n]}, \{[\![\langle \mathbf{u}_i, \mathbf{v}_j\rangle]\!]_{\mathrm{T}}\}_{i\in[m],j\in[n]})$.

- **Guess.** The adversary $\mathcal{A}$ outputs a bit b, which is the output of the experiment.

The adversary can submit zero-test queries at any time during the experiment. In $\mathsf{Exp}_{\mathsf{real}}$, the GGM oracle handles such queries, whereas in $\mathsf{Exp}_{\mathsf{sim}}$ upon receiving a query γ, the challenger runs

$$\mathsf{st}', z \xleftarrow{\$} \widetilde{\mathsf{GrpZT}}^{\mathsf{InPrdZT}(D,\cdot)}(\mathsf{st}, \gamma),$$

where the InPrdZT algorithm answers zero-test queries from $\widetilde{\mathsf{GrpZT}}$ using keys and inner products stored in D. Additionally, in $\mathsf{Exp}_{\mathsf{sim}}$ the challenger updates the state $\mathsf{st} \leftarrow \mathsf{st}'$.

6.2 ABDP Scheme

The IPFE scheme in [ABDP15] is only known to be selectively IND-CPA secure in the standard model. It turns out that the very same scheme satisfies the (much stronger) adaptive simulation security in GGM. We recall the construction below.

Construction 4 ([ABDP15])**.** The construction is described for a fixed value of λ, and λ is suppressed for brevity. The ABDP scheme in pairing groups of order p works as follows:

- $\mathsf{Setup}(1^N)$ samples $a \xleftarrow{\$} \mathbb{Z}_p, \mathbf{w} \xleftarrow{\$} \mathbb{Z}_p^N$ and outputs $\mathsf{impk} = \llbracket a, a\mathbf{w} \rrbracket_1, \mathsf{imsk} = \mathbf{w}$.
- $\mathsf{KeyGen}(\mathsf{imsk}, \llbracket \mathbf{v} \rrbracket_2)$ outputs the secret key $\mathsf{isk} = \mathbf{w}^\intercal \llbracket \mathbf{v} \rrbracket_2$.
- $\mathsf{Enc}(\mathsf{impk}, \llbracket \mathbf{u} \rrbracket_1)$ samples $s \xleftarrow{\$} \mathbb{Z}_p$ and outputs the ciphertext

$$\mathsf{ict} = (\mathsf{ict}_1, \mathsf{ict}_2) = (s\llbracket a \rrbracket_1, s\llbracket a\mathbf{w} \rrbracket_1 + \llbracket \mathbf{u} \rrbracket_1).$$

- $\mathsf{Dec}(\mathsf{isk}, \llbracket \mathbf{v} \rrbracket_2, \mathsf{ict})$ parses ict into $(\mathsf{ict}_1, \mathsf{ict}_2)$ and outputs $-\mathsf{ict}_1 \cdot \mathsf{isk} + \mathsf{ict}_2^\intercal \llbracket \mathbf{v} \rrbracket_2$.

Correctness is readily verified by $-sa \cdot \mathbf{w}^\intercal \mathbf{v} + (sa\mathbf{w} + \mathbf{u})^\intercal \mathbf{v} = \mathbf{u}^\intercal \mathbf{v}$. The scheme is succinct, as each secret key consists of only 1 group element, independent of N.

We note that the above construction has two minor differences compared to the original version in [ABDP15]. One is that in the original version, key vectors are encoded in $\mathbb{Z}_p$ instead of any group G_2 that can pair with G_1. The other is that the master public key in [ABDP15] is simply $\llbracket \mathbf{w} \rrbracket_1$, without the random scalar a. As we will see later, the introduction of random scalar a makes our security analysis easier.

We prove (in Appendix 6.3) adaptive simulation-security security of Construction 4 in GGM.

Proposition 22. *Construction 4 is adaptively simulation-secure in GGM.*

6.3 Proof of Proposition 22

Proof (Proposition 22). Let $\mathcal{A}$ be any efficient adversary. Recall that in the security game, $\mathcal{A}$ first receives a master public key impk, and then at each round receives either a a ciphertext ict_i or a secret key isk_j for its query. $\mathcal{A}$ also receives answers to its zero-test queries whenever it submits such queries. In the end, $\mathcal{A}$ outputs a bit b to distinguish whether its in $\mathsf{Exp}_{\mathsf{real}}$ or $\mathsf{Exp}_{\mathsf{sim}}$.

Our simulator simply produces new group handles for the simulated master public key $\widetilde{\mathsf{impk}}$, secret keys ciphertexts $\{\widetilde{\mathsf{ict}}_i\}$ and $\{\widetilde{\mathsf{isk}}_j\}$. The simulator also updates its state to keep a counter $C = (i, j)$ for the number of (i) queried ciphertexts and (j) keys. We focus on the algorithm $\widetilde{\mathsf{GrpZT}}$ for answering zero-test queries from $\mathcal{A}$ using $\mathsf{InPrdZT}(D, \cdot)$, where $\mathsf{InPrdZT}(D, \gamma')$ honestly evaluates an affine query γ' over values stored in D. We develop $\widetilde{\mathsf{GrpZT}}$ through the following hybrids, and the $\widetilde{\mathsf{GrpZT}}$ operates as described in the final hybrid.

- H_1 is the real-world experiment $\mathsf{Exp}_{\mathsf{real}}$, where the master public key impk ciphertexts $\{\mathsf{ict}_i\}$ and secret keys $\{\mathsf{isk}_j\}$ are generated as

$$\mathsf{impk} = \{\!\{ a, a\mathbf{w} \}\!\}_1, \quad \mathsf{ict}_i = \{\!\{ s_i a, s_i a\mathbf{w} + \mathbf{u}_i \}\!\}_1, \quad \mathsf{isk}_j = \{\!\{ \mathbf{w}^\intercal \mathbf{v}_j \}\!\}_2.$$

  The zero-test queries are answered by the GGM oracle.

- H_2 proceeds identically to H_1, except we change how the master public key and the ciphertexts are created:

$$\mathsf{impk} = \{\!\{ a_0, a_0\mathbf{w} \}\!\}_1, \quad \mathsf{ict}_i = \{\!\{ a_i, a_i\mathbf{w} + \mathbf{u}_i \}\!\}_1, \quad \mathsf{isk}_j = \{\!\{ \mathbf{w}^\intercal \mathbf{v}_j \}\!\}_2,$$

where a_i's are uniformly random over $\mathbb{Z}_p$. For convenience, we write $\mathbf{u}_0 = 0$, and $\mathsf{impk} = [\![a_0, a_0\mathbf{w} + \mathbf{u}_0]\!]_1$. Each zero-test query from $\mathcal{A}$ is of the form

$$\gamma\big((1, \{a_i\}_i, \{a_i\mathbf{w} + \mathbf{u}_i\}_i) \otimes (1, \{\mathbf{v}_j\}_j, \{\mathbf{w}^\intercal\mathbf{v}_j\}_j)\big).$$

Since each s_i in H_1 is sampled independently at random, H_2 is the same as H_1.

