Attribute-Based Encryption for Arithmetic Circuits

Dan Boneh[*] Valeria Nikolaenko[†] Gil Segev[‡]

Abstract

We present an Attribute Based Encryption system where access policies are expressed as polynomial size *arithmetic* circuits. We prove security against arbitrary collusions of users based on the learning with errors problem on integer lattices. The system has two additional useful properties: first, it naturally handles arithmetic circuits with arbitrary fan-in (and fan-out) gates. Second, secret keys are much shorter than in previous schemes: secret key size is proportional to the *depth* of the circuit where as in previous constructions the key size was proportional to the number of gates or wires in the circuit. The system is well suited for environments where access policies are naturally expressed as arithmetic circuits as is the case when policies capture statistical properties of the data or depend on arithmetic transformations of the data. The system also provides complete key delegation capabilities.

1 Introduction

(Key-policy) attribute-based encryption [SW05, GPS+06] is a public-key encryption mechanism where every secret key sk_f is associated with some function $f : \mathcal{X} \to \mathcal{Y}$ and an encryption of a message μ is labeled with a public attribute vector $x \in \mathcal{X}$. The encryption of μ can be decrypted using sk_f only if $f(x) = 0 \in \mathcal{Y}$. We review the syntax and security requirements for an attribute-based system in the next section. Intuitively, the security requirement is collusion resistance: a coalition of users learns nothing about the plaintext message μ if none of their keys are authorized to decrypt the ciphertext.

Attribute-based encryption (ABE) is a generalization of identity-based encryption [Sha84, BF03, Coc01] and fuzzy IBE [SW05, ABV+12] and is a special case of functional encryption [BSW11]. It is used as a building-block in applications that demand complex access control to encrypted data [PTM+06], in designing cryptographic protocols for verifiably outsourcing computations [PRV12], and for single-use functional encryption [GKP+13]. Here we focus on key-policy ABE where the access policy is embedded in the secret key. The dual notion called ciphertext-policy ABE can be realized from this using universal circuits, as explained in [GPS+06, GGH+13b].

The past few years have seen much progress in constructing secure and efficient attribute-based encryption from different assumptions and for different settings. The first constructions [GPS+06, LOS+10, OT10, LW12, Wat12, Boy13, HW13] apply to predicates computable by boolean formulas which are a subclass of log-space computations. More recently important progress has been made on constructions for the set of all polynomial size circuits: Gorbunov, Vaikuntanathan, and Wee [GVW13] gave a construction from the Learning With Errors (LWE) problem and Garg, Gentry, Halevi, Sahai, and Waters [GGH+13b] gave a construction using multilinear maps. In both constructions the circuits are represented as boolean circuits with fan-in 2.

[*]Stanford University, Stanford, CA 94305, USA. Email: `dabo@cs.stanford.edu`.

[†]Stanford University, Stanford, CA 94305, USA. Email: `valerini@stanford.edu`.

[‡]School of Computer Science and Engineering, Hebrew University of Jerusalem, Jerusalem 91904, Israel. Email: `segev@cs.huji.ac.il`. This work was done while the author was visiting Stanford University.

Our results. Inspired by the work of Gorbunov, Vaikuntanathan, and Wee [GVW13] we construct an LWE-based ABE for functions that are represented as polynomial-size *arithmetic circuits*. Moreover, the system naturally handles circuits with arbitrary fan-in gates. The system is considerably more efficient for ABE access policies that are naturally represented as arithmetic circuits. An interesting aspect of the construction is that secret-keys are much shorter than in previous constructions. Secret keys in previous ABE constructions contained an element (such as a matrix) for every gate or wire in the circuit. In our scheme the secret key is a single matrix corresponding only to the final output wire from the circuit. Consequently, the size of our secret keys is a function of the circuit depth, but not its size. We prove *selective* security of the system and observe that by a standard complexity leveraging argument as in [BB11] the system can be made adaptively secure.

Let q be a prime. Our base system handles arithmetic circuits with weighted addition and multiplication gates over $\mathbb{Z}_q$, namely gates of the form

$$g_+(x_1,\ldots,x_k) = \alpha_1 x_1 + \ldots + \alpha_k x_k \qquad \text{and} \qquad g_\times(x_1,\ldots,x_k) = \alpha \cdot x_1 \cdots x_k \tag{1.1}$$

where the weights α_i can be arbitrary elements in $\mathbb{Z}_q$. Addition gates g_+ take arbitrary inputs $x_1,\ldots,x_k \in \mathbb{Z}_q$. However, for multiplication gates $g_\times$, we require that the inputs are somewhat smaller than q, namely in the range $[-p,p]$ for some $p < q$. (In fact, our construction allows for one of the inputs to $g_\times$ to be arbitrarily large in $\mathbb{Z}_q$). We discuss the relation between p and q at the end of the section. Hence, while $f : \mathbb{Z}_q^\ell \to \mathbb{Z}_q$ can be an arbitrary polynomial-size arithmetic circuit, decryption will succeed only for attribute vectors $\mathbf{x}$ for which $f(\mathbf{x}) = 0$ and the inputs to all multiplication gates in the circuit are in $[-p,p]$.

The construction naturally handles access policies expressed as boolean circuits with large fan-in. An AND gate is implemented as $\wedge(x_1,\ldots,x_k) = x_1 \cdots x_k$ and an OR gate is implemented as $\vee(x_1,\ldots,x_k) = 1-(1-x_1)\cdots(1-x_k)$. The OR gate is a simple depth three arithmetic circuit using a single k-way multiplication gate. While ABE for boolean circuits is the settings of [GVW13, GGH+13b], our new system has considerably shorter secret keys and reduces the circuit size and depth by allowing for high fan-in gates which in-turn allows for smaller overall parameters.

Applications. Before discussing how the new system works, we first describe a few applications to illustrate the benefit of arithmetic circuits in the context of ABE.

Natural examples of arithmetic access policies come from linear algebra, graph algorithms, and statistics. For example, suppose an attribute vector corresponds to a vector in $[-B,B]^k$. A key sk_w should decrypt a ciphertext only if the associated vector has ℓ_2 norm equal to w. Using $q \gg kB^2$ this policy can be expressed as an arithmetic circuit of depth 4 and directly implemented in our system. As another example, suppose the attribute vector is an adjacency matrix of a directed graph and a key should decrypt a ciphertext only if in the associated graph there are exactly 10 paths from some specific node s to some other node t. Again, this can easily be expressed as an arithmetic circuit since counting the number of paths from s to t amounts to raising the adjacency matrix to a certain power.

To use an ABE built for boolean circuits (e.g.[GVW13, GGH+13b]) for these tasks we would need to implement the arithmetic circuits as a boolean circuit with gates of fan-in 2. The depth of the resulting circuits would be considerably larger than the depth of the given arithmetic circuit. In [GVW13] the efficiency of the system degrades polynomially with the depth of the circuit so that the resulting ABE would be less efficient than a direct implementation as an arithmetic circuit.

Key delegation. In Section 6 we show that our system supports key delegation. Using the master secret, user Alice can be given a secret key sk_f for a function f that lets her decrypt whenever the attribute vector $\mathbf{x}$ satisfies $f(\mathbf{x}) = 0$. In our system, for any function g, Alice can then issue a delegated secret key $\mathsf{sk}_{f \wedge g}$ to Bob that lets Bob decrypt whenever the attribute vector $\mathbf{x}$ satisfies $f(\mathbf{x}) = g(\mathbf{x}) = 0$. Bob can further delegate to Charlie, and so on. The size of the secret key increases quadratically with the number of delegations.

We note that Gorbunov et al. [GVW13] showed that their ABE system for Boolean circuits supports a somewhat restricted form of delegation. Specifically, they demonstrated that using a secret key sk_f for a function f, and a secret key sk_g for a function g, it is possible to issue a secret key $\mathsf{sk}_{f \wedge g}$ for the function $f \wedge g$. In this light, our work resolves the naturally arising open problem of providing full delegation capabilities (i.e., issuing $\mathsf{sk}_{f \wedge g}$ using only sk_f).

1.1 Building an ABE for arithmetic circuits

Key-homomorphic public-key encryption. We obtain our ABE by constructing a public-key encryption scheme that supports computations on public keys. More precisely, public keys in our system are tuples $\mathbf{x}$ over $\mathbb{Z}_q$ and any such tuple is a public-key. Anyone can encrypt a message μ under any public key $\mathbf{x}$. Now, let $\mathbf{x}$ be a tuple in $\mathbb{Z}_q^\ell$ for some ℓ and let $f : \mathbb{Z}_q^\ell \to \mathbb{Z}_q$ be a function represented as a polynomial-size arithmetic circuit. Key-homomorphism means that:

anyone can transform an encryption under key $\mathbf{x}$ into an encryption under key $f(\mathbf{x})$.

In other words, suppose c is an encryption of message μ under public-key $\mathbf{x} \in \mathbb{Z}_q^\ell$. There is a public algorithm $\mathsf{Eval}_{\mathrm{ct}}(f, \mathbf{x}, c) \longrightarrow c_f$ that outputs a ciphertext c_f that is an encryption of μ under the key $f(\mathbf{x}) \in \mathbb{Z}_q$. In our constructions $\mathsf{Eval}_{\mathrm{ct}}$ is deterministic and its running time is proportional to the size of the arithmetic circuit for f.

If we give user Alice the secret-key for the public-key $0 \in \mathbb{Z}_q$ then Alice can use $\mathsf{Eval}_{\mathrm{ct}}$ to decrypt c whenever $f(\mathbf{x}) = 0$, as required for ABE. Unfortunately, this ABE would be insecure because the secret key is not bound to the function f: Alice could decrypt any ciphertext encrypted under $\mathbf{x}$ by simply finding some function g such that $g(\mathbf{x}) = 0$.

To construct a secure ABE we slightly extend the basic key-homomorphism idea. We treat public keys as tuples over $(\mathbb{Z}_q \times \mathsf{PKT})$ where PKT is a finite set of public-key tags. That is, a public-key is a tuple $\mathsf{pk} = \big((x_1, \mathsf{pkt}_1), \ldots, (x_\ell, \mathsf{pkt}_\ell) \big)$ for some ℓ and any such tuple is a public key. The encryptor can encrypt a message μ to any such public key to obtain a ciphertext c. Key-homomorphism is now described as follows:

Suppose c is an encryption of μ under public-key $\mathsf{pk} = \big((x_i, \mathsf{pkt}_i) \big)_{i=1}^{\ell}$.
Then *anyone* can transform c to an encryption of μ under the public-key $\big(f(x_1, \ldots, x_\ell),\ \mathsf{pkt}_f\big)$.

Here pkt_f is a tag that is *uniquely* determined by the function f and the input tags $\mathsf{pkt}_1, \ldots, \mathsf{pkt}_\ell$. This pkt_f is efficiently computable by a *deterministic* algorithm called $\mathsf{Eval}_{\mathrm{pk}}$: given f and tags $(\mathsf{pkt}_1, \ldots, \mathsf{pkt}_\ell)$ the algorithm outputs pkt_f. Transforming the ciphertext c from one public-key to another is done using algorithm $\mathsf{Eval}_{\mathrm{ct}}(f, \mathsf{pk}, c) \longrightarrow c_f$ as before. The precise syntax and security requirements for key-homomorphic public-key encryption are provided in Appendix A.

To build an ABE we simply publish random tags $(\mathsf{pkt}_1^*, \ldots, \mathsf{pkt}_\ell^*)$ in the ABE's public parameters. To encrypt a message μ with attribute vector $\mathbf{x} = (x_1, \ldots, x_\ell) \in \mathbb{Z}_q^\ell$ the encryptor encrypts μ under the public-key $\mathsf{pk} = \big((x_i, \mathsf{pkt}_i^*) \big)_{i=1}^{\ell}$. Let c be the resulting ciphertext. Given an arithmetic circuit f, the key-homomorphic property lets anyone transform c into an encryption of μ under key $\big(f(\mathbf{x}),\ \mathsf{pkt}_f\big)$ where pkt_f is the output of $\mathsf{Eval}_{\mathrm{pk}}\big(f,\ (\mathsf{pkt}_1^*, \ldots, \mathsf{pkt}_\ell^*)\big)$. The point is that now the secret key for the function f can simply be set to the decryption key for the public-key $(0, \mathsf{pkt}_f)$. This key enables the decryption of c when $f(\mathbf{x}) = 0$ as follows: the decryptor first uses $\mathsf{Eval}_{\mathrm{ct}}(f, \mathsf{pk}, c) \longrightarrow c_f$ to transform the ciphertext to the public key $(f(\mathbf{x}), \mathsf{pkt}_f)$. It can then decrypt c_f using the decryption key it was given whenever $f(\mathbf{x}) = 0$. We will show that this results in a secure ABE.

Note that it is important that pkt_f be uniquely determined by f and $(\mathsf{pkt}_1^*, \ldots, \mathsf{pkt}_\ell^*)$ so that the authority generating secret keys arrive at the same pkt_f as the decryptor.

A construction from learning with errors. Fix some $n \in \mathbb{Z}^+$, prime q, and $m = \Theta(n \log q)$. Let $\mathbf{A}$ and $\mathbf{G}$ be matrices in $\mathbb{Z}_q^{n \times m}$ that will be part of the system parameters. The set PKT of public-key tags we use is the set $\mathbb{Z}_q^{n \times m}$ of all $n \times m$ matrices. Then a public-key is a tuple $\mathsf{pk} = \big((x_1, \mathbf{B}_1), \ldots, (x_\ell, \mathbf{B}_\ell) \big)$ where $\mathbf{x} = (x_1, \ldots, x_\ell) \in \mathbb{Z}_q^\ell$ and $\mathbf{B}_1, \ldots, \mathbf{B}_\ell \in \mathbb{Z}_q^{n \times m}$. To encrypt a message μ under the public key pk we use a variant of dual Regev encryption [Reg05, GPV08] using the following matrix as the public key:

$$\big(\mathbf{A} \mid x_1\mathbf{G} + \mathbf{B}_1 \mid \cdots \mid x_\ell\mathbf{G} + \mathbf{B}_\ell\big) \in \mathbb{Z}_q^{n \times (\ell+1)m} \tag{1.2}$$

We obtain a ciphertext $\mathbf{c}$. We note that this encryption algorithm is similar to encryption in the hierarchical IBE system of [ABB10] and to encryption in the predicate encryption for inner-products of [AFV11].

We show that, remarkably, this system is key-homomorphic: given a function $f : \mathbb{Z}_q^\ell \to \mathbb{Z}_q$ computed by a poly-size arithmetic circuit, anyone can transform the ciphertext $\mathbf{c}$ into a dual Regev encryption for the public-key matrix

$$\big(\mathbf{A} \mid f(\mathbf{x}) \cdot \mathbf{G} + \mathbf{B}_f\big) \in \mathbb{Z}_q^{n \times 2m}$$

where $\mathbf{B}_f \in \mathbb{Z}_q^{n \times m}$ is uniquely determined by f and $\mathbf{B}_1, \ldots, \mathbf{B}_\ell$. The work needed to compute $\mathbf{B}_f$ is proportional to the size of the arithmetic circuit for f. We ensure that $\mathbf{B}_f$ is uniquely determined (as necessary for correctness of the scheme) by relying on the deterministic nature of Babai's nearest plane algorithm [Bab86] to construct certain recoding matrices.

As explained above, this key-homomorphism gives us an ABE for arithmetic circuits: the public parameters contain random matrices $\mathbf{B}_1, \ldots, \mathbf{B}_\ell \in \mathbb{Z}_q^{n \times m}$ and encryption to an attribute vector $\mathbf{x}$ in $\mathbb{Z}_q^\ell$ is done using dual Regev encryption to the matrix (1.2). A decryption key sk_f for an arithmetic circuit $f : \mathbb{Z}_q^\ell \to \mathbb{Z}_q$ is a decryption key for the public-key matrix $(\mathbf{A} \mid 0 \cdot \mathbf{G} + \mathbf{B}_f) = (\mathbf{A}|\mathbf{B}_f)$. This key enables decryption whenever $f(\mathbf{x}) = 0$. The key sk_f can be easily generated using a short basis for the lattice $\Lambda_q^\perp(\mathbf{A})$ which serves as the master secret key for the ABE system.