- H_3 proceeds identically to H_2, except the challenger views each zero-test query from $\mathcal{A}$ as a degree-3 polynomial where $a_i, \mathbf{w}$ are the variables:

$$\begin{aligned}&\gamma\big((1, \{a_i\}_i, \{a_i\mathbf{w} + \mathbf{u}_i\}_i) \otimes (1, \{\mathbf{v}_j\}_j, \{\mathbf{w}^\intercal\mathbf{v}_j\}_j)\big)\\ &= \gamma_0\big(\{\mathbf{u}_i\}_i, \{\mathbf{v}_j\}_j, \{\mathbf{u}_i \otimes \mathbf{v}_j\}_{i,j}\big) + \gamma_1\big(\{\mathbf{v}_j\}_j, \{\mathbf{u}_i \otimes \mathbf{v}_j\big) \cdot \mathbf{w}\\ &\quad + \sum_i \gamma_{2,i}\big(\{\mathbf{v}_j\}_j\big) \cdot a_i + \sum_i \gamma_{3,i}\big(\{\mathbf{v}_j\}_j\big) \cdot a_i\mathbf{w} + \sum_i a_i\mathbf{w}^\intercal \cdot \gamma_{4,i}\big(\{\mathbf{v}_j\}_j\big) \cdot \mathbf{w}.\end{aligned}$$

  The challenger answers with zero if and only if $\gamma_0, \gamma_1, \{\gamma_{2,i}, \gamma_{3,i}\}_i$ all evaluate to zero and $\gamma_{4,i}^\intercal\big(\{\mathbf{v}_j\}_j\big) + \gamma_{4,i}\big(\{\mathbf{v}_j\}_j\big) = 0$ for all i. [7] Since $a_i, \mathbf{w}$ are sampled at random, H_2 and H_3 differ (i.e., γ is a non-zero polynomial over $a_i, \mathbf{w}$, but evaluates to 0) with only negligible probability by Schwartz–Zippel lemma of degree-3.

- H_4 proceeds identically to H_3, except the challenger translates γ_0, γ_1 equivalently into γ_0', γ_1' that are affine functions over only $\{\mathbf{v}_j\}_j$ and $\{\mathbf{u}_i^\intercal\mathbf{v}_j\}_{i,j}$ (in particular, not over $\{\mathbf{u}_i\}_i$ or $\{\mathbf{u}_i \otimes \mathbf{v}_j\}_{i,j}$). The challenger runs $\mathsf{InPrdZT}(D, \cdot)$ to evaluate $\gamma_0', \gamma_1', \{\gamma_{2,i}, \gamma_{3,i}, \gamma_{4,i}\}_i$. Since the translated γ_0', γ_1' are required to be equivalent to γ_0, γ_1, the hybrids H_4 and H_3 are the same.

Note that in H_4, the challenger needs two things to answer zero-test queries: translation from γ_0, γ_1 to γ_0', γ_1' and access to $\mathsf{InPrdZT}(D, \cdot)$. If we can show the translation only requires a counter $C = (i, j)$ of queried ciphertexts and keys, then H_4 indeed specifies GrpZT. Since $\mathcal{A}$ only learns information through zero-test queries, H_4 and $\mathsf{Exp}_{\mathsf{sim}}$ are identical.

We now show the translation from γ_0, γ_1. First, write out a zero-test γ from $\mathcal{A}$ explicitly,

$$\begin{aligned}&\gamma\big((1, \{a_i\}_i, \{a_i\mathbf{w} + \mathbf{u}_i\}_i) \otimes (1, \{\mathbf{v}_j\}_j, \{\mathbf{w}^\intercal\mathbf{v}_j\}_j)\big)\\ &= \alpha + \sum_i \big(b_i \cdot a_i + \mathbf{c}_i^\intercal \cdot (a_i\mathbf{w} + \mathbf{u}_i)\big) + \sum_j \big(d_j \cdot \mathbf{w}^\intercal\mathbf{v}_j + \mathbf{f}_j^\intercal \cdot \mathbf{v}_j\big)\\ &\quad + \sum_{ij} \big(g_{ij} \cdot a_i \cdot \mathbf{w}^\intercal\mathbf{v}_j + \mathbf{h}_{ij}^\intercal \cdot a_i \cdot \mathbf{v}_j\\ &\qquad\qquad + \mathbf{m}_{ij}^\intercal \cdot (a_i\mathbf{w} + \mathbf{u}_i) \cdot \mathbf{w}^\intercal\mathbf{v}_j + (a_i\mathbf{w} + \mathbf{u}_i)^\intercal \cdot \mathbf{N}_{ij} \cdot \mathbf{v}_j\big).\end{aligned}$$

where $\alpha, b_i, \mathbf{c}_i, d_j, \mathbf{f}_j, g_{ij}, \mathbf{h}_{ij}, \mathbf{m}_{ij}, \mathbf{N}_{ij}$ are (vectors or matrices of) coefficients. We remark that the last term $(a_i\mathbf{w} + \mathbf{u}_i)^\intercal \cdot \mathbf{N}_{ij} \cdot \mathbf{v}_j$ enables arbitrary cross-pairing of $(a_i\mathbf{w} + \mathbf{u}_i)$ with $\mathbf{v}_j$ and combining the results. Each entry in $\mathbf{N}_{ij}$ corresponds to the coefficient of one such pairing result. Using these coefficients, we can explicitly write $\gamma_0, \gamma_1, \{\gamma_{2,i}, \gamma_{3,i}, \gamma_{4,i}\}_i$ as follows:

$$\gamma_0\big(\{\mathbf{u}_i\}_i, \{\mathbf{v}_j\}_j, \{\mathbf{u}_i \otimes \mathbf{v}_j\}_{i,j}\big) = \alpha + \sum_i \mathbf{u}_i^\intercal\mathbf{c}_i + \sum_j \mathbf{f}_j^\intercal\mathbf{v}_j + \sum_{ij} \mathbf{u}_i^\intercal\mathbf{N}_{ij}\mathbf{v}_j \tag{9}$$

[7] Due to the symmetry of $\mathbf{w}$ and $\mathbf{w}^\intercal$, $\gamma_{4,i}^\intercal\big(\{\mathbf{v}_j\}_j\big) + \gamma_{4,i}\big(\{\mathbf{v}_j\}_j\big) = 0$ is enough to make all coefficients of terms $a_i\mathbf{w}[a]\mathbf{w}[b]$ zero.

$$\gamma_1(\{\mathbf{v}_j\}_j, \{\mathbf{u}_i \otimes \mathbf{v}_j) = \sum_j d_j \mathbf{v}_j + \sum_{ij} \mathbf{v}_j \mathbf{m}_{ij}^\intercal \mathbf{u}_i \tag{10}$$

$$\gamma_{2,i}(\{\mathbf{v}_j\}_j) = b_i + \sum_j \mathbf{h}_{ij}^\intercal \mathbf{v}_j \tag{11}$$

$$\gamma_{3,i}(\{\mathbf{v}_j\}_j) = \mathbf{c}_i + \sum_j (g_{ij}\mathbf{v}_j + \mathbf{N}_{ij}\mathbf{v}_j) \tag{12}$$

$$\gamma_{4,i}(\{\mathbf{v}_j\}_j) = \sum_j \mathbf{m}_{ij}\mathbf{v}_j^\intercal \tag{13}$$

Since in H_3, the challenger answers zero only if $\gamma_{3,i}(\{\mathbf{v}_j\}_j) = \mathbf{0}$, by (12) we can substitute $\mathbf{c}_i + \sum_j \mathbf{N}_{ij}\mathbf{v}_j$ with $\sum_j g_{ij}\mathbf{v}_j$ in (9) to define

$$\gamma_0'(\{\mathbf{u}_i\}_i, \{\mathbf{v}_j\}_j, \{\mathbf{u}_i^\intercal \mathbf{v}_j\}_{ij}) = \alpha + \sum_j \mathbf{f}_j^\intercal \mathbf{v}_j - \sum_{ij} g_{ij}\mathbf{u}_i^\intercal \mathbf{v}_j.$$

Similarly, since in H_3 the challenger answers zero only if $\gamma_{4,i}^\intercal(\{\mathbf{v}_j\}_j) + \gamma_{4,i}(\{\mathbf{v}_j\}_j) = 0$, by (13) we can substitute $\sum_j \mathbf{v}_j \mathbf{m}_{ij}^\intercal$ with $\sum_j \mathbf{m}_{ij}\mathbf{v}_j^\intercal$ in (10) to define

$$\gamma_1'(\{\mathbf{v}_j\}_j, \{\mathbf{u}_i^\intercal \mathbf{v}_j\}_{ij}) = \sum_j d_j \mathbf{v}_j - \sum_{ij} \mathbf{m}_{ij}\mathbf{v}_j^\intercal \mathbf{u}_i.$$

□

6.4 Doubly Succinct CP-ABE for Boolean Formulae

In this section, we combine a succinct nearly linear secret sharing scheme and a key-succinct IPFE scheme to obtain a doubly succinct CP-ABE scheme.

Construction 5 (doubly succinct CP-ABE). All variables x_λ are indexed by λ. For simplicity of notations, we suppress λ in subscripts. Our construction uses the following two ingredients:

- A key-succinct IPFE scheme ($\mathsf{IPFE.Setup}, \mathsf{IPFE.KeyGen}, \mathsf{IPFE.Enc}, \mathsf{IPFE.Dec}$) with modulus p given by Construction 4.
- A secret sharing scheme ($\mathsf{SS.Setup}, \mathsf{SS.ShareX}, \mathsf{SS.ShareF}, \mathsf{SS.FindCoef}$) for Boolean formulae as in Construction 2. Recall that the scheme has two properties. First, the shares are succinct: $\mathbf{L}_0$, and $\mathbf{L}_i^b$ are vectors of length $m = \mathrm{poly}(\lambda)$, and L_f is a single element. Second, the scheme has nearly linear reconstruction: the algorithm $\mathsf{SS.FindCoef}$ outputs an affine function γ over $L_f, \mathbf{L}_x$ that approximately evaluates to $\mu\lfloor p/2\rceil$.

We construct a CP-ABE scheme for the predicate family $\mathsf{5PBP}$ defined as follows.

$$P_\ell^{\mathsf{5PBP}}(\mathbf{x}, f) = \neg f(\mathbf{x}) \quad \text{for } \mathbf{x} \in \{0,1\}^\ell, f \in \mathsf{5PBP}_\ell,$$
$$\mathsf{5PBP}_\lambda = \{P_\ell^{\mathsf{5PBP}} | \ell \in \mathbb{N}\}.$$

- $\mathsf{Setup}(1^\lambda, P)$ takes as input the security parameter 1^λ and a predicate $P \in \mathsf{5PBP}$. Let ℓ be the attribute length for P, and m be the size of a share output by $\mathsf{SS.ShareX}$. The algorithm runs and sets

$$\mathsf{SS.pp} \xleftarrow{\$} \mathsf{SS.Setup}(1^\lambda, 1^\ell),$$
$$(\mathsf{impk}, \mathsf{imsk}) \xleftarrow{\$} \mathsf{IPFE.Setup}(1^\lambda, 1^N) \text{ for dimension } N = (2\ell + 1),$$
$$\mathsf{mpk} = (\mathsf{SS.pp}, \mathsf{impk}), \quad \mathsf{msk} = \mathsf{imsk}.$$

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

- $\mathsf{KeyGen}(\mathsf{msk}, \mathbf{x})$: takes as input the master secret key msk and an attribute $\mathbf{x} \in \{0,1\}^\ell$. It samples $\delta \xleftarrow{\$} \mathbb{Z}_p \setminus \{0\}$ and defines the "selecting vector" $\mathbf{v} \in \mathbb{Z}_p^{2\ell+1}$ as:

$$\mathbf{v}[j] = \begin{cases} 1, & \text{if } j = 1; \\ b\mathbf{x}[i] + (1-b)(1-\mathbf{x}[i]), & \text{if } j = 2i + b \text{ for } i \in [\ell], b \in \{0,1\}. \end{cases}$$

  it runs $\mathsf{isk} \xleftarrow{\$} \mathsf{IPFE.KeyGen}(\mathsf{imsk}, [\![\delta\mathbf{v}]\!]_2)$ and outputs $\mathsf{sk} = ([\![\delta]\!]_2, \mathsf{isk})$ as the secret key for $\mathbf{x}$.

  Note: *The vector* $\mathbf{v}$ *is formed as follows. First, put a* 1*, which will be used to select the shares* L_f *and* $\mathbf{L}_0$*. Next, for each* $i \in [\ell]$*, append either* $(0,1)$ *if* $\mathbf{x}[i] = 1$*, or* $(1,0)$ *if* $\mathbf{x}[i] = 0$*, which will be used to select the share* $\mathbf{L}_i^{\mathbf{x}[i]}$*. Since the ABE key consists of a single group element and an IPFE key, if the IPFE scheme is key-succinct, then the ABE is also key-succinct.*

- $\mathsf{Enc}(\mathsf{mpk}, f, \mu)$ takes as input the master public key mpk, a policy $f \in \mathsf{5PBP}_\ell$, and a message $\mu \in \{0,1\}$. Let m be the dimensions of $\mathbf{L}_0, \{\mathbf{L}_i^b\}_{i\in[\ell]}^{b\in\{0,1\}}$, and $N = 2\ell + 1$ be the vector length for IPFE. The algorithm runs SS.ShareX and SS.ShareF, and defines vectors

$$\big(\mathbf{L}_0, \{\mathbf{L}_i^b\}_{i\in[\ell]}^{b\in\{0,1\}}, r\big) \xleftarrow{\$} \mathsf{SS.ShareX}(\mathsf{SS.pp}), \quad L_f \xleftarrow{\$} \mathsf{SS.ShareF}(\mathsf{SS.pp}, f, \mu, r),$$

$$\text{for } j' \in [N]: \qquad \mathbf{u}_f[j'] = \begin{cases} L_f, & \text{if } j' = 1; \\ 0, & \text{otherwise.} \end{cases}$$

$$\text{for } j \in [m], j' \in [N]: \qquad \mathbf{u}_{0,j}[j'] = \begin{cases} \mathbf{L}_0[j], & \text{if } j' = 1; \\ 0, & \text{otherwise.} \end{cases}$$

$$\text{for } i \in [\ell], b \in \{0,1\}, j \in [m], j' \in [N]: \qquad \mathbf{u}_{i,b,j}[j'] = \begin{cases} \mathbf{L}_i^b[j], & \text{if } j' = 2i + b; \\ 0, & \text{otherwise.} \end{cases}$$

  It then generates IPFE ciphertexts

$$\begin{aligned} & \mathsf{ict}_f \xleftarrow{\$} \mathsf{IPFE.Enc}(\mathsf{impk}, [\![\mathbf{u}_f]\!]_1), \\ \text{for } j \in [m]: \quad & \mathsf{ict}_{0,j} \xleftarrow{\$} \mathsf{IPFE.Enc}(\mathsf{impk}, [\![\mathbf{u}_{0,j}]\!]_1), \\ \text{for } i \in [\ell], b \in \{0,1\}, j \in [m]: \quad & \mathsf{ict}_{i,b,j} \xleftarrow{\$} \mathsf{IPFE.Enc}(\mathsf{impk}, [\![\mathbf{u}_{i,b,j}]\!]_1),, \end{aligned}$$

  and sets

$$\mathsf{ct} = \big(\mathsf{ict}_f, \{\mathsf{ict}_{0,j}\}_{j\in[m]}, \{\mathsf{ict}_{i,b,j}\}_{i\in[\ell],b\in\{0,1\},j\in[m]}\big).$$

  The algorithm outputs ct as the ciphertext for μ with policy f.