We prove selective security from the learning with errors problem (LWE) by using another homomorphic property of the system implemented in an algorithm called $\mathsf{Eval}_{\mathrm{sim}}$. Using $\mathsf{Eval}_{\mathrm{sim}}$ the simulator responds to the adversary's private key queries and then solves the given LWE challenge.

Parameters and performance. Applying algorithm $\mathsf{Eval}_{\mathrm{ct}}(f, \mathsf{pk}, c)$ to a ciphertext c increases the magnitude of the noise in the ciphertext by a factor that depends on the depth of the circuit for f. A k-way addition gate (g_+) increases the norm of the noise by a factor of $O(km)$. A k-way multiplication gate $(g_\times)$ where all (but one) of the inputs are in $[-p, p]$ increases the norm of the noise by a factor of $O(p^{k-1}m)$. Therefore, if the circuit for f has depth d, the noise in c grows in the worst case by a factor of $O((p^{k-1}m)^d)$. Note that the weights α_i used in the gates g_+ and $g_\times$ have no effect on the amount of noise added.

For decryption to work correctly the modulus q should be slightly larger than the noise in the ciphertext. Hence, we need q on the order of $\Omega(B \cdot (p^{k-1}m)^d)$ where B is the maximum magnitude of the noise added to the ciphertext during encryption. For security we need the learning with errors (LWE) problem to be hard. LWE hardness depends on the ratio q/B — the smaller the ratio the harder the problem. In our settings $q/B = \Omega\big((p^{k-1}m)^d\big)$. Regev [Reg05] showed that n-dimensional LWE is as hard on average as (quantumly) approximating certain worst case lattice problems to a factor of $\tilde{O}(n \cdot q/B)$. These lattice problems are believed to be hard to approximate even when q/B is $2^{(n^\epsilon)}$ for some fixed $0 < \epsilon < 1/2$. Then to support circuits of depth $t(\lambda)$ for some polynomial $t(\cdot)$ we choose n such that $n \geq t(\lambda)^{(1/\epsilon)} \cdot (2\log_2 n + k \log p)^{1/\epsilon}$ and set $q = 2^{(n^\epsilon)}$. As usual $m = \Theta(n \log q)$. We set the LWE noise bound to $B = O(n)$. This ensures correctness of decryption and hardness of LWE since we have $\Omega((p^k m)^{t(\lambda)}) < q \leq 2^{(n^\epsilon)}$, as required. The system of [GVW13] uses similar parameters due to a similar growth in noise as a function of circuit depth.

Secret key size. A decryption key in our system is a single $2m \times m$ low-norm matrix. Since $m = \Theta(n \log q)$ and $\log_2 q$ grows linearly with the circuit depth d, the overall secret key size grows as $O(d^2)$ with the depth. In previous ABE systems for circuits [GVW13, GGH$^+$13b] secret keys grew as $O(d^2 s)$ where s is the number of boolean gates or wires in the circuit.

Polynomial gates. The modulus q used in our scheme depends on the maximum circuit depth that we need to support. We show in Section 5 that the depth can be further reduced by using gates that compute more general multivariate polynomials beyond the two polynomials g_+ and $g_\times$ from (1.1). For example, we can implement the entire k-way OR polynomial $1-(1-x_1)\cdots(1-x_k)$ using a *single* gate thereby reducing overall circuit depth.

Other related work. Predicate encryption [BW07, KSW08] provides a stronger privacy guarantee than ABE by additionally hiding the attribute vector $\mathbf{x}$. Predicate encryption systems for inner product functionalities can be built from bilinear maps [KSW08] and LWE [AFV11]. More recently, Garg et al. [GGH$^+$13a] constructed functional encryption (which implies predicate encryption) for all polynomial-size functionalities using indistinguishability obfuscation.

The encryption algorithm in our system is similar to encryption in the hierarchical-IBE of Agrawal, Boneh, and Boyen [ABB10]. We show that this system is key-homomorphic for polynomial-size arithmetic circuits which gives us an ABE for such circuits. The first hint of the key homomorphic properties of the [ABB10] hierarchical-IBE was presented by Agrawal, Freeman, and Vaikuntanathan [AFV11] who showed that the system is key-homomorphic with respect to low-weight linear transformations and used this fact to construct a predicate encryption system for inner-products. To handle high-weight linear transformations [AFV11] used bit decomposition to represent the large weights as bits. This expands the ciphertext by a factor of $\log_2 q$, but adds more functionality to the system. Our ABE, when presented with a circuit containing only linear gates (i.e. only g_+ gates), also provides a predicate encryption system for inner products in the same security model as [AFV11], but can handle high-weight linear transformations directly, without bit decomposition, thereby obtaining shorter ciphertexts and public-keys.

A completely different approach to building circuit ABE was presented by Garg, Gentry, Sahai, and Waters [GGS$^+$13] who showed that a general primitive they named *witness encryption* implies circuit ABE when combined with witness indistinguishable proofs.

2 Preliminaries

For a random variable X we denote by $x \leftarrow X$ the process of sampling a value x according to the distribution of X. Similarly, for a finite set S we denote by $x \leftarrow S$ the process of sampling a value x according to the uniform distribution over S. A non-negative function $\nu(\lambda)$ is *negligible* if for every polynomial $p(\lambda)$ it holds that $\nu(\lambda) \leq 1/p(\lambda)$ for all sufficiently large $\lambda \in \mathbb{N}$.

2.1 Attribute-Based Encryption

An attribute-based encryption (ABE) scheme for a class of functions $\mathcal{F}_\lambda = \{f : \mathcal{X}_\lambda \to \mathcal{Y}_\lambda\}$ is a quadruple $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{E}, \mathsf{D})$ of probabilistic polynomial-time algorithms: Setup takes a unary representation of the security parameter λ and outputs public parameters mpk and a master secret key msk; $\mathsf{KeyGen}(\mathsf{msk},\ f \in \mathcal{F}_\lambda)$ output a decryption key sk_f; $\mathsf{E}(\mathsf{mpk},\ x \in \mathcal{X}_\lambda,\ \mu)$ outputs ciphertext c, the encryption of message μ labeled with attribute vector x; $\mathsf{D}(\mathsf{sk}_f, c)$ outputs a message μ or $\perp$.

An attribute-based encryption (ABE) scheme for a class of functions $\mathcal{F}_\lambda = \{f : \mathcal{X}_\lambda \to \mathcal{Y}_\lambda\}$ is a quadruple $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{E}, \mathsf{D})$ of probabilistic polynomial-time algorithms.

Correctness. We require that for every circuit $f \in \mathcal{F}$, index $x \in \mathcal{X}$ where $f(x) = 0$, and message μ, it holds that $\mathsf{D}(\mathsf{sk}_f, c) = \mu$ with an overwhelming probability over the choice of $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(\lambda)$, $c \leftarrow \mathsf{E}(\mathsf{mpk}, x, \mu)$, and $\mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)$.

Security. We consider the standard notion of selective security for ABE schemes [GPS+06]. Specifically, we consider adversaries that first announce a challenge index x^*, and then receive the public parameters mpk of the scheme and oracle access to a key-generation oracle $\mathsf{KG}(\mathsf{msk}, x^*, f)$ that returns the secret key sk_f for $f \in \mathcal{F}$ if $f(x^*) \neq 0$ and returns $\perp$ otherwise. We require that any such efficient adversary has only a negligible probability in distinguishing between encryptions of two different messages. Formally, security is captured by the following definition.

Definition 2.1 (Selectively-secure ABE)**.** An attribute-based encryption scheme $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{E}, \mathsf{D})$ for a class of functions $\mathcal{F}_\lambda = \{f : \mathcal{X}_\lambda \to \mathcal{Y}_\lambda\}$ is *selectively secure* if for all probabilistic polynomial-time adversaries $\mathcal{A}$ where $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$, there is a negligible function $\nu(\lambda)$ such that

$$\mathbf{Adv}^{\mathsf{ABE}}_{\Pi,\mathcal{A}}(\lambda) \overset{\mathsf{def}}{=} \left| \Pr\left[\mathrm{EXP}^{(0)}_{\mathsf{ABE},\Pi,\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathrm{EXP}^{(1)}_{\mathsf{ABE},\Pi,\mathcal{A}}(\lambda) = 1\right] \right| \leq \nu(\lambda),$$

where for each $b \in \{0,1\}$ and $\lambda \in \mathbb{N}$ the experiment $\mathrm{EXP}^{(b)}_{\mathsf{ABE},\Pi,\mathcal{A}}(\lambda)$ is defined as follows:

1. $(x^*, \mathsf{state}_1) \leftarrow \mathcal{A}_1(\lambda)$, where $x^* \in \mathcal{X}_\lambda$ // $\mathcal{A}$ commits to challenge index x^*
2. $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(\lambda)$
3. $(\mu_0, \mu_1, \mathsf{state}_2) \leftarrow \mathcal{A}_2^{\mathsf{KG}(\mathsf{msk},x^*,\cdot)}(\mathsf{mpk}, \mathsf{state}_1)$ // $\mathcal{A}$ outputs challenge messages μ_0, μ_1
4. $c^* \leftarrow \mathsf{E}(\mathsf{mpk}, x^*, \mu_b)$
5. $b' \leftarrow \mathcal{A}_3^{\mathsf{KG}(\mathsf{msk},x^*,\cdot)}(c^*, \mathsf{state}_2)$ // $\mathcal{A}$ outputs a guess b' for b
6. Output $b' \in \{0,1\}$

where $\mathsf{KG}(\mathsf{msk}, x^*, f)$ returns a secret key $\mathsf{sk}_f = \mathsf{KeyGen}(\mathsf{msk}, f)$ if $f(x^*) \neq 0$ and $\perp$ otherwise.

2.2 Background on Lattices

Lattices. Let q, n, m be positive integers.
For a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ we let $\Lambda_q^{\perp}(\mathbf{A})$ denote the lattice $\{\mathbf{x} \in \mathbb{Z}^m \ : \ \mathbf{A}\,\mathbf{x} = 0 \text{ in } \mathbb{Z}_q\}$.
More generally, for $\mathbf{u} \in \mathbb{Z}_q^n$ we let $\Lambda_q^{\mathbf{u}}(\mathbf{A})$ denote the shifted lattice $\{\mathbf{x} \in \mathbb{Z}^m \ : \ \mathbf{A}\,\mathbf{x} = \mathbf{u} \text{ in } \mathbb{Z}_q\}$.

Observe that if the columns of $\mathbf{T}_A \in \mathbb{Z}^{m\times m}$ are a basis of the lattice $\Lambda_q^{\perp}(A)$, then they are also a basis for the lattice $\Lambda_q^{\perp}(xA)$ for any nonzero $x \in \mathbb{Z}_q$.

Learning with errors (LWE) [Reg05]. Fix integers n, m, q and a noise distribution χ over $\mathbb{Z}_q$ with q prime. The (n, m, q, χ)-LWE problem is to distinguish the following two distributions:

$$(\mathbf{A},\ \mathbf{A}^{\mathsf{T}}\mathbf{s} + \mathbf{e}) \qquad \text{and} \qquad (\mathbf{A}, \mathbf{u})$$

where $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}$, $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, $\mathbf{e} \leftarrow \chi^m$, $\mathbf{u} \leftarrow \mathbb{Z}_q^m$ are independently sampled. Throughout the paper we always set $m = \Theta(n \log q)$ and simply refer to the (n, q, χ)-LWE problem.

We say that a noise distribution χ is B-bounded if its support is in $[-B, B]$. For any fixed $d > 0$ and sufficiently large q, Regev [Reg05] (through a quantum reduction) and Peikert [Pei09] (through a classical reduction) show that taking χ as a certain q/n^d-bounded distribution, the (n, q, χ)-LWE problem is as hard as approximating the worst-case GapSVP to $\text{poly}(n)$ factors, which is believed to be intractable. More generally, let $\chi_{\max} < q$ be the bound on the noise distribution. The difficulty of the LWE problem is measured by the ratio $q/\chi_{\max}$. This ratio is always bigger than 1 and the smaller it is the harder the problem. The problem appears to remain hard even when $q/\chi_{\max} < 2^{(n^\epsilon)}$ for some fixed $\epsilon \in (0, 1/2)$.

Matrix norms. For a vector $\mathbf{u}$ we let $\|\mathbf{u}\|$ denote its ℓ_2 norm. For a matrix $\mathbf{R} \in \mathbb{Z}^{k\times m}$ we define three matrix norms:

$\|\mathbf{R}\|$ denotes the ℓ_2 length of the longest column of $\mathbf{R}$.
$\|\mathbf{R}\|_{\mathsf{GS}}$ denotes $\|\tilde{\mathbf{R}}\|$ where $\tilde{\mathbf{R}}$ is the result of applying Gram-Schmidt to the columns of $\mathbf{R}$.
$\|\mathbf{R}\|_2$ is the operator norm of $\mathbf{R}$ defined as $\|\mathbf{R}\|_2 = \sup_{\|\mathbf{x}\|=1} \|\mathbf{R}\mathbf{x}\|$.

Note that always $\|\mathbf{R}\|_{\mathsf{GS}} \le \|\mathbf{R}\| \le \|\mathbf{R}\|_2 \le \sqrt{k}\|\mathbf{R}\|$ and that $\|\mathbf{R} \cdot \mathbf{S}\|_2 \le \|\mathbf{R}\|_2 \cdot \|\mathbf{S}\|_2$.

Trapdoor generators. The following describes algorithms for generating short basis of lattices.

Lemma 2.2. *Let $n, m, q > 0$ be integers with q prime. There are polynomial time algorithms such that:*

1. *([Ajt99, AP09]):* $\mathsf{TrapGen}(1^n, 1^m, q) \longrightarrow (\mathbf{A}, \mathbf{T_A})$
a randomized algorithm that, when $m = \Theta(n \log q)$, outputs a full-rank matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and basis $\mathbf{T_A} \in \mathbb{Z}^{m\times m}$ for $\Lambda_q^\perp(A)$ such that $\mathbf{A}$ is negl(n)-close to uniform and $\|\mathbf{T}\|_{\mathsf{GS}} = O(\sqrt{n \log q})$, with all but negligible probability in n.

2. *([CHK+10]):* $\mathsf{ExtendRight}(\mathbf{A}, \mathbf{T_A}, \mathbf{B}) \longrightarrow \mathbf{T}_{(\mathbf{A}|\mathbf{B})}$
a deterministic algorithm that given full-rank matrices $\mathbf{A}, \mathbf{B} \in \mathbb{Z}_q^{n\times m}$ and a basis $\mathbf{T_A}$ of $\Lambda_q^\perp(\mathbf{A})$ outputs a basis $\mathbf{T}_{(\mathbf{A}|\mathbf{B})}$ of $\Lambda_q^\perp(\mathbf{A}|\mathbf{B})$ such that $\|\mathbf{T_A}\|_{\mathsf{GS}} = \|\mathbf{T}_{(\mathbf{A}|\mathbf{B})}\|_{\mathsf{GS}}$.

3. *([ABB10]):* $\mathsf{ExtendLeft}(\mathbf{A}, \mathbf{G}, \mathbf{T_G}, \mathbf{S}) \longrightarrow \mathbf{T_H}$ *where* $\mathbf{H} = (\mathbf{A} \mid \mathbf{G} + \mathbf{A}\ \mathbf{S})$
a deterministic algorithm that given full-rank matrices $\mathbf{A}, \mathbf{G} \in \mathbb{Z}_q^{n\times m}$ and a basis $\mathbf{T_G}$ of $\Lambda_q^\perp(\mathbf{G})$ outputs a basis $\mathbf{T_H}$ of $\Lambda_q^\perp(\mathbf{H})$ such that $\|\mathbf{T_H}\|_{\mathsf{GS}} \le \|\mathbf{T_G}\|_{\mathsf{GS}} \cdot (1 + \|\mathbf{S}\|_2)$.