  Note: *The vectors* $\mathbf{u}_f, \mathbf{u}_{0,j}$*'s put each component of* $L_f, \mathbf{L}_0$ *at the beginning so that they can always be "picked up" by the* $\mathbf{v}$*'s used in* KeyGen*. The vectors* $\mathbf{u}_{i,b,j}$*'s put each component of* $\mathbf{L}_i^b$ *at appropriate positions so that they can be "picked up" by* $\mathbf{v}$ *if and only if* $\mathbf{x}[i] = b$*. Since each component in the shares contributes one IPFE ciphertext and each IPFE ciphertext is of size polynomial in* λ, ℓ*, if the secret sharing scheme is succinct, then the ABE is ciphertext-succinct.*

- $\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}, \mathbf{x}, \mathsf{ct}, f)$ takes as input the master public key mpk, a secret key sk, its associated attribute $\mathbf{x}$, a ciphertext ct, and its associated policy f. If $P(\mathbf{x}, f) = 0$, the algorithm outputs $\perp$ and terminates. Otherwise, it parses $\mathsf{sk} = ([\![\delta]\!]_2, \mathsf{isk})$, recomputes the vector $\mathbf{v}$ for $\mathbf{x}$ as in KeyGen, and parses ct into ict_f, $\{\mathsf{ict}_{0,j}\}$ and $\{\mathsf{ict}_{i,b,j}\}$. It runs

$$
\begin{aligned}
 & (\gamma, 1^B) \leftarrow \mathsf{SS.FindCoef}(\mathsf{SS.pp}, f, \mathbf{x}),\\
 & \Lambda_f \leftarrow \mathsf{IPFE.Dec}(\mathsf{isk}, [\![\delta]\!]_2\mathbf{v}, \mathsf{ict}_f),\\
\text{for } j \in [m]: \quad & \Lambda_{0,j} \leftarrow \mathsf{IPFE.Dec}(\mathsf{isk}, [\![\delta]\!]_2\mathbf{v}, \mathsf{ict}_{0,j}),\\
\text{for } i \in [\ell], j \in [m]: \quad & \Lambda_{i,\mathbf{x}[i],j} \leftarrow \mathsf{IPFE.Dec}(\mathsf{isk}, [\![\delta]\!]_2\mathbf{v}, \mathsf{ict}_{i,\mathbf{x}[i],j}).
\end{aligned}
$$

  Let $\Lambda^{\mathbf{x}} = \big(\{\Lambda_{0,j}\}_{j\in[m]}, \{\Lambda_{i,\mathbf{x}[i],j}\}_{i\in[\ell],j\in[m]}\big)$. The algorithm applies the affine function γ homomorphically in the exponent of G_{T} to compute $\gamma(\Lambda_f, \Lambda^{\mathbf{x}})$. It then finds and outputs the unique $\mu' \in \{0,1\}$ (as the decrypted message) such that

$$\gamma(\Lambda_f, \Lambda^{\mathbf{x}}) = [\![\mu'\lfloor p/2\rceil + e]\!]_1[\![\delta]\!]_2,$$

  for some $e \in [-B..B]$, by enumerating over all possible e.

  Note: *We show that the scheme is correct. By definition and the correctness of IPFE, we have* $\Lambda_f = [\![\delta L_f]\!]_{\mathrm{T}}$*, and*

$$\Lambda_{0,j} = [\![\delta\mathbf{L}_0[j]]\!]_{\mathrm{T}}, \quad \Lambda_{i,\mathbf{x}[i],j} = [\![\delta\mathbf{L}_i^{\mathbf{x}[i]}[j]]\!]_{\mathrm{T}} \quad \Longrightarrow \quad \Lambda^{\mathbf{x}} = [\![\delta\mathbf{L}^{\mathbf{x}}]\!]_{\mathrm{T}},$$

  and therefore, $\gamma(\Lambda_f, \Lambda^{\mathbf{x}}) = [\![\delta\gamma(\mathbf{L}_f, \mathbf{L}^{\mathbf{x}})]\!]_{\mathrm{T}} = [\![\gamma(\mathbf{L}_f, \mathbf{L}^{\mathbf{x}})]\!]_1[\![\delta]\!]_2$. *By the correctness of the nearly linear secret sharing scheme, the decryption algorithm outputs the correct bit* $\mu' = \mu$.

Efficiency. By Construction 4, for input vector length $N = 2\ell + 1$, the IPFE components have bit lengths $|\mathsf{impk}| = \mathrm{poly}(\lambda)\cdot\ell$, $|\mathsf{imsk}| = \mathrm{poly}(\lambda)\cdot\ell$, $|\mathsf{ict}| = \mathrm{poly}(\lambda)\cdot\ell$, $|\mathsf{isk}| = \mathrm{poly}(\lambda)$. Also recall that the secret sharing components have bit lengths $|\mathsf{SS.pp}| = \mathrm{poly}(\lambda)\cdot\ell$, and $|\mathbf{L}_f| = \mathrm{poly}(\lambda)$. In the above construction,

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

- The master secret key consists of imsk, hence has bit length
  $|\mathsf{msk}| = |\mathsf{imsk}| = \mathrm{poly}(\lambda)\cdot\ell$.

- A secret key consists of a single isk, and $[\![\delta]\!]_2$ in G_2, hence has bit length
  $|\mathsf{sk}| = |\mathsf{isk}| + |G_2| = \mathrm{poly}(\lambda)$.

- A ciphertext consists of $1 + (2\ell+1)m$ ict's, hence has bit length
  $|\mathsf{ct}| = (1 + (2\ell+1)m)|\mathsf{ict}| = \mathrm{poly}(\lambda)\cdot\ell^2$.

We now state and prove (in Section 6.5) adaptive IND-CPA security of the scheme.

Proposition 23. *Suppose in Construction 5, the IPFE scheme is adaptively simulation-secure, and the secret sharing scheme is non-annihilable for* $\mathbf{L}_f$ *and* $\mathbf{L}^{\mathbf{x}}$*. Then the constructed CP-ABE scheme is adaptively IND-CPA in GGM. Specifically, for all efficient adversary* $\mathcal{A}$ *that distinguishes the adaptive ABE experiment with advantage* $\varepsilon_{\mathcal{A}}$*, there exist an efficient* $\mathcal{B}_1$

that distinguishes the adaptive simulation security experiments of IPFE with advantage $\varepsilon_{\mathsf{IPFE}}$, and an efficient $\mathcal{B}_2, \mathcal{B}_3$ that wins the non-annihilable games for $\mathbf{L}_f$ and $\mathbf{L}^{\mathbf{x}}$ with advantages $\varepsilon_{\text{ANN-f}}, \varepsilon_{\text{ANN-x}}$, such that:

$$\varepsilon_{\mathcal{A}} \leq \varepsilon_{\mathsf{IPFE}} + \frac{Q}{q} + TQ(\varepsilon_{\text{ANN-f}} + \varepsilon_{\text{ANN-x}}), \tag{14}$$

where T, Q are polynomial upper bounds on the number of zero-test queries and key queries from $\mathcal{A}$.

Combining Construction 5, Proposition 23, Construction 2, Proposition 12, Proposition 13, Construction 4, and Proposition 22, we have the following theorem:

Theorem 24 (Doubly Succinct CP-ABE). *Assume the polynomial hardness of adaptive LWE, with super-polynomial modulus-to-noise ratio. That is,* $\text{ALWE}_{n,q,\chi}$ *with* $\alpha = \frac{q}{B} = 2^{\omega(\log n)}$ *holds, where χ is a B-bounded error distribution.*

Let λ be the security parameter, and p a super-polynomially large prime bounded by $\mathrm{O}(\alpha \operatorname{poly}(\lambda))$. *In the generic pairing group model with order p, there exists a* doubly succinct *CP-ABE scheme for boolean formulae that*

- *satisfies the adaptive IND-CPA security, and*
- *has constant-size secret keys* $|\mathsf{sk}_{\mathbf{x}}| = \operatorname{poly}(\lambda)$ *(concretely, containing 2 group elements) and ciphertexts of size* $|\mathsf{ct}_f| = \operatorname{poly}(\lambda)\ell^2$, *where ℓ is the attribute length.*

Specifically, for all $\lambda \in \mathbb{N}$, for all efficient adversary $\mathcal{A}$ that distinguishes the adaptive ABE experiments with advantage $\varepsilon_{\mathcal{A}}$, and for all $k \in \mathbb{N}$, there exist an efficient $\mathcal{B}_1$ that distinguishes the selective simulation security experiments of IPFE with advantage $\varepsilon_{\mathsf{IPFE}}$, and an efficient $\mathcal{B}_2$ that distinguishes the adaptive LWE experiments with advantage $\varepsilon_{\text{ALWE}}$, such that

$$\varepsilon_{\mathcal{A}} \leq \varepsilon_{\mathsf{IPFE}} + \lambda^{-k} + \operatorname{poly}(|\mathcal{A}|)\varepsilon_{\text{ALWE}} + \frac{\operatorname{poly}(|\mathcal{A}|, \lambda)}{p}$$

Remark 4. In Construction 5, if the secret sharing scheme is only selectively non-annihilable for $\mathbf{L}_{\mathbf{x}}$ and $\mathbf{L}_f$, then the resultant CP-ABE achieves *very selective* security. This is because the adversary must commit to all the $\mathbf{x}_q$'s (for sk_q's) for the reduction to selective security of the secret sharing scheme to apply. The advantage relation in Proposition 23 holds, except $\varepsilon_{\mathcal{A}}, \varepsilon_{\text{ANN-f}}, \varepsilon_{\text{ANN-x}}$ are the advantages in the selective games.

6.5 Proof of Proposition 23

Proof (Proposition 23). Fix an an efficient adversary $\mathcal{A}$, we construct adversaries $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3$ against the selective simulation security of IPFE, and non-annihilability for $\mathbf{L}_f$ and $\mathbf{L}^{\mathbf{x}}$. We bound the advantage $\varepsilon_{\mathcal{A}}$ of $\mathcal{A}$ in the non-annihilability game as in (14) from the proposition statement.

Recall that in the adaptive security game of CP-ABE, the adversary first repeatedly queries attributes $\{\mathbf{x}_i\}$ and receives secret keys $\{\mathsf{sk}_i\}$ for them. It next adaptively chooses a policy f as its challenge, and receives a ciphertext ct for it. After the challenge, it again repeatedly chooses attributes $\{\mathbf{x}_i\}$ and receives secret keys $\{\mathsf{sk}_i\}$ for them. In the end, it outputs a bit b to distinguish whether the encrypted message μ in ct is 0 or 1.

Recall that the adaptive simulation-security of IPFE guarantees a simulator ($\mathsf{IPFE}.\widetilde{\mathsf{Setup}}$, $\mathsf{IPFE}.\widetilde{\mathsf{KeyGen}}$, $\mathsf{IPFE}.\widetilde{\mathsf{Enc}}$, $\mathsf{IPFE}.\widetilde{\mathsf{GrpZT}}$). We consider the following hybrids, and let A_i^{μ} be distinguishing advantages of $\mathcal{A}$ in H_i^{μ} for $i \in [3]$.

- H_1^{μ} is the adaptive security experiment $\mathsf{Exp}^{\mu}_{\mathrm{CPA}}$. Specifically, the challenger first runs

$$(\mathsf{impk}, \mathsf{imsk}) \xleftarrow{\$} \mathsf{IPFE.Setup}(1^{\lambda}, 1^{N}),$$

  and then computes each queried secret key for $\mathbf{x}_i$ as follows.

$$\mathsf{isk}_i \xleftarrow{\$} \mathsf{IPFE.KeyGen}(\mathsf{imsk}, [\![\delta_i \mathbf{v}_i]\!]_2), \qquad \mathsf{sk}_i = ([\![\delta_i]\!]_2, \mathsf{isk}_i)$$

  where δ_i is a fresh random non-zero element, and $\mathbf{v}_i$ is the selecting vectors for $\mathbf{x}_i$. The ciphertext for f is computed by first generating shares $(\mathbf{L}_0, \{\mathbf{L}_i^b\}), \mathbf{L}_f$ from running SS.ShareX and SS.ShareF, and then encoding the shares into vectors $\{\mathbf{u}_{0,j}\}, \{\mathbf{u}_{i,b,j}\}, \mathbf{u}_f$, as described in Enc. Finally, the ciphertext consists of icts encrypting those $\mathbf{u}$ vectors.

- H_2^{μ} Now the challenger simulates the GGM oracles for $\mathcal{A}$. The game proceeds similarly to H_1 except that we replace the IPFE scheme with the simulator. Specifically, the challenger first runs

$$(\widetilde{\mathsf{impk}}, \mathsf{st}) \xleftarrow{\$} \mathsf{IPFE.}\widetilde{\mathsf{Setup}}(1^{\lambda}, 1^{N}),$$

  and initialize a database of IPFE key vectors and inner-products $D = (\emptyset, \emptyset)$.

  When receiving a key query for $\mathbf{x}_i$, the challenger simulates the isk_i component using $\widetilde{\mathsf{KeyGen}}$:

$$(\widetilde{\mathsf{isk}}_i, \mathsf{st}') \xleftarrow{\$} \widetilde{\mathsf{KeyGen}}(\mathsf{st},) \quad \mathsf{sk}_i = ([\![\delta_i]\!]_2, \widetilde{\mathsf{isk}}_i).$$

  It updates $\mathsf{st} \leftarrow \mathsf{st}'$, and records the IPFE key vector $[\![\delta_i \mathbf{v}_i]\!]_2$ into the database D. If a ciphertext query for f is already received (hence the $\mathbf{u}$ vectors already created), then it computes all inner-products between $\mathbf{v}_i$ and the $\mathbf{u}$ vectors and update them into the databse D.