4. *([MP12]): For $m = n\lceil \log q\rceil$ there is a fixed full-rank matrix $\mathbf{G} \in \mathbb{Z}_q^{n\times m}$ such that the lattice $\Lambda_q^\perp(\mathbf{G})$ has a publicly known basis $\mathbf{T_G} \in \mathbb{Z}^{m\times m}$ with $\|\mathbf{T_G}\|_{\mathsf{GS}} \le \sqrt{5}$.*

To simplify the notation we will always assume that the matrix $\mathbf{G}$ from part 4 of Lemma 2.2 has the same width m as the matrix $\mathbf{A}$ output by algorithm $\mathsf{TrapGen}$ from part 1 of the lemma. We do so without loss of generality since $\mathbf{G}$ can always be extended to the size of $\mathbf{A}$ by adding zero columns on the right of $\mathbf{G}$.

Discrete Gaussians. Regev [Reg05] defined a natural distribution on $\Lambda_q^{\mathbf{u}}(\mathbf{A})$ called a *discrete Gaussian* parameterized by a scalar $\sigma > 0$. We use $\mathcal{D}_\sigma(\Lambda_q^{\mathbf{u}}(\mathbf{A}))$ to denote this distribution. For a random matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and $\sigma > \tilde{O}(\sqrt{n})$, a vector $\mathbf{x}$ sampled from $\mathcal{D}_\sigma(\Lambda_q^{\mathbf{u}}(\mathbf{A}))$ has ℓ_2 norm less than $\sigma\sqrt{m}$ with probability at least $1 - \text{negl}(m)$.

For a matrix $\mathbf{U} = (\mathbf{u}_1|\cdots|\mathbf{u}_k) \in \mathbb{Z}_q^{n\times k}$ we let $\mathcal{D}_\sigma(\Lambda_q^{\mathbf{U}}(\mathbf{A}))$ be a distribution on matrices in $\mathbb{Z}^{m\times k}$ where the i-th column is sampled from $\mathcal{D}_\sigma(\Lambda_q^{\mathbf{u}_i}(\mathbf{A}))$ for $i = 1,\ldots,k$. Clearly if $\mathbf{R}$ is sampled from $\mathcal{D}_\sigma(\Lambda_q^{\mathbf{U}}(\mathbf{A}))$ then $\mathbf{AR} = \mathbf{U}$ in $\mathbb{Z}_q$.

Lemma 2.3. *For integers $n, m, k, q, \sigma > 0$, matrices $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{U} \in \mathbb{Z}_q^{n\times k}$, if $\mathbf{R} \in \mathbb{Z}^{m\times k}$ is sampled from $\mathcal{D}_\sigma(\Lambda_q^{\mathbf{U}}(\mathbf{A}))$ and $\mathbf{S}$ is sampled uniformly in $\{\pm 1\}^{m\times m}$ then*

$$\|\mathbf{R}^\mathsf{T}\|_2 \le \sigma\sqrt{mk} \quad , \quad \|\mathbf{R}\|_2 \le \sigma\sqrt{mk} \quad , \quad \|\mathbf{S}\|_2 \le 20\sqrt{m}$$

with overwhelming probability in m.

Proof. For the $\{\pm1\}$ matrix $\mathbf{S}$ the lemma follows from Litvak et al. [LPR+05] (Fact 2.4). For the matrix $\mathbf{R}$ the lemma follow from the fact that $\|\mathbf{R}^\mathsf{T}\|_2 \le \sqrt{k}\cdot\|\mathbf{R}\| < \sqrt{k}(\sigma\sqrt{m})$. ■

Solving $\mathbf{AX} = \mathbf{U}$. We review two algorithms, one deterministic [Bab86] and one randomized [GPV08], for finding a low-norm matrix $\mathbf{X} \in \mathbb{Z}^{m\times k}$ such that $\mathbf{AX} = \mathbf{U}$. Both need a short basis $\mathbf{T}$ of $\Lambda_q^\perp(\mathbf{A})$ and both construct the matrix $\mathbf{X}$ one column at a time.

Lemma 2.4. *Let $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{T_A} \in \mathbb{Z}^{m\times m}$ be a basis for $\Lambda_q^\perp(\mathbf{A})$. Let $\mathbf{U} \in \mathbb{Z}_q^{n\times k}$. There are polynomial time algorithms that output $\mathbf{X} \in \mathbb{Z}^{m\times k}$ satisfying $\mathbf{AX} = \mathbf{U}$ such that:*

1. *([Bab86]):* $\mathsf{SolveR}(\mathbf{A}, \mathbf{T_A}, \mathbf{U}) \longrightarrow \mathbf{X}$
a deterministic algorithm that outputs an $\mathbf{X}$ satisfying $\|\mathbf{X}\| \le \frac{1}{2}\sqrt{m}\cdot\|\mathbf{T_A}\|_{\text{GS}}$.
2. *([GPV08]):* $\mathsf{SampleD}(\mathbf{A}, \mathbf{T_A}, \mathbf{U}, \sigma) \longrightarrow \mathbf{X}$
a randomized algorithm that, when $\sigma = \|\mathbf{T}_A\|_{\text{GS}}\cdot\omega(\sqrt{\log m})$, outputs a random sample $\mathbf{X}$ from the distribution $\mathcal{D}_\sigma(\Lambda_q^{\mathbf{U}}(\mathbf{A}))$.
3. *([CHK+10]):* $\mathsf{RandBasis}(\mathbf{A}, \mathbf{T_A}, \sigma) \longrightarrow \mathbf{T}'_\mathbf{A}$
a randomized algorithm that, when $\sigma = \|\mathbf{T}_A\|_{\text{GS}}\cdot\omega(\sqrt{\log m})$, outputs a basis $\mathbf{T}'_\mathbf{A}$ of $\Lambda_q^\perp(\mathbf{A})$ sampled from $(\mathcal{D}_\sigma(\Lambda_q^\perp(\mathbf{A})))^m$. Note that $\|\mathbf{T}'_\mathbf{A}\|_{\text{GS}} < \sigma\sqrt{m}$ with all but negligible probability.

Lemma 2.5. *Let $\mathbf{G}$ be the matrix from Lemma 2.2 part 4. Then for all $\mathbf{U} \in \mathbb{Z}_q^{m\times n}$, $\mathbf{R} \in \mathbb{Z}_q^{m\times m}$, and $\mathbf{R} = \mathsf{SolveR}(\mathbf{G}, \mathbf{T_G}, \mathbf{U})$ we have that $\|\mathbf{R}\|_2 \le 2m$ and $\|\mathbf{R}^\mathsf{T}\|_2 \le 2m$.*

Proof. The lemma follows from the fact that $\|\mathbf{R}^T\|_2$ and $\|\mathbf{R}\|_2$ are upper-bounded by $\sqrt{m}\cdot\|\mathbf{R}\|$ and the following gives an upper-bound on $||\mathbf{R}||$:

$$||\mathbf{R}|| \overset{\textit{Lemma } 2.4}{\le} \frac{1}{2}\sqrt{m}\cdot\|\mathbf{T_G}\|_{\text{GS}} \overset{\textit{Lemma } 2.2}{\le} \frac{1}{2}\sqrt{m}\cdot\sqrt{5} \le 2\sqrt{m}$$

as required. ■

Randomness extraction. We conclude with a variant of the left-over hash lemma from [ABB10].

Lemma 2.6. *Suppose that $m > (n+1)\log_2 q + \omega(\log n)$ and that $q > 2$ is prime. Let $\mathbf{S}$ be an $m \times k$ matrix chosen uniformly in $\{1, -1\}^{m\times k} \bmod q$ where $k = k(n)$ is polynomial in n. Let $\mathbf{A}$ and $\mathbf{B}$ be matrices chosen uniformly in $\mathbb{Z}_q^{n\times m}$ and $\mathbb{Z}_q^{n\times k}$ respectively. Then, for all vectors $\mathbf{e}$ in $\mathbb{Z}_q^m$, the distribution $(\mathbf{A},\ \mathbf{AS},\ \mathbf{S}^{\mathsf{T}}\mathbf{e})$ is statistically close to the distribution $(\mathbf{A},\ \mathbf{B},\ \mathbf{S}^{\mathsf{T}}\mathbf{e})$.*

Note that the lemma holds for every vector $\mathbf{e}$ in $\mathbb{Z}_q^m$, including low norm vectors.

3 An ABE for arithmetic circuits from LWE

We now turn to building an ABE for arithmetic circuits from the learning with errors (LWE) problem. Our construction follows the key-homomorphism paradigm outlined in the introduction, but rather than describe it as a key-homomorphic system we directly describe the resulting ABE.

For integers n and $q = q(n)$ let $m = \Theta(n \log q)$. Let $\mathbf{G} \in \mathbb{Z}_q^{n\times m}$ be the fixed matrix from Lemma 2.2 (part 4). For $x \in \mathbb{Z}_q$, $\mathbf{B} \in \mathbb{Z}_q^{n\times m}$, $\mathbf{s} \in \mathbb{Z}_q^n$, and $\delta > 0$ define the set

$$E_{\mathbf{s},\delta}(x, \mathbf{B}) = \left\{ (x\mathbf{G} + \mathbf{B})^{\mathsf{T}}\mathbf{s} + \mathbf{e} \in \mathbb{Z}_q^m \quad \text{where} \quad \|\mathbf{e}\| < \delta \right\}$$

For now we will assume the existence of three efficient *deterministic* algorithms $\mathsf{Eval}_{\mathrm{pk}}, \mathsf{Eval}_{\mathrm{ct}}, \mathsf{Eval}_{\mathrm{sim}}$ that implement the key-homomorphic features of the scheme and are at the heart of the construction. We present them in the next section. These three algorithms must satisfy the following properties with respect to some family of functions $\mathcal{F} = \{f : (\mathbb{Z}_q)^\ell \to \mathbb{Z}_q\}$ and a function $\alpha_{\mathcal{F}} : \mathbb{Z} \to \mathbb{Z}$.

- $\mathsf{Eval}_{\mathrm{pk}}(\ f \in \mathcal{F},\ \ \vec{\mathbf{B}} \in (\mathbb{Z}_q^{n\times m})^\ell\ ) \longrightarrow \mathbf{B}_f\ \in \mathbb{Z}_q^{n\times m}$.

- $\mathsf{Eval}_{\mathrm{ct}}(\ f \in \mathcal{F},\ \ \big((x_i, \mathbf{B}_i, \mathbf{c}_i)\big)_{i=1}^{\ell}\ ) \longrightarrow \mathbf{c}_f\ \in \mathbb{Z}_q^m$. Here $x_i \in \mathbb{Z}_q$, $\mathbf{B}_i \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{c}_i \in E_{\mathbf{s},\delta}(x_i, \mathbf{B}_i)$ for some $\mathbf{s} \in \mathbb{Z}_q^n$ and $\delta > 0$. Note that the same $\mathbf{s}$ is used for all $\mathbf{c}_i$. The output $\mathbf{c}_f$ must satisfy

$$\mathbf{c}_f \in E_{\mathbf{s},\Delta}(f(\mathbf{x}), \mathbf{B}_f) \quad \text{where} \quad \mathbf{B}_f = \mathsf{Eval}_{\mathrm{pk}}(f,\ (\mathbf{B}_1, \ldots, \mathbf{B}_\ell))$$

  and $\mathbf{x} = (x_1, \ldots, x_\ell)$. We further require that $\Delta < \delta \cdot \alpha_{\mathcal{F}}(n)$ for some function $\alpha_{\mathcal{F}}(n)$ that measures the increase in the noise magnitude in $\mathbf{c}_f$ compared to the input ciphertexts.

  This algorithm captures the key-homomorphic property: it translates ciphertexts encrypted under public-keys $\{(x_i, \mathbf{B}_i)\}_{i=1}^{\ell}$ into a ciphertext $\mathbf{c}_f$ encrypted under public-key $(f(\mathbf{x}), \mathbf{B}_f)$.

- $\mathsf{Eval}_{\mathrm{sim}}(\ f \in \mathcal{F},\ \ \big((x_i^*, \mathbf{S}_i)\big)_{i=1}^{\ell},\ \ \mathbf{A}) \longrightarrow \mathbf{S}_f \in \mathbb{Z}_q^{m\times m}$. Here $x_i^* \in \mathbb{Z}_q$ and $\mathbf{S}_i \in \mathbb{Z}_q^{m\times m}$. With $\mathbf{x}^* = (x_1^*, \ldots, x_n^*)$, the output $\mathbf{S}_f$ satisfies

$$\mathbf{AS}_f - f(\mathbf{x}^*)\mathbf{G} = \mathbf{B}_f \quad \text{where} \quad \mathbf{B}_f = \mathsf{Eval}_{\mathrm{pk}}\big(f,\ (\mathbf{AS}_1 - x_1^*\mathbf{G}, \ldots, \mathbf{AS}_\ell - x_\ell^*\mathbf{G})\big)\ .$$

  We further require that for all $f \in \mathcal{F}$, if $\mathbf{S}_1, \ldots, \mathbf{S}_\ell$ are random matrices in $\{\pm 1\}^{m\times m}$ then $\|\mathbf{S}_f\|_2 < \alpha_{\mathcal{F}}(n)$ with all but negligible probability.

Definition 3.1. The deterministic algorithms $(\mathsf{Eval}_{\mathrm{pk}}, \mathsf{Eval}_{\mathrm{ct}}, \mathsf{Eval}_{\mathrm{sim}})$ are $\alpha_{\mathcal{F}}$-*ABE enabling* for some family of functions $\mathcal{F} = \{f : (\mathbb{Z}_q)^\ell \to \mathbb{Z}_q\}$ if there are functions $q = q(n)$ and $\alpha_{\mathcal{F}} = \alpha_{\mathcal{F}}(n)$ for which the properties above are satisfied.

We want $\alpha_{\mathcal{F}}$-ABE enabling algorithms for a large function family $\mathcal{F}$ and the smallest possible $\alpha_{\mathcal{F}}$. In the next section we build these algorithms for polynomial-size arithmetic circuits. The function $\alpha_{\mathcal{F}}(n)$ will depend on the depth of circuits in the family.

The ABE system. Given ABE-enabling algorithms $(\mathsf{Eval}_{\rm pk}, \mathsf{Eval}_{\rm ct}, \mathsf{Eval}_{\rm sim})$ for a family of functions $\mathcal{F} = \{f : (\mathbb{Z}_q)^\ell \to \mathbb{Z}_q\}$ we build an ABE for the same family of functions $\mathcal{F}$. We prove selective security based on the learning with errors problem.

Parameters : Choose n and $q = q(n)$ as needed for $(\mathsf{Eval}_{\rm pk}, \mathsf{Eval}_{\rm ct}, \mathsf{Eval}_{\rm sim})$ to be $\alpha_{\mathcal{F}}$-*ABE enabling* for the function family $\mathcal{F}$. In addition, let χ be a $\chi_{\max}$-bounded noise distribution for which the (n, q, χ)-LWE problem is hard as discussed in Section 2.2. As usual, we set $m = \Theta(n \log q)$.

Set $\sigma = \omega(\alpha_{\mathcal{F}} \cdot \sqrt{\log m})$. We instantiate these parameters concretely in the next section.

For correctness of the scheme we require that $\alpha_{\mathcal{F}}^2 \cdot m < \frac{1}{12} \cdot (q/\chi_{\max})$ and $\alpha_{\mathcal{F}} > \sqrt{n \log m}$.

$\mathsf{Setup}(1^\lambda, \ell)$: Run algorithm $\mathsf{TrapGen}(1^n, 1^m, q)$ from Lemma 2.2 (part 1) to generate $(\mathbf{A}, \mathbf{T_A})$ where $\mathbf{A}$ is a uniform full-rank matrix in $\mathbb{Z}_q^{n\times m}$.
Choose random matrices $\mathbf{D}, \mathbf{B}_1, \dots, \mathbf{B}_\ell \in \mathbb{Z}_q^{n\times m}$ and output the ABE parameters:

$$\mathsf{mpk} = (\mathbf{A}, \mathbf{D}, \mathbf{B}_1, \dots, \mathbf{B}_\ell) \quad ; \quad \mathsf{msk} = (\mathbf{T_A}, \mathbf{D}, \mathbf{B}_1, \dots, \mathbf{B}_\ell)$$

$\mathsf{KeyGen}(\mathsf{msk}, f)$: Let $\mathbf{B}_f = \mathsf{Eval}_{\rm pk}(f,\ (\mathbf{B}_1, \dots, \mathbf{B}_\ell))$.
Output $\mathsf{sk}_f := \mathbf{R}_f$ where $\mathbf{R}_f$ is a low-norm matrix in $\mathbb{Z}^{2m\times m}$ sampled from the discrete Gaussian distribution $\mathcal{D}_\sigma(\Lambda_q^{\mathbf{D}}(\mathbf{A}|\mathbf{B}_f))$ so that $(\mathbf{A}|\mathbf{B}_f) \cdot \mathbf{R}_f = \mathbf{D}$.

To construct $\mathbf{R}_f$ build the basis $\mathbf{T_F}$ for $\mathbf{F} = (\mathbf{A}|\mathbf{B}_f) \in \mathbb{Z}_q^{n\times 2m}$ as $\mathbf{T_F} \leftarrow \mathsf{ExtendRight}(\mathbf{A}, \mathbf{T_A}, \mathbf{B})$ from Lemma 2.2 (part 2).
Then run $\mathbf{R}_f \leftarrow \mathsf{SampleD}(\ \mathbf{F},\ \mathbf{T_F},\ \mathbf{D},\ \sigma)$. Here σ is sufficiently large for algorithm $\mathsf{SampleD}$ (Lemma 2.4 part 2) since $\sigma = \|\mathbf{T_F}\|_{\mathsf{GS}} \cdot \omega(\sqrt{\log m})$. where $\|\mathbf{T_F}\|_{\mathsf{GS}} = \|\mathbf{T_A}\|_{\mathsf{GS}} = O(\sqrt{n \log q})$.

Note that the secret key sk_f is always in $\mathbb{Z}^{2m\times m}$ independent of the complexity of the function f. We assume sk_f also implicitly includes mpk.

$\mathsf{E}(\mathsf{mpk},\ \mathbf{x} \in \mathbb{Z}_q^\ell,\ \mu \in \{0,1\}^m)$: Choose a random n dimensional vector $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ and error vectors $\mathbf{e}_0, \mathbf{e}_1 \leftarrow \chi^m$. Choose ℓ uniformly random matrices $\mathbf{S}_i \leftarrow \{\pm 1\}^{m\times m}$ for $i \in [\ell]$.
Set $\mathbf{H} \in \mathbb{Z}_q^{n\times(\ell+1)m}$ and $\mathbf{e} \in \mathbb{Z}_q^{(\ell+1)m}$ as

$$\begin{aligned}\mathbf{H} &= (\mathbf{A} \mid x_1\mathbf{G} + \mathbf{B}_1 \mid \cdots \mid x_\ell\mathbf{G} + \mathbf{B}_\ell) \quad \in \mathbb{Z}_q^{n\times(\ell+1)m} \\ \mathbf{e} &= (\mathbf{I}_m|\mathbf{S}_1|\dots|\mathbf{S}_\ell)^\mathsf{T} \cdot \mathbf{e}_0 \quad \in \mathbb{Z}_q^{(\ell+1)m}\end{aligned}$$

Let $\mathbf{c} = (\mathbf{H}^T\mathbf{s} + \mathbf{e},\ \ \mathbf{D}^T\mathbf{s} + \mathbf{e}_1 + \lceil q/2 \rceil \mu) \in \mathbb{Z}_q^{(\ell+2)m}$. Output the ciphertext $(\mathbf{x}, \mathbf{c})$.

$\mathsf{D}\big(\mathsf{sk}_f,\ (\mathbf{x}, \mathbf{c})\big)$: If $f(\mathbf{x}) \neq 0$ output $\perp$. Otherwise, let $\mathbf{c} = (\mathbf{c}_{in}, \mathbf{c}_1, \dots, \mathbf{c}_\ell, \mathbf{c}_{out}) \in \mathbb{Z}_q^{(\ell+2)m}$.

Set $\mathbf{c}_f = \mathsf{Eval}_{\rm ct}\big(f,\ \{(x_i, \mathbf{B}_i, \mathbf{c}_i)\}_{i=1}^\ell\big) \in \mathbb{Z}_q^m$.

Let $\mathbf{c}'_f = (\mathbf{c}_{in}|\mathbf{c}_f) \in \mathbb{Z}_q^{2m}$ and output $\mathsf{Round}(\mathbf{c}_{out} - \mathbf{R}_f^\mathsf{T}\mathbf{c}'_f) \in \{0,1\}^m$.

This completes the description of the system.

Correctness. The correctness of the scheme follows from our choice of parameters and, in particular, from the requirement $\alpha_{\mathcal{F}}^2 \cdot m < \frac{1}{12} \cdot (q/\chi_{\max})$. Specifically, to show correctness, first note that when $f(\mathbf{x}) = 0$ we know by the requirement on $\mathsf{Eval}_{\rm ct}$ that $\mathbf{c}_f$ is in $E_{\mathbf{s},\Delta}(0, \mathbf{B}_f)$ so that $\mathbf{c}_f = \mathbf{B}_f^\mathsf{T}\mathbf{s} + \mathbf{e}$ with $\|\mathbf{e}\| < \Delta$. Consequently,

$$\mathbf{c}'_f = (\mathbf{c}_{in}|\mathbf{c}_f) = (\mathbf{A}|\mathbf{B}_f)^\mathsf{T}\mathbf{s} + \mathbf{e}' \quad \text{where} \quad \|\mathbf{e}'\| < \Delta + \chi_{\max} < (\alpha_{\mathcal{F}} + 1)\chi_{\max} \ .$$

Since $\mathbf{R}_f \in \mathbb{Z}^{2m \times m}$ is sampled from the distribution $\mathcal{D}_\sigma(\Lambda_q^{\mathbf{D}}(\mathbf{A}|\mathbf{B}_f))$ we know that $(\mathbf{A}|\mathbf{B}_f) \cdot \mathbf{R}_f = \mathbf{D}$ and, by Lemma 2.3, $\|\mathbf{R}_f^{\mathsf{T}}\|_2 < 2m\sigma$ with overwhelming probability. Therefore

$$\mathbf{c}_{out} - \mathbf{R}_f^{\mathsf{T}} \mathbf{c}'_f = (\mathbf{D}^{\mathsf{T}}\mathbf{s} + \mathbf{e}_1) - (\mathbf{D}^{\mathsf{T}}\mathbf{s} + \mathbf{R}_f^{\mathsf{T}}\mathbf{e}') = \mathbf{e}_1 - \mathbf{R}_f^{\mathsf{T}}\mathbf{e}'$$

and $\quad \|\mathbf{e}_1 - \mathbf{R}_f^{\mathsf{T}}\mathbf{e}'\| \leq \chi_{\max} + 2m\sigma \cdot (\alpha_{\mathcal{F}} + 1)\chi_{\max} \leq 3\alpha_{\mathcal{F}}^2 \cdot \chi_{\max} \cdot m \quad$ with overwhelming probability. By the bounds on $\alpha_{\mathcal{F}}$ this quantity is less than $q/4$ thereby ensuring correct decryption of all bits of $\mu \in \{0,1\}^m$.

Security. Next we prove that our ABE is selectively secure for the family of functions $\mathcal{F}$ for which algorithms $(\mathsf{Eval}_{\mathrm{pk}}, \mathsf{Eval}_{\mathrm{ct}}, \mathsf{Eval}_{\mathrm{sim}})$ are ABE-enabling.

Theorem 3.2. *Given the three algorithms* $(\mathsf{Eval}_{pk}, \mathsf{Eval}_{ct}, \mathsf{Eval}_{sim})$ *for the family of functions* $\mathcal{F}$*, the ABE system above is selectively secure with respect to* $\mathcal{F}$*, assuming the* (n, q, χ)*-LWE assumption holds where* n, q, χ *are the parameters for the ABE.*

Proof idea. Before giving the complete proof we first briefly sketch the main proof idea which hinges on the properties of algorithms $(\mathsf{Eval}_{\mathrm{pk}}, \mathsf{Eval}_{\mathrm{ct}}, \mathsf{Eval}_{\mathrm{sim}})$. The proof employs some of the techniques from [CHK+10, ABB10].

We build an LWE algorithm $\mathcal{B}$ that uses a selective ABE attacker $\mathcal{A}$ to solve LWE. Algorithm $\mathcal{B}$ is given an LWE challenge matrix $(\mathbf{A}|\mathbf{D}) \in \mathbb{Z}_q^{n \times 2m}$ and two vectors $\mathbf{c}_{in}, \mathbf{c}_{out} \in \mathbb{Z}_q^m$ that are either random or their concatenation equals $(\mathbf{A}|\mathbf{D})^{\mathsf{T}}\mathbf{s} + \mathbf{e}$ for some small noise vector $\mathbf{e}$.

$\mathcal{A}$ starts by committing to the target attribute vector $\mathbf{x} = (x_1^*, \ldots, x_\ell^*) \in \mathbb{Z}_q^\ell$. In response $\mathcal{B}$ constructs the ABE public parameters by choosing random matrices $\mathbf{S}_1^*, \ldots, \mathbf{S}_\ell^*$ in $\{\pm 1\}^{m \times m}$ and setting $\mathbf{B}_i = \mathbf{A}\,\mathbf{S}_i^* - x_i^*\mathbf{G}$. It gives $\mathcal{A}$ the public parameters $\mathsf{mpk} = (\mathbf{A}, \mathbf{D}, \mathbf{B}_1, \ldots, \mathbf{B}_\ell)$. A standard argument shows that each of $\mathbf{A}\,\mathbf{S}_i^*$ is uniformly distributed in $\mathbb{Z}_q^{n \times m}$ so that all $\mathbf{B}_i$ are uniform as required for the public parameters.

Now, consider a private key query from $\mathcal{A}$ for a function $f \in \mathcal{F}$. Only functions f for which $y^* = f(x_1^*, \ldots, x_\ell^*) \neq 0$ are allowed. Let $\mathbf{B}_f = \mathsf{Eval}_{\mathrm{pk}}\big(f,\ (\mathbf{B}_1, \ldots, \mathbf{B}_\ell)\big)$. Then $\mathcal{B}$ needs to produce a matrix $\mathbf{R}_f$ in $\mathbb{Z}^{2m \times m}$ satisfying $\quad (\mathbf{A}|\mathbf{B}_f) \cdot \mathbf{R}_f = \mathbf{D}$. To do so $\mathcal{B}$ needs a short basis for the lattice $\Lambda_q^\perp(\mathbf{F})$ where $\mathbf{F} = (\mathbf{A}|\mathbf{B}_f)$. In the real key generation algorithm this short basis is derived from a short basis for $\Lambda_q^\perp(\mathbf{A})$ using algorithm $\mathsf{ExtendRight}$. Unfortunately, $\mathcal{B}$ has no short basis for $\Lambda_q^\perp(\mathbf{A})$.

Instead, as explained below, $\mathcal{B}$ builds a low-norm matrix $\mathbf{S}_f \in \mathbb{Z}_q^{m \times m}$ such that $\mathbf{B}_f = \mathbf{A}\mathbf{S}_f - y^*\mathbf{G}$. Then $\mathbf{F} = (\mathbf{A} \mid \mathbf{A}\mathbf{S}_f - y^*\mathbf{G})$. Because $y^* \neq 0$, algorithm $\mathcal{B}$ can construct the short basis $\mathbf{T}_{\mathbf{F}}$ for $\Lambda_q^\perp(\mathbf{F})$ using algorithm $\mathsf{ExtendLeft}(y^*\mathbf{G}, \mathbf{T}_G, \mathbf{A}, \mathbf{S}_f)$ from Lemma 2.2 part 3. Using $\mathbf{T}_{\mathbf{F}}$ algorithm $\mathcal{B}$ can now generate the required key as $\mathbf{R}_f \leftarrow \mathsf{SampleD}(\mathbf{F}, \mathbf{T}_{\mathbf{F}}, \mathbf{D}, \sigma)$. Note that when $y^* = 0$ algorithm $\mathsf{ExtendLeft}$ cannot be applied and $\mathcal{B}$ cannot generate secret keys for such functions f.

The remaining question is how does algorithm $\mathcal{B}$ build a low-norm matrix $\mathbf{S}_f \in \mathbb{Z}_q^{m \times m}$ such that $\mathbf{B}_f = \mathbf{A}\mathbf{S}_f - y^*\mathbf{G}$. To so $\mathcal{B}$ uses $\mathsf{Eval}_{\mathrm{sim}}$ giving it the secret matrices $\mathbf{S}_i^*$. More precisely, $\mathcal{B}$ runs $\mathsf{Eval}_{\mathrm{sim}}\big(f,\ \big((x_i^*, \mathbf{S}_i^*)\big)_{i=1}^\ell,\ \mathbf{A}\big)$ and obtains the required $\mathbf{S}_f$. This lets $\mathcal{B}$ answer all private key queries.

To complete the proof it is not difficult to show that $\mathcal{B}$ can build a challenge ciphertext $\mathbf{c}^*$ for the attribute vector $\mathbf{x} \in \mathbb{Z}_q^\ell$ that lets it solve the given LWE instance using adversary $\mathcal{A}$. An important point is that $\mathcal{B}$ cannot construct a key that will let it decrypt $\mathbf{c}^*$. The reason is that it cannot build a secret key sk_f for functions where $f(\mathbf{x}) = 0$ and these are the only keys that will decrypt $\mathbf{c}^*$.

Proof of Theorem 3.2. The proof proceeds in a sequence of games where the first game is identical to the ABE game from Definition 2.1. In the last game in the sequence the adversary has advantage zero. We show that a PPT adversary cannot distinguish between the games which will prove that the adversary has negligible advantage in winning the original ABE security game. The LWE problem is used in proving that Games 2 and 3 are indistinguishable.

Game 0. This is the original ABE security game from Definition 2.1 between an attacker $\mathcal{A}$ against our scheme and an ABE challenger.

Game 1. Recall that in Game 0 part of the public parameters mpk are generated by choosing random matrices $\mathbf{B}_1, \dots, \mathbf{B}_\ell$ in $\mathbb{Z}_q^{n \times m}$. At the challenge phase (step 4 in Definition 2.1) a challenge ciphertext $\mathbf{c}^*$ is generated. We let $\mathbf{S}_1^*, \dots, \mathbf{S}_\ell^* \in \{-1, 1\}^{m \times m}$ denote the random matrices generated for the creation of $\mathbf{c}^*$ in the encryption algorithm E.

In Game 1 we slightly change how the matrices $\mathbf{B}_1, \dots, \mathbf{B}_\ell$ are generated for the public parameters. Let $\mathbf{x}^* = (x_1^*, \dots, x_\ell^*) \in \mathbb{Z}_q^\ell$ be the target point that $\mathcal{A}$ intends to attack. In Game 1 the random matrices $\mathbf{S}_1^*, \dots, \mathbf{S}_\ell^*$ in $\{\pm 1\}^{m \times m}$ are chosen at the setup phase (step 2) and the matrices $\mathbf{B}_1, \dots, \mathbf{B}_\ell$ are constructed as

$$\mathbf{B}_i := \mathbf{A}\,\mathbf{S}_i^* - x_i^* \mathbf{G} \tag{3.1}$$

The remainder of the game is unchanged.