  When receiving a ciphertext query for f, the challenger simulates the ict components similarly by running $\widetilde{\mathsf{Enc}}(1^N, \mathsf{st})$. It also generate shares $\mathbf{L}_0, \{\mathbf{L}_i^b\}, L_f$ from running SS.ShareX and SS.ShareF, and then the $\mathbf{u}$ vectors encoding these shares. It finally computes inner-products between all $\mathbf{u}$ vectors and stored $\mathbf{v}_i$ vectors, and update them into the database D.

  When receiving a zero-test query γ, the challenger runs

$$\mathsf{st}', z \leftarrow \widetilde{\mathsf{GrpZT}}^{\mathsf{InPrdZT}(D,\cdot)}(\mathsf{st}, \gamma),$$

  where the InPrdZT algorithm answers zero-test queries from $\widetilde{\mathsf{GrpZT}}$ over keys and inner products stored in D. It updates $\mathsf{st} \leftarrow \mathsf{st}'$, and sends the answer z back to $\mathcal{A}$. That is, the zero-test query γ is translated by $\widetilde{\mathsf{GrpZT}}$ into zero-tests over values in D, and the challenger simply answer the translated queries by looking at values in D.

  Claim 25. In GGM, there exists an efficient $\mathcal{B}_1$ that distinguishes the adaptive simulation experiments with advantage $\varepsilon_{\mathsf{IPFE}}$ such that $|A_1^{\mu} - A_2^{\mu}| \leq \varepsilon_{\mathsf{IPFE}}$.

- In H_3^{μ}, the challenger does not run the secret sharing algorithms anymore. Recall that in H_2^{μ}, the challenger needs to run SS.ShareX, SS.ShareF to compute the $\mathbf{u}$ vectors, and eventually inner-products between $\mathbf{v}_i$ and the $\mathbf{u}$ vectors. By construction, the inner-products are exactly $[\![\delta_i \mathbf{L}^{\mathbf{x}_i}]\!]_{\mathrm{T}}$ and $[\![\mathbf{L}_f]\!]_{\mathrm{T}}$. In H_3^{μ}, the challenger instead obtain them by interacting (as $\mathcal{A}'$) with the $\mathsf{Exp}^{\mu}_{\mathrm{CP}}$ experiment, where $\mathsf{Exp}^{\mu}_{\mathrm{CP}}(1^{\lambda})$ with a machine $\mathcal{A}'$ proceeds as follows:

- **Setup.** Launch $\mathcal{A}'(1^\lambda)$ and receive from it the input length 1^ℓ. First, run $\mathsf{SS.Setup}(1^\lambda, 1^\ell)$ to generate pp and send it to $\mathcal{A}'$. Next, run $\mathsf{SS.Share}(\mathsf{pp})$ to generate $(\mathbf{L}_0, \{\mathbf{L}_k^b\}_{k\in[\ell]}^{b\in\{0,1\}}, r)$.
- **Query I.** Repeat the following for arbitrarily many rounds determined by $\mathcal{A}'$: In each round, $\mathcal{A}'$ submit some $\mathbf{x}_i \in \{0,1\}^\ell$. Upon this query, sample a random element $[\![\delta_i]\!]_2$ encoded in G_2 and send $([\![\delta_i \mathbf{L}^{\mathbf{x}_i}]\!]_2, [\![\delta_i]\!]_2)$ to $\mathcal{A}'$.
- **Challenge.** $\mathcal{A}'$ submits some $f^* \in \mathcal{F}_{\lambda,\ell}$. Run $\mathbf{L}_{f^*} \xleftarrow{\$} \mathsf{SS.ShareF}(\mathsf{pp}, f^*, \mu, r)$ and return $[\![\mathbf{L}_{f^*}]\!]_1$ to $\mathcal{A}'$.
- **Query II.** Same as Query I.
- **Guess.** $\mathcal{A}'$ outputs a bit μ'. The outcome of the experiment is μ' if $f^*(\mathbf{x}_i) = 0$ for all x_i queried in Query I/II. Otherwise, the outcome is set to 0.

$|A_2^\mu - A_3^\mu| = 0$ as H_3^μ is the same as H_2^μ by construction. We show the following claim:

Claim 26. In GGM, there exists efficient $\mathcal{B}_2, \mathcal{B}_3$ that wins the non-annihilability games for $\mathbf{L}_f, \mathbf{L}_x$ with advantages $\varepsilon_{\text{ANN-f}}, \varepsilon_{\text{ANN-x}}$, such that

$$|A_3^\mu| \le \frac{Q}{p} + TQ(\varepsilon_{\text{ANN-f}} + \varepsilon_{\text{ANN-x}}),$$

where T, Q are polynomial upper bounds on the number of zero-test queries and key queries from $\mathcal{A}$.

$\square$

Proof (Claim 25). We prove this claim by reduction to the adaptive simulation security of the IPFE scheme in GGM (see Definition 19). We construct a distinguisher $\mathcal{B}$ for the adaptive simulation games $\mathsf{Exp}_{\text{real}}$ and $\mathsf{Exp}_{\text{sim}}$ as follows:

1. $\mathcal{B}$ launches $\mathcal{A}(1^\lambda)$ with fresh randomness $r_\mathcal{A}$, and receives from it a predicate $P \in \mathcal{P}_\lambda$. Let ℓ be the attribute length for P and let $N = (2\ell + 1)$. The distinguisher $\mathcal{B}$ sends 1^N to the challenger.

2. $\mathcal{B}$ receives impk from the challenger, and runs

$$\mathsf{SS.pp} \xleftarrow{\$} \mathsf{SS.Setup}(1^\lambda, 1^\ell), \quad \left(\mathbf{L}_0, \{\mathbf{L}_i^b\}_i^b, r\right) \xleftarrow{\$} \mathsf{SS.ShareX}(\mathsf{SS.pp}).$$

   $\mathcal{B}$ sends $\mathsf{mpk} = (\mathsf{SS.pp}, \mathsf{impk})$ to the adversary $\mathcal{A}$.

3. Upon receiving a query $\mathbf{x}_i \in \{0,1\}^\ell$ from $\mathcal{A}$, the distinguisher $\mathcal{B}$ samples a random non-zero element δ_i, and computes the selecting vectors $\mathbf{v}_i$ for $\mathbf{x}_i$. It then sends $[\![\delta_i \mathbf{v}_i]\!]_2$ to the challenger, and receives isk_i back. Finally, $\mathcal{B}$ answers $\mathcal{A}$ with $\mathsf{sk}_i = ([\![\delta_i]\!]_2, \mathsf{isk}_i)$.

4. Upon receiving the challenge f, the distinguisher $\mathcal{B}$ runs $\mathsf{SS.ShareF}(\mathsf{SS.pp}, f, \mu, r)$ to generate L_f, and then encode $\mathbf{L}_0, \{\mathbf{L}_i^b\}_i^b, L_f$ into the $\mathbf{u}$ vectors as in Enc. It sends the $\mathbf{u}$ vectors in G_1 to the challenger, and gets back icts. It answers $\mathcal{A}$ with ct consisting of those received icts.

5. Upon receiving a zero-test query, the distinguisher $\mathcal{B}$ forwards it to the challenger, and return the answer from the challenger directly to $\mathcal{A}$.

6. In the end, $\mathcal{B}$ receives an output bit b from $\mathcal{A}$. It outputs 1 if and only if $b = \mu$.

First note that $\mathcal{B}$ is efficient. In $\mathsf{Exp}_{\text{real}}$, the distinguisher $\mathcal{B}$ emulates H_1^μ for $\mathcal{A}$, and outputs whether $\mathcal{A}$'s output matches μ. In $\mathsf{Exp}_{\text{sim}}$, it does so for H_2^μ. Therefore,

$$|A_1^\mu - A_2^\mu| = \left|\Pr[\mathcal{B} \to 1 \text{ in } \mathsf{Exp}_{\text{real}}] - \Pr[\mathcal{B} \to 1 \text{ in } \mathsf{Exp}_{\text{sim}}]\right|$$

□

Proof (Claim 26). Let $\mathcal{A}'$ be the challenger in H_3. We consider the following hybrids, and let $A_i'^\mu$ be distinguishing advantages of $\mathcal{A}'$ in G_i^μ for $i \in [3]$. (Note that $A_1'^\mu = A_3^\mu$.)

- G_1^μ is the experiment $\mathsf{Exp}_{\text{CP}}^\mu$. Specifically, throughout the experiment, the GGM oracle answers zero-test queries from the adversary of the form

$$\gamma\big((1, \mathbf{L}_{f^*}) \otimes (1, \{\delta_i \mathbf{L}^{\mathbf{x}_i}\}_i, \{\delta_i\}_i)\big).$$

- G_2^μ proceeds identically as G_1^μ, except that the challenger views each zero-test query from $\mathcal{A}'$ as a degree-1 polynomial where δ_i are the variables:

$$\gamma\big((1, \mathbf{L}_{f^*}) \otimes (1, \{\delta_i \mathbf{L}^{\mathbf{x}_i}\}_i, \{\delta_i\}_i)\big) = \sum_i \gamma_i\big((1, \mathbf{L}_{f^*}) \otimes (1, \mathbf{L}^{\mathbf{x}_i})\big)\delta_i + \gamma_0(\mathbf{L}_{f^*})$$

  The challenger answers the query with zero if and only if γ_i evaluates to zero for all i.

  Let Q be the maximum number of zero-test queries from $\mathcal{A}'$. Since δ_i are sampled independently at random, by Schwartz–Zippel lemma of degree-1 we have $|A_1'^\mu - A_2'^\mu| \le \frac{Q}{p}$.

- G_3^μ proceeds identically as G_2^μ, except that the challenger answers zero-test queries with zero if and only all γ_i are the zero function.

  Note that in GGM, $\mathcal{A}'$ only gains information through zero-test queries. In G_3, all queries are answered independently of the shares. Hence $\mathcal{A}'$ has zero advantage, i.e., $A_3'^\mu = 0$. It remains to construct efficient $\mathcal{B}_2$ and $\mathcal{B}_3$ that wins the non-annihilability games for $\mathbf{L}_f, \mathbf{L}_x$ with advantages $\varepsilon_{\text{ANN-f}}, \varepsilon_{\text{ANN-x}}$, such that

$$|A_2'^\mu - A_3'^\mu| \le TQ(\varepsilon_{\text{ANN-f}} + \varepsilon_{\text{ANN-x}}),$$

  where T, Q are polynomial upper bounds on the number of zero-test queries and key queries (for $\mathbf{x}_i$) from $\mathcal{A}'$.

Note that G_2^μ and G_3^μ differs if and only if there exists i, j such that when viewing the j^{th} zero-test query from $\mathcal{A}'$ as a polynomial over the δ's, the affine function γ_i is non-zero, but evaluates to zero. We also distinguish two cases:

- Case 1: when plugging in $\mathbf{L}^{\mathbf{x}_i}$, γ_i is the zero function over $\mathbf{L}_f$.

- Case 2: when plugging in $\mathbf{L}^{\mathbf{x}_i}$, γ_i is a non-zero function over $\mathbf{L}_f$ (but evaluates to 0).

Let $E_{i,j}$ denote such an event for i, j, and let E be the union of all $E_{i,j}$. Let T, Q be polynomial upper bounds on the number of zero-test queries and key queries (for $\mathbf{x}_i$) from $\mathcal{A}'$. We construct efficient adversaries $\mathcal{B}_d$ for $d = 2, 3$ against the non-annihilability games for $\mathbf{L}_f, \mathbf{L}_x$ as follows

- $\mathcal{B}_d$ samples $t \xleftarrow{\$} [T]$ and $q \xleftarrow{\$} [Q]$, and then launches $\mathcal{A}'(1^\lambda)$.
- $\mathcal{B}_d$ receives the input length 1^ℓ from $\mathcal{A}'$, and forwards 1^ℓ to the challenger. $\mathcal{B}_d$ next receives $\mathsf{SS.pp}$ from the challenger, and forwards it to $\mathcal{A}'$.
- Upon receiving a query $\mathbf{x}_i$, the adversary $\mathcal{B}_d$ answers with new group handles in G_2.
- Upon receiving the challenge f^*, the adversary $\mathcal{B}_d$ answers with new group handles in G_1.
- Upon receiving the j^{th} zero-test query, if $j < q$ then $\mathcal{B}_d$ answers as in G_3^μ.

  If $j = q$, but $\mathcal{A}$ has not submitted the challenge f^*:

  - if $d = 2$, then $\mathcal{B}_d$ aborts.
  - if $d = 3$, then $\mathcal{B}_d$ sends $\mathbf{x}_t$ to the challenger, and outputs γ_t.

  If $j = q$, and $\mathcal{A}'$ has submitted the challenge f^*, then $\mathcal{B}_d$ sends $\mathbf{x}_t$ to the challenger.

  - If $d = 2$, then $\mathcal{B}_2$ receives $\mathbf{L}^{\mathbf{x}}$. It plugs $\mathbf{L}^{\mathbf{x}}$ into γ_t and outputs the resulting affine function γ_t' over $\mathbf{L}_f$.
  - if $d = 3$, then $\mathcal{B}_3$ view γ_t as an affine function γ_t' over $\mathbf{L}_f$, and outputs the first non-zero coefficient (as an affine function over $\mathbf{x}_t$).

Note that if E happens, and $\mathcal{B}_d$ samples t, q exactly equal to the smallest i, j such that $E_{i,j}$ happens, then either Case 1 or Case 2 happens. In Case 1, $\mathcal{B}_3$ wins the non-annihilability game for $\mathbf{L}^{\mathbf{x}}$, while in Case 2, $\mathcal{B}_2$ wins the non-annihilability game for $\mathbf{L}_f$. That is:

$$|A_2'^\mu - A_3'^\mu| \le \Pr[E] \le TQ \cdot \left(\Pr[\mathcal{B}_2 \text{ wins in } \mathsf{Exp}_{\mathrm{ANN}}] + \Pr[\mathcal{B}_3 \text{ wins in } \mathsf{Exp}'_{\mathrm{ANN}}]\right).$$

□

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.