We show that Game 0 is statistically indistinguishable from Game 1 by Lemma 2.6. Observe that in Game 1 the matrices $\mathbf{S}_i^*$ are used only in the construction of $\mathbf{B}_i$ and in the construction of the challenge ciphertext where $\mathbf{e} := (\mathbf{I}_m | \mathbf{S}_1^* | \cdots | \mathbf{S}_\ell^*)^\mathsf{T} \cdot \mathbf{e}_0$ is used as the noise vector for some $\mathbf{e}_0 \in \mathbb{Z}_q^m$. Let $\mathbf{S}^* = (\mathbf{S}_1^* | \cdots | \mathbf{S}_\ell^*)$, then by Lemma 2.6 the distribution $(\mathbf{A},\ \mathbf{A}\,\mathbf{S}^*,\ \mathbf{e})$ is statistically close to the distribution $(\mathbf{A},\ \mathbf{A}',\ \mathbf{e})$ where $\mathbf{A}'$ is a uniform matrix in $\mathbb{Z}_q^{n \times \ell m}$. It follows that in the adversary's view, all the matrices $\mathbf{A}\,\mathbf{S}_i^*$ are statistically close to uniform and therefore the $\mathbf{B}_i$ as defined in (3.1) are close to uniform. Hence, the $\mathbf{B}_i$ in Games 0 and 1 are statistically indistinguishable.

Game 2. We now change how $\mathbf{A}$ in mpk is chosen. In Game 2 we generate $\mathbf{A}$ as a random matrix in $\mathbb{Z}_q^{n \times m}$. The construction of $\mathbf{B}_1, \dots, \mathbf{B}_\ell$ remains as in Game 1, namely $\mathbf{B}_i = \mathbf{A}\ \mathbf{S}_i^* - x_i^* \mathbf{G}$.

The key generation oracle responds to private key queries (in steps 3 and 5 of Definition 2.1) using the trapdoor $\mathbf{T}_\mathbf{G}$. Consider a private key query for function $f \in \mathcal{F}$. Only f such that $y^* = f(x_1^*, \dots, x_\ell^*) \neq 0$ are allowed. To respond, the key generation oracle computes $\mathbf{B}_f = \mathsf{Eval}_{\mathrm{pk}}\big(f,\ (\mathbf{B}_1, \dots, \mathbf{B}_\ell)\big)$ and needs to produce a matrix $\mathbf{R}_f$ in $\mathbb{Z}^{2m \times m}$ satisfying

$$(\mathbf{A} | \mathbf{B}_f) \cdot \mathbf{R}_f = \mathbf{D} \quad \text{in } \mathbb{Z}_q\ .$$

To do so the key generation oracle does:

- It runs $\mathbf{S}_f \leftarrow \mathsf{Eval}_{\mathrm{sim}}(f,\ \big((x_i^*, \mathbf{S}_i^*)\big)_{i=1}^\ell,\ \mathbf{A})$ and obtains a low-norm matrix $\mathbf{S}_f \in \mathbb{Z}_q^{m \times m}$ such that $\mathbf{A}\mathbf{S}_f - y^*\mathbf{G} = \mathbf{B}_f$. By definition of $\mathsf{Eval}_{\mathrm{sim}}$ we know that $\|\mathbf{S}_f\|_2 \leq \alpha_\mathcal{F}$.
- Let $\mathbf{F} = (\mathbf{A} | \mathbf{B}_f) = (\mathbf{A} | \mathbf{A}\mathbf{S}_f - y^*\mathbf{G})$. Because $y^* \neq 0$ the key generation oracle can obtain a trapdoor $\mathbf{T}_\mathbf{F}$ by running
$$\mathbf{T}_\mathbf{F} \leftarrow \mathsf{ExtendLeft}(y^*\mathbf{G}, \mathbf{T}_G, \mathbf{A}, \mathbf{S}_f)$$
By Lemma 2.2 (part 3) this trapdoor satisfies
$$\|\mathbf{T}_\mathbf{F}\|_{\mathsf{GS}} \leq \|\mathbf{T}_\mathbf{G}\|_{\mathsf{GS}} \cdot (1 + \|\mathbf{S}_f\|_2) \leq \sqrt{5}\ (1 + \alpha_\mathcal{F}(n))$$
where the bound on $\|\mathbf{T}_\mathbf{G}\|_{\mathsf{GS}}$ is from Lemma 2.2 (part 4).
- Finally, it responds with $\mathbf{R}_f = \mathsf{SampleD}(\mathbf{F},\ \mathbf{T}_\mathbf{F},\ \mathbf{D},\ \sigma)$.
By definition of $\mathsf{SampleD}$ we know that $\mathbf{R}_f$ is distributed as $\mathcal{D}_\sigma(\Lambda_q^\mathbf{D}(\mathbf{F}))$ as required. Indeed $\sigma = \|\mathbf{T}_\mathbf{F}\|_{\mathsf{GS}} \cdot \omega(\sqrt{\log m})$ as needed for algorithm $\mathsf{SampleD}$ in Lemma 2.4 (part 2).

Game 2 is otherwise the same as Game 1. Since the public parameters and responses to private key queries are statistically close to those in Game 1, the adversary's advantage in Game 2 is at most negligibly different from its advantage in Game 1.

Game 3. Game 3 is identical to Game 2 except that in the challenge ciphertext $(\mathbf{x}^*, \mathbf{c}^*)$ the vector $\mathbf{c}^* = (\mathbf{c}_{in}|\mathbf{c}_1|\cdots|\mathbf{c}_\ell|\mathbf{c}_{out}) \in \mathbb{Z}_q^{(\ell+2)m}$ is chosen as a random independent vector in $\mathbb{Z}_q^{(\ell+2)m}$. Since the challenge ciphertext is always a fresh random element in the ciphertext space, $\mathcal{A}$'s advantage in this game is zero.

It remains to show that Game 2 and Game 3 are computationally indistinguishable for a PPT adversary, which we do by giving a reduction from the LWE problem.

Reduction from LWE. Suppose $\mathcal{A}$ has non-negligible advantage in distinguishing Games 2 and 3. We use $\mathcal{A}$ to construct an LWE algorithm $\mathcal{B}$.

LWE Instance. $\mathcal{B}$ begins by obtaining an LWE challenge consisting of two random matrices $\mathbf{A}, \mathbf{D}$ in $\mathbb{Z}_q^{n\times m}$ and two vectors $\mathbf{c}_{in}, \mathbf{c}_{out}$ in $\mathbb{Z}_q^m$. We know that $\mathbf{c}_{in}, \mathbf{c}_{out}$ are either random in $\mathbb{Z}_q^m$ or

$$\mathbf{c}_{in} = \mathbf{A}^\mathsf{T}\mathbf{s} + \mathbf{e}_0 \quad \text{and} \quad \mathbf{c}_{out} = \mathbf{D}^\mathsf{T}\mathbf{s} + \mathbf{e}_1 \tag{3.2}$$

for some random vector $\mathbf{s} \in \mathbb{Z}_q^n$ and $\mathbf{e}_0, \mathbf{e}_1 \leftarrow \chi^m$. Algorithm $\mathcal{B}$'s goal is to distinguish these two cases with non-negligible advantage by using $\mathcal{A}$.

Public parameters. $\mathcal{A}$ begins by committing to a target point $\mathbf{x} = (x_1^*, \ldots, x_\ell^*) \in \mathbb{Z}_q^m$ where it wishes to be challenged. $\mathcal{B}$ assembles the public parameters mpk as in Game 2: choose random matrices $\mathbf{S}_1^*, \ldots, \mathbf{S}_\ell^*$ in $\{\pm 1\}^{m\times m}$ and set $\mathbf{B}_i = \mathbf{A}\,\mathbf{S}_i^* - x_i^*\mathbf{G}$. It gives $\mathcal{A}$ the public parameters

$$\mathsf{mpk} = (\mathbf{A}, \mathbf{D}, \mathbf{B}_1, \ldots, \mathbf{B}_\ell)$$

Private key queries. $\mathcal{B}$ answers $\mathcal{A}$'s private-key queries (in steps 3 and 5 of Definition 2.1) as in Game 2.

Challenge ciphertext. When $\mathcal{B}$ receives two messages $\mu_0, \mu_1 \in \{0,1\}^m$ from $\mathcal{A}$, it prepares a challenge ciphertext by choosing a random $b \leftarrow \{0,1\}$ and computing

$$\mathbf{c}_0^* \;=\; (\mathbf{I}_m|\mathbf{S}_1^*|\ldots|\mathbf{S}_\ell^*)^\mathsf{T} \cdot \mathbf{c}_{in} \quad \in \mathbb{Z}_q^{(\ell+1)m} \tag{3.3}$$

and $\mathbf{c}^* = (\mathbf{c}_0^*,\ \mathbf{c}_{out} + \lceil q/2 \rceil \mu_b) \in \mathbb{Z}_q^{(\ell+2)m}$. $\mathcal{B}$ sends $(\mathbf{x}^*, \mathbf{c}^*)$ as the challenge ciphertext to $\mathcal{A}$.

We argue that when the LWE challenge is pseudorandom (namely (3.2) holds) then $\mathbf{c}^*$ is distributed exactly as in Game 2. First, observe that when encrypting $(\mathbf{x}^*, \mu_b)$ the matrix $\mathbf{H}$ constructed in the encryption algorithm E is

$$\begin{aligned}\mathbf{H} &= (\mathbf{A} \mid x_1^*\mathbf{G} + \mathbf{B}_1 \mid \cdots \mid x_\ell^*\mathbf{G} + \mathbf{B}_\ell) \\ &= \big(\mathbf{A} \mid x_1^*\mathbf{G} + (\mathbf{A}\mathbf{S}_1^* - x_1^*\mathbf{G}) \mid \cdots \mid x_\ell^*\mathbf{G} + (\mathbf{A}\mathbf{S}_\ell^* - x_\ell^*\mathbf{G})\big) = (\mathbf{A} \mid \mathbf{A}\mathbf{S}_1^* \mid \cdots \mid \mathbf{A}\mathbf{S}_\ell^*)\end{aligned}$$

Therefore, $\mathbf{c}_0^*$ defined in (3.3) satisfies:

$$\begin{aligned}\mathbf{c}_0^* &= (\mathbf{I}_m|\mathbf{S}_1^*|\ldots|\mathbf{S}_\ell^*)^\mathsf{T} \cdot (\mathbf{A}^\mathsf{T}\mathbf{s} + \mathbf{e}_0) \\ &= (\mathbf{A}|\mathbf{A}\mathbf{S}_1^* \mid \cdots \mid \mathbf{A}\mathbf{S}_\ell^*)^\mathsf{T} \cdot \mathbf{s} + (\mathbf{I}_m|\mathbf{S}_1^*|\cdots|\mathbf{S}_\ell^*)^\mathsf{T} \cdot \mathbf{e}_0 = \mathbf{H}^\mathsf{T}\mathbf{s} + \mathbf{e}\end{aligned}$$

where $\mathbf{e} = (\mathbf{I}_m|\mathbf{S}_1^*|\cdots|\mathbf{S}_\ell^*)^\mathsf{T} \cdot \mathbf{e}_0$. This $\mathbf{e}$ is sampled from the same distribution as the noise vector $\mathbf{e}$ in algorithm E. We therefore conclude that $\mathbf{c}_0^*$ is computed as in Game 2. Moreover, since $\mathbf{c}_{out} = \mathbf{D}^{\mathsf{T}}\mathbf{s} + \mathbf{e}_1$ we know that the entire challenge ciphertext $\mathbf{c}^*$ is a valid encryption of $(\mathbf{x}^*, \mu_b)$ as required.

When the LWE challenge is random we know that $\mathbf{c}_{in}$ and $\mathbf{c}_{out}$ are uniform in $\mathbb{Z}_q^m$. Therefore the public parameters and $\mathbf{c}_0^*$ defined in (3.3) are uniform and independent in $\mathbb{Z}_q^{(\ell+1)m}$ by a standard application of the left over hash lemma (e.g. Theorem 8.38 of [Sho08]) where the universal hash function is defined as multiplication by the random matrix $(\mathbf{A}^{\mathsf{T}}|\mathbf{c}_{in})^{\mathsf{T}}$. Since $\mathbf{c}_{out}$ is also uniform, the challenge ciphertext overall is uniform in $\mathbb{Z}_q^{(\ell+2)m}$, as in Game 3.

Guess. Finally, $\mathcal{A}$ guesses if it is interacting with a Game 2 or Game 3 challenger. $\mathcal{B}$ outputs $\mathcal{A}$'s guess as the answer to the LWE challenge it is trying to solve.

We already argued that when the LWE challenge is pseudorandom the adversary's view is as in Game 2. When the LWE challenge is random the adversary's view is as in Game 3. Hence, $\mathcal{B}$'s advantage in solving LWE is the same as $\mathcal{A}$'s advantage in distinguishing Games 2 and 3, as required. This completes the description of algorithm $\mathcal{B}$ and completes the proof. ■

4 Evaluation Algorithms for Arithmetic Circuits

In this section we build the *ABE-enabling* algorithms $(\mathsf{Eval}_{\text{pk}}, \mathsf{Eval}_{\text{ct}}, \mathsf{Eval}_{\text{sim}})$ that are at the heart of the ABE construction in Section 3. We do so for the family of polynomial depth, unbounded fan-in arithmetic circuits.

4.1 Evaluation algorithms for gates

We first describe Eval algorithms for single gates, i.e. when $\mathcal{G}$ is the set of functions that each takes k inputs and computes either weighted addition or multiplication:

$$\mathcal{G} = \bigcup_{\alpha,\alpha_1,\alpha_2,\ldots,\alpha_k \in \mathbb{Z}_q} \left\{ g \mid g : \mathbb{Z}_q^k \to \mathbb{Z}_q, \quad \begin{array}{c} g(x_1,\ldots,x_k) = \alpha_1 x_1 + \alpha_2 x_2 + \ldots + \alpha_k x_k \\ \text{or} \\ g(x_1,\ldots,x_k) = \alpha \cdot x_1 \cdot x_2 \cdot \ldots \cdot x_k \end{array} \right\} \tag{4.1}$$

We assume that all the inputs to a multiplication gate (except possibly one input) are integers in the interval $[-p, p]$ for some bound $p < q$.

We present all three deterministic Eval algorithms at once:

$\mathsf{Eval}_{\text{pk}}(g \in \mathcal{G},\ \vec{\mathbf{B}} \in (\mathbb{Z}_q^{n\times m})^k\ ) \longrightarrow \mathbf{B}_g\ \in \mathbb{Z}_q^{n\times m}$
$\mathsf{Eval}_{\text{ct}}(g \in \mathcal{G},\ \big((x_i, \mathbf{B}_i, \mathbf{c}_i)\big)_{i=1}^k\ ) \longrightarrow \mathbf{c}_g\ \in \mathbb{Z}_q^m$
$\mathsf{Eval}_{\text{sim}}(g \in \mathcal{G},\ \big((x_i, \mathbf{S}_i)\big)_{i=1}^k,\ \mathbf{A}) \longrightarrow \mathbf{S}_g \in \mathbb{Z}_q^{m\times m}$

- For a weighted **addition** gate $g(x_1,\ldots,x_k) = \alpha_1 x_1 + \cdots + \alpha_k x_k$ do:
  For $i \in [k]$ generate matrix $\mathbf{R}_i \in \mathbb{Z}_q^{m\times m}$ such that

  $$\mathbf{G}\mathbf{R}_i = \alpha_i \mathbf{G}\ :\ \mathbf{R}_i = \mathsf{SolveR}(\mathbf{G}, \mathbf{T}_G, \alpha_i \mathbf{G}) \qquad \text{(as in Lemma 2.4 part 1).} \tag{4.2}$$

  Output the following matrices and the ciphertext:

  $$\mathbf{B}_g = \sum_{i=1}^k \mathbf{B}_i \mathbf{R}_i, \qquad \mathbf{S}_g = \sum_{i=1}^k \mathbf{S}_i \mathbf{R}_i, \qquad \mathbf{c}_g = \sum_{i=1}^k \mathbf{R}_i^T \mathbf{c}_i \tag{4.3}$$

- For a weighted **multiplication** gate $g(x_1, \dots, x_k) = \alpha x_1 \cdot \dots \cdot x_k$ do:
  For $i \in [k]$ generate matrices $\mathbf{R}_i \in \mathbb{Z}_q^{m\times m}$ such that

$$\mathbf{G}\mathbf{R}_1 = \alpha\mathbf{G} \ : \ \mathbf{R}_1 = \mathsf{SolveR}(\mathbf{G}, \mathbf{T}_G, \alpha\mathbf{G}) \tag{4.4}$$

$$\mathbf{G}\mathbf{R}_i = -\mathbf{B}_{i-1}\mathbf{R}_{i-1} \ : \ \mathbf{R}_i = \mathsf{SolveR}(\mathbf{G}, \mathbf{T}_G, -\mathbf{B}_{i-1}\mathbf{R}_{i-1}) \quad \text{for all } i \in \{2, 3, \dots, k\} \tag{4.5}$$

  Output the following matrices and the ciphertext:

$$\mathbf{B}_g = \mathbf{B}_k\mathbf{R}_k, \qquad \mathbf{S}_g = \sum_{j=1}^{k}\left(\prod_{i=j+1}^{k} x_i\right)\mathbf{S}_j\mathbf{R}_j, \qquad \mathbf{c}_g = \sum_{j=1}^{k}\left(\prod_{i=j+1}^{k} x_i\right)\mathbf{R}_j^T\mathbf{c}_j \tag{4.6}$$

  For example, for $k = 2$, $\mathbf{B}_g = \mathbf{B}_2\mathbf{R}_2$, $\mathbf{S}_g = x_2\mathbf{S}_1\mathbf{R}_1 + \mathbf{S}_2\mathbf{R}_2$, $\mathbf{c}_g = x_2\mathbf{R}_1^T\mathbf{c}_1 + \mathbf{R}_2^T\mathbf{c}_2$.

For multiplication gates, the reason we need an upper bound p on all but one of the inputs x_i is that these x_i values are used in (4.6) and we need the norm of $\mathbf{S}_g$ and the norm of the noise in the ciphertext $\mathbf{c}_g$ to be bounded from above.

The next two lemmas show that these algorithms satisfy the required properties to be ABE enabling.

Lemma 4.1. *Let $\beta_g(m) = 2km$. For a* **weighted addition** *gate $g(\mathbf{x}) = \alpha_1 x_1 + \ldots + \alpha_k x_k$ we have:*

1. *If $\mathbf{c}_i \in E_{\mathbf{s},\delta}(x_i, \mathbf{B}_i)$ for some $\mathbf{s} \in \mathbb{Z}_q^n$ and $\delta > 0$, then $\mathbf{c}_g \in E_{\mathbf{s},\Delta}(g(\mathbf{x}), \mathbf{B}_g)$ where $\Delta \le \beta_g(m)\cdot\delta$ and $\mathbf{B}_g = \mathsf{Eval}_{pk}(g,\ (\mathbf{B}_1, \dots, \mathbf{B}_k))$.*
2. *The output $\mathbf{S}_g$ satisfies $\mathbf{A}\mathbf{S}_g - g(\mathbf{x})\mathbf{G} = \mathbf{B}_g$ where $\|\mathbf{S}_g\|_2 \le \beta_g(m)\cdot \max_{i\in[k]}\|\mathbf{S}_i\|_2$ and $\mathbf{B}_g = \mathsf{Eval}_{pk}\big(g,\ (\mathbf{A}\mathbf{S}_1 - x_1\mathbf{G}, \dots, \mathbf{A}\mathbf{S}_k - x_k\mathbf{G})\big)$.*

Proof. By Eq. 4.3 the output ciphertext is computed as follows:

$$\begin{aligned}
\mathbf{c}_g &= \sum_{i=1}^{k}\mathbf{R}_i^T\cdot\mathbf{c}_i = \sum_{i=1}^{k}\mathbf{R}_i^T\cdot\Big((x_i\mathbf{G} + \mathbf{B}_i)^T\mathbf{s} + \mathbf{e}_i\Big) = \qquad // \text{ substitute for } \mathbf{c}_i = (x_i\mathbf{G} + \mathbf{B}_i)^T\mathbf{s} + \mathbf{e}_i\\
&= \sum_{i=1}^{k}(x_i\mathbf{G}\mathbf{R}_i)^T\mathbf{s} + \sum_{i=1}^{k}(\mathbf{B}_i\mathbf{R}_i)^T\mathbf{s} + \sum_{i=1}^{k}(\mathbf{R}_i^T\mathbf{e}_i) = \qquad // \text{ break the product into components}\\
&= \left(\sum_{i=1}^{k}\alpha_i x_i\right)\mathbf{G}^T\mathbf{s} + \mathbf{B}_g^T\mathbf{s} + \mathbf{e}_g = \qquad // \ \mathbf{G}\mathbf{R}_i = \alpha_i\mathbf{R}_i \text{ from Eq. 4.2 and } \mathbf{B}_g = \sum_{i=1}^{k}\mathbf{B}_i\mathbf{R}_i \text{ from Eq. 4.3}\\
&= [g(\mathbf{x})\mathbf{G} + \mathbf{B}_g]^T\mathbf{s} + \mathbf{e}_g
\end{aligned}$$

The noise bound is: $\|\mathbf{e}_g\| = \|\mathbf{R}_1^T\mathbf{e}_1 + \cdots + \mathbf{R}_k^T\mathbf{e}_k\| \le k\cdot\max_{j\in[k]}\left(\|\mathbf{R}_j^T\|_2\cdot\|\mathbf{e}_j\|\right) \overset{\textit{Lemma } 2.5}{\le} 2km\cdot\delta$. This completes the proof of the first part of the lemma.

In the second part of the lemma, by Eq. 4.3 the output matrix $\mathbf{B}_g$ is computed as follows:

$$\begin{aligned}
\mathbf{B}_g &= \sum_{i=1}^{k}(\mathbf{A}\mathbf{S}_i - x_i\mathbf{G})\mathbf{R}_i = \qquad // \text{ plug-in matrices given in the lemma into Eq. 4.3}\\
&= \mathbf{A}\sum_{i=1}^{k}\mathbf{S}_i\mathbf{R}_i - \sum_{i=1}^{k}\alpha_i x_i\mathbf{G} = \mathbf{A}\mathbf{S}_g - g(x)\mathbf{G} \qquad // \ \mathbf{G}\mathbf{R}_i = \alpha_i\mathbf{R}_i \text{ from Eq. 4.2}
\end{aligned}$$

Moreover $\|\mathbf{S}_g\|_2 = \|\sum_{i=1}^{k}\mathbf{S}_i\mathbf{R}_i\|_2 \le k\cdot\max_{i\in[k]}\left(\|\mathbf{S}_i\|_2\cdot\|\mathbf{R}_i\|_2\right) \overset{\textit{Lemma } 2.5}{\le} 2km\cdot\max_{i\in[k]}\left(\|\mathbf{S}_i\|_2\right)$ as required. ■

The next Lemma proves similar bounds for a multiplication gate.

Lemma 4.2. *For a* **multiplication** *gate* $g(\mathbf{x}) = \alpha \prod_{i=1}^{k} x_i$ *we have the same bounds on* $\mathbf{c}_g$ *and* $\mathbf{S}_g$ *as in Lemma 4.1 with* $\beta_g(m) = 2\frac{p^k-1}{p-1}m = O(p^{k-1}m)$.

Proof. Set $\mathbf{e}_g = \sum_{j=1}^{k} \left(\prod_{i=j+1}^{k} x_i\right) \mathbf{R}_j^T \mathbf{e}_j$. Then the output ciphertext is computed as follows:

$$\begin{aligned}
\mathbf{c}_g =& \sum_{j=1}^{k} \left(\prod_{i=j+1}^{k} x_i\right) \mathbf{R}_j^T \mathbf{c}_j = \sum_{j=1}^{k} \left(\prod_{i=j+1}^{k} x_i\right) \mathbf{R}_j^T \left((x_j\mathbf{G} + \mathbf{B}_j)^T \mathbf{s} + \mathbf{e}_j\right) = \qquad \text{// substitute for } \mathbf{c}_j \\
=& \left[\left(\prod_{i=1}^{k} x_i\right) \mathbf{G}\mathbf{R}_1 + \sum_{j=2}^{k} \left(\prod_{i=j}^{k} x_i\right) \cancel{(\mathbf{G}\mathbf{R}_j + \mathbf{B}_{j-1}\mathbf{R}_{j-1})} + \mathbf{B}_k\mathbf{R}_k\right]^T \mathbf{s} + \mathbf{e}_g = \qquad \text{// regroup} \\
=& \left[\left(\prod_{i=1}^{k} x_i\right) \mathbf{G}\mathbf{R}_1 + \mathbf{B}_k\mathbf{R}_k\right]^T \mathbf{s} + \mathbf{e}_g = \qquad \text{// use Eq. 4.5 to cancel terms} \\
=& \left[g(\mathbf{x})\mathbf{G} + \mathbf{B}_g\right]^T \mathbf{s} + \mathbf{e}_g \quad \text{// use the facts } \mathbf{G}\mathbf{R}_1 = \alpha\mathbf{G} \text{ (Eq. 4.4), } \mathbf{B}_g = \mathbf{B}_k\mathbf{R}_k \text{ (Eq. 4.6)}
\end{aligned}$$

The bound on the noise $\|\mathbf{e}_g\|$ is:

$$\|\mathbf{e}_g\| = \left\|\sum_{j=1}^{k} \left(\prod_{i=j+1}^{k} x_i\right) \mathbf{R}_j^T \mathbf{e}_j\right\| \le \left(1 + p + \ldots + p^{k-1}\right) \cdot \max_{j\in[k]} \left[\|\mathbf{R}_j^T\|_2 \cdot \|\mathbf{e}_j\|\right] \overset{\text{Lemma. 2.5}}{\le} \frac{p^k-1}{p-1} 2m \cdot \delta$$

This completes the first part of the lemma. In the second part of the lemma, the output matrix $\mathbf{B}_g$ is computed as follows:

$$\begin{aligned}
\mathbf{B}_g =& (\mathbf{A}\mathbf{S}_k - x_k\mathbf{G})\mathbf{R}_k \overset{\text{Eq. 4.5}}{=} \qquad \text{// by (4.5) we have } \mathbf{G}\mathbf{R}_k = -(\mathbf{A}\mathbf{S}_{k-1} - x_{k-1}\mathbf{G})\mathbf{R}_{k-1} \\
&= (\mathbf{A}\mathbf{S}_k\mathbf{R}_k + x_k\mathbf{A}\mathbf{S}_{k-1}\mathbf{R}_{k-1} - x_k \cdot x_{k-1}\mathbf{G}\mathbf{R}_{k-1}) \overset{\text{Eq. 4.5}}{=} \ldots \overset{\text{Eq. 4.5}}{=} \\
&= (\mathbf{A}\mathbf{S}_k\mathbf{R}_k + x_k\mathbf{A}\mathbf{S}_{k-1}\mathbf{R}_{k-1} + x_k \cdot x_{k-1}\mathbf{A}\mathbf{S}_{k-2}\mathbf{R}_{k-2} + \ldots + (-x_1 \cdots x_k\mathbf{G}\mathbf{R}_1)) \overset{\text{Eq. 4.4}}{=} \\
&= (\mathbf{A}\mathbf{S}_g - \alpha x_1 \cdots x_k\mathbf{G}) = (\mathbf{A}\mathbf{S}_g - g(\mathbf{x})\mathbf{G})
\end{aligned}$$

Moreover, the bound on the norm of $\mathbf{S}_g$ is:

$$\begin{aligned}
\|\mathbf{S}_g\|_2 =& \left\|\sum_{j=1}^{k} \left(\prod_{i=j+1}^{k} x_i\right) \mathbf{S}_j\mathbf{R}_j\right\|_2 \\
&\le \left(1 + p + \ldots + p^{k-1}\right) \max_{i\in[k]} \left(\|\mathbf{S}_i\|_2 \cdot \|\mathbf{R}_i\|_2\right) \overset{\text{Lemma. 2.5}}{\le} \frac{p^k-1}{p-1} 2m \cdot \max_{i\in[k]} \left(\|\mathbf{S}_i\|_2\right)
\end{aligned}$$

as required. ∎

4.2 Evaluation algorithms for circuits

We will now show how using the algorithms for single gates, that compute weighted additions and multiplications as described above, to build algorithms for the depth d, unbounded fan-in circuits.

Let $\{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ be a family of polynomial-size arithmetic circuits. For each $\mathcal{C} \in \mathcal{C}_\lambda$ we index the wires of $\mathcal{C}$ following the notation in [GVW13]. The input wires are indexed 1 to ℓ, the internal wires have indices $\ell+1, \ell+2, \dots, |\mathcal{C}|-1$ and the output wire has index $|\mathcal{C}|$, which also denotes the size of the circuit. Every gate is indexed as a tuple $(w_1, \dots, w_{k_w}, w)$ where k_w is the fan-in of the gate, $w_1, \dots, w_{k_w}$ are the incoming wire indices, and $w > \max\{w_1, \dots, w_k\}$ is the outgoing wire index. The gate computes the function $g_w : \mathbb{Z}_q^{k_w} \to \mathbb{Z}_q$ in $\mathcal{G}$, where $\mathcal{G}$ is defined in (4.1). We assume that all (but possibly one) of the input values to the multiplication gates are bounded by p which is smaller than scheme modulus q. The "fan-out wires" in the circuit are given a single number. That is, if the outgoing wire of a gate feeds into the input of multiple gates, then all these wires are indexed the same.

For some $\lambda \in \mathbb{N}$, define the family of functions $\mathcal{F} = \{f : f \text{ can be computed by some } \mathcal{C} \in \mathcal{C}_\lambda\}$. Again we will describe the three Eval algorithms together, but it is easy to see that they can be separated.

$$\begin{aligned}
&\mathsf{Eval}_{\mathrm{pk}}(f \in \mathcal{F},\ \vec{\mathbf{B}} \in (\mathbb{Z}_q^{n\times m})^\ell\ ) \longrightarrow \mathbf{B}_f\ \in \mathbb{Z}_q^{n\times m}\\
&\mathsf{Eval}_{\mathrm{ct}}(f \in \mathcal{F},\ \big((x_i, \mathbf{B}_i, \mathbf{c}_i)\big)_{i=1}^{\ell}\ ) \longrightarrow \mathbf{c}_f\ \in \mathbb{Z}_q^{m}\\
&\mathsf{Eval}_{\mathrm{sim}}(f \in \mathcal{F},\ \big((x_i, \mathbf{S}_i)\big)_{i=1}^{\ell},\ \mathbf{A}) \longrightarrow \mathbf{S}_f \in \mathbb{Z}_q^{m\times m}
\end{aligned}$$

Let f be computed by some circuit $\mathcal{C} \in \mathcal{C}_\lambda$, that has ℓ input wires. We construct the required matrices gate-by-gate: we construct the matrix for the output wire of every gate once the matrices on the input wires to this gate are computed.

For all $w \in [\mathcal{C}]$ denote the value that wire w carries when circuit $\mathcal{C}$ is evaluated on $\mathbf{x}$ to be x_w. Consider an arbitrary gate of fan-in k_w (we will omit the subscript w where it is clear from the context): $(w_1, \dots, w_k, w)$ that computes the function $g_w : \mathbb{Z}_q^k \to \mathbb{Z}_q$. Suppose we already computed $\mathbf{B}_{w_1}, \dots, \mathbf{B}_{w_k}$, $\mathbf{S}_{w_1}, \dots, \mathbf{S}_{w_k}$ and $\mathbf{c}_{w_1}, \dots, \mathbf{c}_{w_k}$, note that if $w_1, \dots, w_k$ are all in $\{1, 2, \dots, \ell\}$ then these matrices and vectors are the inputs of the corresponding Eval functions.