References

[ABDP15] Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. Simple functional encryption schemes for inner products. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 733–751. Springer, Heidelberg, March / April 2015.

[ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 595–618. Springer, Heidelberg, August 2009.

[ALdP11] Nuttapong Attrapadung, Benoît Libert, and Elie de Panafieu. Expressive key-policy attribute-based encryption with constant-size ciphertexts. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011*, volume 6571 of *LNCS*, pages 90–108. Springer, Heidelberg, March 2011.

[ALS16] Shweta Agrawal, Benoît Libert, and Damien Stehlé. Fully secure functional encryption for inner products, from standard assumptions. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 333–362. Springer, Heidelberg, August 2016.

[AT20] Nuttapong Attrapadung and Junichi Tomida. Unbounded dynamic predicate compositions in ABE from standard assumptions. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part III*, volume 12493 of *LNCS*, pages 405–436. Springer, Heidelberg, December 2020.

[Att16] Nuttapong Attrapadung. Dual system encryption framework in prime-order groups via computational pair encodings. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 591–623. Springer, Heidelberg, December 2016.

[AWY20] Shweta Agrawal, Daniel Wichs, and Shota Yamada. Optimal broadcast encryption from LWE and pairings in the standard model. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part I*, volume 12550 of *LNCS*, pages 149–178. Springer, Heidelberg, November 2020.

[AY20] Shweta Agrawal and Shota Yamada. Optimal broadcast encryption from pairings and LWE. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 13–43. Springer, Heidelberg, May 2020.

[Bar86] David A. Mix Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. In *18th ACM STOC*, pages 1–5. ACM Press, May 1986.

[BDHM92] Gerhard Buntrock, Carsten Damm, Ulrich Hertrampf, and Christoph Meinel. Structure and importance of logspace-MOD class. *Mathematical Systems Theory*, 25(3):223–237, 1992.

[Bei96] Amos Beimel. *Secure Schemes for Secret Sharing and Key Distribution*. PhD thesis, Technion–Israel Institute of Technology, 1996.

[Ber84] Stuart J. Berkowitz. On computing the determinant in small parallel time using a small number of processors. *Information Processing Letters*, 18(3):147–150, 1984.

[BGG$^+$14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 533–556. Springer, Heidelberg, May 2014.

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

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 253–273. Springer, Heidelberg, March 2011.

[BTVW17] Zvika Brakerski, Rotem Tsabary, Vinod Vaikuntanathan, and Hoeteck Wee. Private constrained PRFs (and more) from LWE. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 264–302. Springer, Heidelberg, November 2017.

[BV15] Zvika Brakerski and Vinod Vaikuntanathan. Constrained key-homomorphic PRFs from standard lattice assumptions - or: How to secretly embed a circuit in your PRF. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 1–30. Springer, Heidelberg, March 2015.

[CCL18] Yi-Hsiu Chen, Kai-Min Chung, and Jyun-Jie Liao. On the complexity of simulating auxiliary input. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 371–390. Springer, Heidelberg, April / May 2018.

[GGH$^+$13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, October 2013.

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 467–476. ACM Press, June 2013.

[GKP$^+$13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. How to run turing machines on encrypted data. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 536–553. Springer, Heidelberg, August 2013.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 89–98. ACM Press, October / November 2006. Available as Cryptology ePrint Archive Report 2006/309.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 75–92. Springer, Heidelberg, August 2013.

[GV15] Sergey Gorbunov and Dhinakaran Vinayagamurthy. Riding on asymmetry: Efficient ABE for branching programs. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 550–574. Springer, Heidelberg, November / December 2015.

[GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 162–179. Springer, Heidelberg, August 2012.

[GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 545–554. ACM Press, June 2013.

[JP14] Dimitar Jetchev and Krzysztof Pietrzak. How to fake auxiliary input. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 566–590. Springer, Heidelberg, February 2014.

[KNTY19] Fuyuki Kitagawa, Ryo Nishimaki, Keisuke Tanaka, and Takashi Yamakawa. Adaptively secure and succinct functional encryption: Improving security and efficiency, simultaneously. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 521–551. Springer, Heidelberg, August 2019.

[KW93] Mauricio Karchmer and Avi Wigderson. On span programs. In *Proceedings of Structures in Complexity Theory*, pages 102–111, 1993.

[KW19] Lucas Kowalczyk and Hoeteck Wee. Compact adaptively secure ABE for NC^1 from k-Lin. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 3–33. Springer, Heidelberg, May 2019.

[LL20] Huijia Lin and Ji Luo. Succinct and adaptively secure ABE for ABP from k-lin. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part III*, volume 12493 of *LNCS*, pages 437–466. Springer, Heidelberg, December 2020.

[LOS$^+$10] Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 62–91. Springer, Heidelberg, May / June 2010.

[LW12] Allison B. Lewko and Brent Waters. New proof methods for attribute-based encryption: Achieving full security through selective techniques. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 180–198. Springer, Heidelberg, August 2012.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 700–718. Springer, Heidelberg, April 2012.

[Mul87] Ketan Mulmuley. A fast parallel algorithm to compute the rank of a matrix over an arbitrary field. *Combinatorica*, 7(1):101–104, 1987.

[OT10] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 191–208. Springer, Heidelberg, August 2010.

[Ps16] Rafael Pass and abhi shelat. Impossibility of VBB obfuscation with ideal constant-degree graded encodings. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part I*, volume 9562 of *LNCS*, pages 3–17. Springer, Heidelberg, January 2016.

[QWW18] Willy Quach, Hoeteck Wee, and Daniel Wichs. Laconic function evaluation and applications. In Mikkel Thorup, editor, *59th FOCS*, pages 859–870. IEEE Computer Society Press, October 2018.

[SW05] Amit Sahai and Brent R. Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 457–473. Springer, Heidelberg, May 2005.

[Tak14] Katsuyuki Takashima. Expressive attribute-based encryption with constant-size ciphertexts from the decisional linear assumption. In Michel Abdalla and Roberto De Prisco, editors, *SCN 14*, volume 8642 of *LNCS*, pages 298–317. Springer, Heidelberg, September 2014.

[Tsa19] Rotem Tsabary. Fully secure attribute-based encryption for t-CNF from LWE. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 62–85. Springer, Heidelberg, August 2019.

[Wat15] Brent Waters. A punctured programming approach to adaptively secure functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 678–697. Springer, Heidelberg, August 2015.

[Wee17] Hoeteck Wee. Attribute-hiding predicate encryption in bilinear groups, revisited. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 206–233. Springer, Heidelberg, November 2017.

[YAHK14] Shota Yamada, Nuttapong Attrapadung, Goichiro Hanaoka, and Noboru Kunihiro. A framework and compact constructions for non-monotonic attribute-based encryption. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 275–292. Springer, Heidelberg, March 2014.

[ZGT+16] Kai Zhang, Junqing Gong, Shaohua Tang, Jie Chen, Xiangxue Li, Haifeng Qian, and Zhenfu Cao. Practical and efficient attribute-based encryption with constant-size ciphertexts in outsourced verifiable computation. In Xiaofeng Chen, XiaoFeng Wang, and Xinyi Huang, editors, *ASIACCS 16*, pages 269–279. ACM Press, May / June 2016.