Using Eval algorithms described in Section 4.1, compute

$$\begin{aligned}
\mathbf{B}_w &= \mathsf{Eval}_{\mathrm{pk}}(g_w,\ (\mathbf{B}_{w_1}\ ,\ \dots\ ,\ \mathbf{B}_{w_k}))\\
\mathbf{c}_w &= \mathsf{Eval}_{\mathrm{ct}}(g_w,\ \big((x_{w_i}, \mathbf{B}_{w_i}, \mathbf{c}_{w_i})\big)_{i=1}^{k})\\
\mathbf{S}_w &= \mathsf{Eval}_{\mathrm{sim}}(g_w,\ \big((x_{w_i}, \mathbf{S}_{w_i})\big)_{i=1}^{k},\ \mathbf{A})
\end{aligned}$$

Output $\mathbf{B}_f := \mathbf{B}_{|\mathcal{C}|}$, $\mathbf{c}_f := \mathbf{c}_{|\mathcal{C}|}$, $\mathbf{S}_f := \mathbf{S}_{|\mathcal{C}|}$. Next we show that these outputs satisfy the required properties.

Lemma 4.3. *Let $\beta(m) = 2\frac{p^k-1}{p-1}m$. If $\mathbf{c}_i \in E_{\mathbf{s},\delta}(x_i, \mathbf{B}_i)$ for some $\mathbf{s} \in \mathbb{Z}_q^n$ and $\delta > 0$, then $\mathbf{c}_f \in E_{\mathbf{s},\Delta}(f(\mathbf{x}), \mathbf{B}_f)$ where $\Delta < (\beta(m))^d \cdot \delta$ and $\mathbf{B}_f = \mathsf{Eval}_{pk}(f,\ (\mathbf{B}_1, \dots, \mathbf{B}_\ell))$.*

Proof. By Lemma 4.1 and 4.2, after each level of the circuit the noise is multiplied by $\beta_{g_w}(m)$, which is upperbounded by $\beta(m)$ and the total number of levels is equal to the depth d of the circuit. The lemma follows. ∎

Lemma 4.4. *Let $\beta(m)$ be as defined in Lemma 4.3. If $\mathbf{S}_1, \dots, \mathbf{S}_\ell$ are random matrices in $\{\pm 1\}^{m\times m}$, then the output $\mathbf{S}_f$ satisfies $\mathbf{A}\mathbf{S}_f - f(\mathbf{x})\mathbf{G} = \mathbf{B}_f$ where $\|\mathbf{S}_f\|_2 \le (\beta(m))^d \cdot 20\sqrt{m}$ and $\mathbf{B}_f = \mathsf{Eval}_{pk}\big(f,\ (\mathbf{A}\mathbf{S}_1 - x_1\mathbf{G}, \dots, \mathbf{A}\mathbf{S}_\ell - x_\ell\mathbf{G})\big)$.*

Proof. Since the input $\mathbf{S}_i$ for $i \in [\ell]$ are random matrices in $\{\pm 1\}^{m\times m}$, by Lemma 2.3 for all $i \in [\ell]$, $\|\mathbf{S}_i\|_2 < 20\sqrt{m}$. By Lemma 4.1 and 4.2, after each level of the circuit the bound on $\mathbf{S}$ gets multiplied by at most $\beta(m)$, therefore after d levels, which is the depth of the circuit, the bound on the output matrix will be $\|\mathbf{S}_f\|_2 \le (\beta(m))^d \cdot 20\sqrt{m}$. The lemma follows. ∎

In summary, algorithms $(\mathsf{Eval}_{\mathrm{pk}}, \mathsf{Eval}_{\mathrm{ct}}, \mathsf{Eval}_{\mathrm{sim}})$ are $\alpha_{\mathcal{F}}$-ABE enabling for

$$\alpha_{\mathcal{F}}(n) = (\beta(m))^d \cdot 20\sqrt{m} = O\big((p^{k-1}m)^d\sqrt{m}\big), \quad \text{where} \quad m = \Theta(n \log q). \tag{4.7}$$

This is sufficient for polynomial depth arithmetic circuits as discussed in the introduction.

5 Extension: Polynomial gates

We can further reduce the depth of a given arithmetic circuit (and thereby shrink the required lattice sizes) by allowing the circuit to use more general gates than simple addition and multiplication. For example, the k-way OR gate polynomial can be implemented using a single gate.

Definition 5.1. An ℓ-variate polynomial is said to have *restricted arithmetic complexity* (ℓ, d, g) if it can be computed by a depth-d circuit that takes ℓ inputs $x_1, \ldots, x_\ell \in \mathbb{Z}_q$ and outputs a single $x \in \mathbb{Z}_q$. The circuit contains g gates, each of them is either a fan-in 2 addition gate or a fan-in 2 multiplication gate. Multiplication gates are further restricted to have one of their two inputs be one of the inputs to the circuit: $x_1, \ldots, x_\ell$.

We build the Eval algorithms for polynomials with complexity (ℓ, d, g) whose running time is proportional to g and that increase the magnitude of the noise in a given ciphertext by a factor of at most $O(p^d \cdot m)$, where p is the bound on all the intermediate values. Were we to directly use the Eval algorithms from the previous section on this polynomial, the magnitude of the noise would increase by $O((pm)^d)$ which is considerably larger, especially when p is small (e.g. $p = 1$).

We can build arithmetic circuits using polynomials with complexity (ℓ, d, g) as gates. Evaluating a depth D arithmetic circuit with such polynomial gates would increase the magnitude of the noise by at most a factor of $O((p^d \cdot m)^D)$. Again, if we were to simply treat the circuit as a standard arithmetic circuit with basic addition and multiplication gates the noise would instead grow as $O((pm)^{dD})$ which is larger.

Next we present ABE-enabling algorithms $\mathsf{Eval}_{\mathrm{pk}}, \mathsf{Eval}_{\mathrm{ct}}, \mathsf{Eval}_{\mathrm{sim}}$ for these enhanced polynomial gates with the noise bounds discussed in the previous paragraph. To support multiplication and addition of constants, we may assume that we have an extra 0-th input to the circuit that always carries the value 1. We present all three algorithms at the same time. Suppose that f is a polynomial with complexity (ℓ, d, g), then the three algorithms work as follows:

$\mathsf{Eval}_{\mathrm{pk}}(f,\ \vec{\mathbf{B}} \in (\mathbb{Z}_q^{n\times m})^\ell\ ) \longrightarrow \mathbf{B}_f\ \in \mathbb{Z}_q^{n\times m}$
$\mathsf{Eval}_{\mathrm{ct}}(f,\ \big((x_i, \mathbf{B}_i, \mathbf{c}_i)\big)_{i=1}^{\ell}\ ) \longrightarrow \mathbf{c}_f\ \in \mathbb{Z}_q^{m}$
$\mathsf{Eval}_{\mathrm{sim}}(f,\ \big((x_i, \mathbf{S}_i)\big)_{i=1}^{\ell},\ \mathbf{A}) \longrightarrow \mathbf{S}_f \in \mathbb{Z}_q^{m\times m}$

For each wire $w \in [|f|]$ (here $|f|$ denotes the total number of wires in the circuit and the notation of naming the wires is as described in Section 4.2) starting from the input wires and proceeding to the output we will construct the matrices $\mathbf{B}_w \in \mathbb{Z}_q^{n\times m}$, $\mathbf{S}_w \in \mathbb{Z}_q^{m\times m}$, $\mathbf{c}_w \in \mathbb{Z}_q^{m}$. Finally we output $\mathbf{B}_f = \mathbf{B}_{|f|}$, $\mathbf{S}_f = \mathbf{S}_{|f|}$, $\mathbf{c}_f = \mathbf{c}_{|f|}$. Consider an arbitrary gate and suppose that matrices on the input wires are computed, then to compute the matrices on the output wire do the following:

- Suppose the gate computes addition, has input wires w_1 and w_2 and output wire w. Then set the output matrices on wire w to be:

$$\mathbf{B}_w = \mathbf{B}_{w_1} + \mathbf{B}_{w_2}, \qquad \mathbf{S}_w = \mathbf{S}_{w_1} + \mathbf{S}_{w_2}, \qquad \mathbf{c}_w = \mathbf{c}_{w_1} + \mathbf{c}_{w_2}\ .$$

- Suppose the gate computes the multiplication by x_i for some $i \in [\ell]$, the input wires are u and i, the output wire is w. Then generate matrix $\mathbf{R} \in \mathbb{Z}_q^{m\times m}$ to satisfy $\mathbf{GR} = -\mathbf{B}_u$ by running $\mathbf{R} = \mathsf{SolveR}(\mathbf{G}, \mathbf{T}_G, -\mathbf{B}_u)$. Output

$$\mathbf{B}_w = \mathbf{B}_i\mathbf{R}, \qquad \mathbf{S}_w = \mathbf{S}_i\mathbf{R} + x_w\mathbf{S}_u, \qquad \mathbf{c}_w = x_i\mathbf{c}_u + \mathbf{R}^T\mathbf{c}_i \ .$$

Note that the amount of work required to run the Eval algorithms is proportional to the number of gates g in the circuit.

The following lemma shows that the noise in the output ciphertext grows by at most the factor of $O(p^d m)$, where p is the upper bound on the intermediate values in the circuit.

Lemma 5.2. *If $\mathbf{c}_i \in E_{\mathbf{s},\delta}(x_i, \mathbf{B}_i)$ for some $\mathbf{s} \in \mathbb{Z}_q^n$, $\delta > 0$ and the bound on the numbers $p \geq 2$, then for the polynomial f of complexity (ℓ, d, g) with $\beta_d = (1 + p + \ldots + p^d)\cdot 2m$ we have:*

- *$\mathbf{c}_f$ satisfies $\mathbf{c}_f \in E_{\mathbf{s},\Delta}(f(\mathbf{x}), \mathbf{B}_f)$ where $\mathbf{B}_f = \mathsf{Eval}_{pk}(f,\ (\mathbf{B}_1, \ldots, \mathbf{B}_\ell))$ and $\Delta < \beta_d(m)\cdot\delta$,*
- *$\mathbf{S}_f$ satisfies $\mathbf{AS}_f - f(\mathbf{x})\mathbf{G} = \mathbf{B}_f$ where $\mathbf{B}_f = \mathsf{Eval}_{pk}\big(f,\ (\mathbf{AS}_1 - x_1\mathbf{G}, \ldots, \mathbf{AS}_\ell - x_\ell\mathbf{G})\big)$ and $||\mathbf{S}_f||_2 \leq \beta_d(m)\cdot\gamma$ where $\gamma = \max_{i\in[\ell]} ||\mathbf{S}_i||_2$.*

Proof. We prove the lemma by induction.

- Consider an addition gate at level i with input wires w_1 and w_2 and output wire w. Suppose for $j \in [2]$, the noise in the ciphertexts $||\mathbf{e}_{w_j}|| \leq \beta_{i-1}(m)\delta$ and $||\mathbf{S}_{w_i}||_2 \leq \beta_{i-1}(m)\cdot\gamma$.
  - $\mathbf{c}_w = \mathbf{c}_{w_1} + \mathbf{c}_{w_2} = (x_{w_1}\mathbf{G} + \mathbf{B}_{w_1})^T s + \mathbf{e}_{w_1} + (x_{w_2}\mathbf{G} + \mathbf{B}_{w_2})^T s + \mathbf{e}_{w_2} = (x_w\mathbf{G} + \mathbf{B}_w)^T s + \mathbf{e}_w$
  - $||\mathbf{e}_w|| = ||\mathbf{e}_{w_1} + \mathbf{e}_{w_2}|| \leq ||\mathbf{e}_{w_1}|| + ||\mathbf{e}_{w_2}|| \leq (\beta_{i-1}(m) + \beta_{i-1}(m))\delta \leq \beta_i(m)\delta$
  - $\mathbf{B}_w = \mathbf{B}_{w_1} + \mathbf{B}_{w_2} = (\mathbf{AS}_{w_1} - x_{w_1}\mathbf{G}) + (\mathbf{AS}_{w_2} - x_{w_2}\mathbf{G}) = \mathbf{A}(\mathbf{S}_{w_1} + \mathbf{S}_{w_2}) - (x_{w_1} + x_{w_2})\mathbf{G} = \mathbf{AS}_w - x_w\mathbf{G}$
  - $||\mathbf{S}_w||_2 = ||\mathbf{S}_{w_1} + \mathbf{S}_{w_2}||_2 \leq ||\mathbf{S}_{w_1}||_2 + ||\mathbf{S}_{w_2}||_2 \leq (\beta_{i-1}(m) + \beta_{i-1}(m))\cdot\gamma \leq \beta_i(m)\cdot\gamma$.
- Consider a gate which has input wires u and $i \in [\ell]$, output wire w and which computes multiplication. Suppose $||\mathbf{e}_u|| \leq \beta_{i-1}(m)$ and $||\mathbf{S}_u||_2 \leq \beta_{i-1}(m)\cdot\gamma$, then the following holds
  - $\mathbf{c}_w = x_i\mathbf{c}_u + \mathbf{R}^T\mathbf{c}_i = x_i(x_u\mathbf{G} + \mathbf{B}_u)^T s + x_i\mathbf{e}_u + \mathbf{R}^T(x_i\mathbf{G} + \mathbf{B}_i)^T s + \mathbf{R}^T\mathbf{e}_i =$ $(x_w\mathbf{G} + x_i\cancel{(\mathbf{B}_g + \mathbf{GR})} + \mathbf{B}_w)^T s + \mathbf{e}_w$
  - $||\mathbf{e}_w||_2 = ||x_i\mathbf{e}_u + \mathbf{R}^T\mathbf{e}_i||_2 \leq p||\mathbf{e}_u||_2 + 2m||\mathbf{e}_i||_2 \leq (p\beta_{i-1}(m) + 2m)\delta \leq \beta_i(m)\cdot\delta$
  - $\mathbf{B}_w = \mathbf{B}_i\mathbf{R} = (\mathbf{AS}_i - x_i\mathbf{G})\mathbf{R} = \mathbf{AS}_i\mathbf{R} + x_i\mathbf{B}_u = \mathbf{AS}_i\mathbf{R} + x_i(\mathbf{AS}_u - x_u\mathbf{G}) =$ $\mathbf{A}(x_i\mathbf{S}_u + \mathbf{S}_i\mathbf{R}) - (x_i x_u)\mathbf{G} = \mathbf{AS}_w - x_w\mathbf{G}$
  - $||\mathbf{S}_w||_2 = ||x_i\mathbf{S}_u + \mathbf{S}_i\mathbf{R}||_2 \leq (p\beta_{i-1}(m) + 2m)\cdot\gamma \leq \beta_i(m)\cdot\gamma$.

as required. ■

Now combining Lemma 5.2 and lemmas analogous to Lemmas 4.3, 4.4 we can build an ABE system for a set of functions $\mathcal{F}$ which can be computed by depth D circuits with (k, d, g)-complexity gates. The bound function will then be

$$\alpha_{\mathcal{F}}(n) = (\beta_d(m))^D \ \cdot\ 20\sqrt{m} = O((p^d m)^D\sqrt{m}).$$

The time complexity of the Eval algorithms for circuit C that consists of (k, d, g)-complexity gates will be $O(g\cdot|C|)$.

5.1 Example applications for polynomial gates

Unbounded fan-in OR gate. Assuming that boolean inputs are interpreted as integers in $\{0, 1\}$, the OR gate of ℓ inputs can be computed with the following recursive formula:

$$\mathsf{OR}_{\ell+1}(x_1, \ldots, x_\ell, x_{\ell+1}) = x_{\ell+1} + (1 - x_{\ell+1}) \cdot \mathsf{OR}_\ell(x_1, \ldots, x_\ell), \quad \text{where} \quad \mathsf{OR}_1(x_1) = x_1.$$

It is easy to see that OR_ℓ has restricted complexity $(\ell, 3\ell, 3\ell)$, since at each of the ℓ iterations we do one multiplication by $x_{\ell+1}$ and two fan-in 2 additions. Therefore, by Lemma 5.2, an OR_ℓ gate increases the noise in the ciphertext by a factor of $O(\ell \cdot m)$.

If we were computing the OR_ℓ function with addition and multiplication gates as in Section 4.2, the most efficient way would be to use the De Morgan's law:

$$\mathsf{OR}_{\ell+1}(x_1, \ldots, x_\ell, x_{\ell+1}) = 1 - (1 - x_1)(1 - x_2) \ldots (1 - x_\ell).$$

This function can be computed with one level of ℓ fan-in-2 addition gates (to compute $(1 - x_i)$ for $i \in [\ell]$), one level of a single fan-in-ℓ multiplication gate (to compute $\prod_{i=1}^{\ell}(1 - x_i)$) and one more level of a single fan-in-2 addition gate. The noise then will grow by a factor of $O(\ell \cdot m^3)$, which will make the scheme 3 times less efficient.

The Fibonacci polynomial. Consider the following polynomial, defined for $\mathbf{x} \in [-p, p]^\ell$ using the following recurrence:

$$\Pi_1(\mathbf{x}) = x_1, \qquad \Pi_2(\mathbf{x}) = x_2$$
$$\Pi_{i+2}(\mathbf{x}) = \Pi_{i+1}(\mathbf{x}) + \Pi_i(\mathbf{x}) \cdot x_{i+2} \quad \text{for} \quad i \in \{1, \ldots, \ell - 2\}$$

If expanded, the number of monomials in Π_ℓ is equal to the ℓ-th Fibonacci number, which is exponential in ℓ. The degree of the polynomial is $\lfloor \frac{\ell}{2} \rfloor$. The recurrence shows that the restricted arithmetic complexity of this polynomial is $(\ell, \ell, 2\ell)$. Therefore, we can compute it with a single polynomial gate and, by Lemma 5.2, the growth in ciphertext noise will be proportional to $p^\ell \cdot m$.

We conjecture that computing this polynomial with a polynomial-size arithmetic circuit requires linear depth in ℓ. Therefore, the growth in ciphertext noise using the approach of Section 4.2 will be proportional to $(pm)^{O(\ell)}$ which is much worse.

6 Extension: Delegatable ABE

Our ABE easily extends to support full key delegation. We first sketch the main idea for adding key delegation and then describe the resulting ABE system.

In the ABE scheme from Section 3, a secret key for a function f is a matrix $\mathbf{R}_f$ that maps $(\mathbf{A}|\mathbf{B}_f)$ to some fixed matrix $\mathbf{D}$. Instead, we can give as a secret key for f a trapdoor (i.e. a short basis) $\mathbf{T}_\mathbf{F}$ for the matrix $\mathbf{F} = (\mathbf{A}|\mathbf{B}_f)$. The decryptor could use $\mathbf{T}_\mathbf{F}$ to generate the matrix $\mathbf{R}_f$ herself using algorithm SolveR. Now, for a given function g, to construct a secret key that decrypts whenever the attribute vector $\mathbf{x}$ satisfies $f(\mathbf{x}) = g(\mathbf{x}) = 0$ we extend the trapdoor for $\mathbf{F}$ into a trapdoor for $(\mathbf{F}|\mathbf{B}_g) = (\mathbf{A}|\mathbf{B}_f|\mathbf{B}_g)$ using algorithm $\mathsf{ExtendRight}$. We give a randomized version of this trapdoor as a delegated secret key for $f \wedge g$. Intuitively this trapdoor can only be used to decrypt if the decryptor can obtain the ciphertexts under matrices $\mathbf{B}_f$ and $\mathbf{B}_g$ which by security of ABE can only happen if the ciphertexts was created for an attribute vector $\mathbf{x}$ satisfying $f(\mathbf{x}) = g(\mathbf{x}) = 0$.

The top level secret key generated by KeyGen is a $(2m \times 2m)$ matrix in $\mathbb{Z}$. After k delegations the secret key becomes a $((k+1)m \times (k+1)m)$ matrix. Hence, the delegated key grows quadratically with the number of delegations k.

Definition. Formally, a delegatable attribute-based encryption (DABE) scheme is an attribute-based encryption scheme that in addition to four standard algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{E}, \mathsf{D})$ offers a delegation algorithm $\mathsf{Delegate}$. Consider a ciphertext c encrypted for index vector $\mathbf{x}$. The algorithm KeyGen returns the secret key sk_f for function f and this key allows to decrypt the ciphertext c only if $f(\mathbf{x}) = 0$. The delegation algorithm given the key sk_f and a function g outputs a "delegated" secret key that allows to decrypt the ciphertext only if $f(\mathbf{x}) = 0 \wedge g(\mathbf{x}) = 0$, which is a more restrictive condition. The idea can be generalized to arbitrary number of delegations:

$\mathsf{Delegate}(\mathsf{mpk}, sk_{f_1,\ldots,f_k}, f_{k+1}) \to \mathsf{sk}_{f_1,\ldots,f_{k+1}}$:

Takes as input the master secret key msk, the function $f_{k+1} \in \mathcal{F}$ and the secret key $sk_{f_1,\ldots,f_k}$ that was generated either by algorithm KeyGen, if $k = 1$ or by algorithm $\mathsf{Delegate}$, if $k > 1$. Outputs a secret key $\mathsf{sk}_{f_1,\ldots,f_{k+1}}$.

Correctness. We require the scheme to give a correct ABE as discussed in Section 2.1 and in addition to satisfy the following requirement. For all sequence of functions $f_1, \ldots, f_k \in \mathcal{F}$, a message $m \in \mathcal{M}$ and index $\mathbf{x} \in \mathbb{Z}_q^\ell$, s.t. $f_1(\mathbf{x}) = 0 \wedge \ldots \wedge f_k(\mathbf{x}) = 0$ it holds that $\mu = \mathsf{D}(\mathsf{sk}_{f_1,\ldots,f_k}, (\mathbf{x}, c))$ with an overwhelming probability over the choice of $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \ell)$, $c \leftarrow \mathsf{E}(\mathsf{mpk},\ \mathbf{x} \in \mathcal{X}^\ell,\ \mu)$, $\mathsf{sk}_{f_1} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f_1)$ and $sk_{f_1,\ldots,f_{i+1}} \leftarrow \mathsf{Delegate}(\mathsf{mpk}, sk_{f_1,\ldots,f_i}, f_{i+1})$ for all $i \in [k]$.

Security. The security of DABE schemes is derived from definition of selective security for ABE scheme (see Definition 2.1) by providing the adversary with access to a key-generation oracle.

Definition 6.1 (Selectively-secure DABE)**.** A DABE scheme $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{E}, \mathsf{D}, \mathsf{Delegate})$ for a class of functions $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ with $\ell = \ell(\lambda)$ inputs over an index space $\mathcal{X}^\ell = \{\mathcal{X}_\lambda^\ell\}_{\lambda\in\mathbb{N}}$ and a message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ is *selectively secure* if for any probabilistic polynomial-time adversary $\mathcal{A}$, there exists a negligible function $\nu(\lambda)$ such that

$$\mathbf{Adv}^{\mathsf{sDABE}}_{\Pi,\mathcal{A}}(\lambda) \overset{\mathsf{def}}{=} \left|\Pr\left[\mathsf{Expt}^{(0)}_{\mathsf{sDABE},\Pi,\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Expt}^{(1)}_{\mathsf{sDABE},\Pi,\mathcal{A}}(\lambda) = 1\right]\right| \leq \nu(\lambda),$$

where for each $b \in \{0, 1\}$ and $\lambda \in \mathbb{N}$ the experiment $\mathsf{Expt}^{(b)}_{\mathsf{sDABE},\Pi,\mathcal{A}}(\lambda)$ is defined as follows:

1. $(\mathbf{x}^*, \mathsf{state}_1) \leftarrow \mathcal{A}(\lambda)$, where $\mathbf{x}^* \in \mathcal{X}^\ell$.
2. $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(\lambda)$.
3. $(\mu_0, \mu_1, \mathsf{state}_2) \leftarrow \mathcal{A}^{\mathsf{KG}(\mathsf{msk},\mathbf{x}^*,\cdot)}(\mathsf{mpk}, \mathsf{state}_1)$, where $\mu_0, \mu_1 \in \mathcal{M}_\lambda$.
4. $c^* \leftarrow \mathsf{E}(\mathsf{mpk}, \mathbf{x}^*, \mu_b)$.
5. $b' \leftarrow \mathcal{A}^{\mathsf{KG}(\mathsf{msk},\mathbf{x}^*,\cdot)}(c^*, \mathsf{state}_2)$.
6. Output $b' \in \{0, 1\}$.

Here the key-generation oracle $\mathsf{KG}(\mathsf{msk}, \mathbf{x}^*, (f_1, \ldots, f_k))$ takes a set of functions $f_1, \ldots, f_k \in \mathcal{F}$ and returns the secret key $\mathsf{sk}_{f_1,\ldots,f_k}$ if $f_1(\mathbf{x}^*) \neq 0 \vee \ldots \vee f_k(\mathbf{x}^*) \neq 0$ and otherwise the oracle returns $\perp$. The secret key $\mathsf{sk}_{f_1,\ldots,f_k}$ is defined as follows: $\mathsf{sk}_{f_1} = \mathsf{KeyGen}(\mathsf{msk}, f_1)$ and for all $i \in \{2, \ldots, k\}$ $\quad \mathsf{sk}_{f_1,\ldots,f_i} = \mathsf{Delegate}(\mathsf{mpk}, \mathsf{sk}_{f_1,\ldots,f_{i-1}}, f_i)$.

6.1 A delegatable ABE scheme from LWE

The DABE scheme will be almost identical to ABE scheme described earlier, except as a secret key for function f instead of recoding matrix from $(\mathbf{A}|\mathbf{B}_f)$ to $\mathbf{D}$ we will give the rerandomized trapdoor for $(\mathbf{A}|\mathbf{B}_f)$ and then the decryptor can build the recoding matrix to $\mathbf{D}$ himself.

$\mathsf{KeyGen}(\mathsf{msk}, f)$:

Let $\mathbf{B}_f = \mathsf{Eval}_{\mathrm{pk}}(f,\ (\mathbf{B}_1, \dots, \mathbf{B}_\ell))$.

Build the basis $\mathbf{T}_f$ for $\mathbf{F} = (\mathbf{A}|\mathbf{B}_f) \in \mathbb{Z}_q^{n\times 2m}$ as $\mathbf{T}_f \leftarrow \mathsf{RandBasis}(\mathbf{F}, \mathsf{ExtendRight}(\mathbf{A}, \mathbf{T}_A, \mathbf{B}_f), \sigma)$, for big enough $\sigma = \|\mathbf{T}_\mathbf{A}\|_{\mathsf{GS}} \cdot \omega(\sqrt{\log m})$ (we will set σ as before: $\sigma = \omega(\alpha_\mathcal{F} \cdot \sqrt{\log m})$).

Output $\mathsf{sk}_f := \mathbf{T}_f$.

$\mathsf{Delegate}(\mathsf{mpk}, \mathsf{sk}_{f_1,\dots,f_k}, g)$:

Parse the secret key $\mathsf{sk}_{f_1,\dots,f_k}$ as a matrix $\mathbf{T}_k \in \mathbb{Z}_q^{(k+1)m\times(k+1)m}$ which is a trapdoor for the matrix $(\mathbf{A}|\mathbf{B}_{f_1}|\dots|\mathbf{B}_{f_k}) \in \mathbb{Z}_q^{n\times(k+1)m}$.

Let $\mathbf{B}_g = \mathsf{Eval}_{\mathrm{pk}}(g,\ (\mathbf{B}_1, \dots, \mathbf{B}_\ell))$.

Build the basis for matrix $\mathbf{F} = (\mathbf{A}|\mathbf{B}_{f_1}|\dots|\mathbf{B}_{f_k}|\mathbf{B}_g) \in \mathbb{Z}_q^{n\times(k+2)m}$:

$$\mathbf{T}_{k+1} \ = \ \mathsf{RandBasis}(\mathbf{F}, \mathsf{ExtendRight}((\mathbf{A}|\mathbf{B}_{f_1}|\dots|\mathbf{B}_{f_k}), \mathbf{T}_k, \mathbf{B}_g), \sigma_k).$$

Here $\sigma_k = \sigma \cdot (\sqrt{m\log m})^k$. Output $\mathsf{sk}_{f_1,\dots,f_k,g} := \mathbf{T}_{k+1} \in \mathbb{Z}_q^{(k+2)m\times(k+2)m}$. Note that the size of the key grows quadratically with the number of delegations k.

$\mathsf{Dec}\big(\mathsf{sk}_{f_1,\dots,f_k},\ (\mathbf{x}, \mathbf{c})\big)$: If $f_1(\mathbf{x}) \neq 0 \vee \dots \vee f_k(\mathbf{x}) \neq 0$ output $\perp$.

Otherwise parse the secret key $\mathsf{sk}_{f_1,\dots,f_k}$ as a matrix $\mathbf{T}_k \in \mathbb{Z}_q^{(k+1)m\times(k+1)m}$ which is a trapdoor for the matrix $(\mathbf{A}|\mathbf{B}_{f_1}|\dots|\mathbf{B}_{f_k})$.

Run $\mathbf{R} \leftarrow \mathsf{SampleD}(\ (\mathbf{A}|\mathbf{B}_{f_1}|\dots|\mathbf{B}_{f_k}),\ \mathbf{T}_k,\ \mathbf{D},\ \sigma_k)$ to generate a low-norm matrix $\mathbf{R} \in \mathbb{Z}_q^{(k+1)m\times m}$ such that $(\mathbf{A}|\mathbf{B}_{f_1}|\dots|\mathbf{B}_{f_k}) \cdot \mathbf{R} = \mathbf{D}$.

For all $j \in [k]$, compute $(\mathbf{c}_{in}, \mathbf{c}_j, \mathbf{c}_{out}) = \mathsf{Eval}_{\mathrm{ct}}\big(\{(x_i, \mathbf{B}_i)\}_{i=1}^{\ell},\ \mathbf{c},\ f_i\big) \quad \in \mathbb{Z}_q^{3m}$. Note that $\mathbf{c}_{in}$ and $\mathbf{c}_{out}$ stay the same across all $i \in [k]$.

Let $\mathbf{c}' = (\mathbf{c}_{in}|\mathbf{c}_1|\dots|\mathbf{c}_k) \in \mathbb{Z}_q^{(k+1)m}$. Output $\mu = \mathsf{Round}(\mathbf{c}_{out} - \mathbf{R}^T\mathbf{c}')$.

Correctness. To show correctness, note that when $f_1(\mathbf{x}) = 0 \wedge \dots \wedge f_k(\mathbf{x}) = 0$ we know by the requirement on $\mathsf{Eval}_{\mathrm{ct}}$ that the resulting ciphertexts $\mathbf{c}_{f_i} \in \mathsf{E}_{\mathbf{s},\Delta}(0, \mathbf{B}_{f_i})$ for $\forall i \in [k]$. Consequently,

$$(\mathbf{c}_{in}|\mathbf{c}_{f_1}|\dots|\mathbf{c}_{f_k}) = (\mathbf{A}|\mathbf{B}_{f_1}|\dots|\mathbf{B}_{f_k})^T\mathbf{s} + \mathbf{e}' \qquad \text{where} \qquad ||\mathbf{e}'|| < k\Delta + \chi_{max} < (k\alpha_\mathcal{F} + 1)\chi_{\max}.$$

We know that $(\mathbf{A}|\mathbf{B}_{f_1}|\dots|\mathbf{B}_{f_k}) \cdot \mathbf{R} = \mathbf{D}$ and $||\mathbf{R}^T||_2 < (k+1)m\sigma_k$ with overwhelming probability by Lemma 2.3. Therefore

$$\mathbf{c}_{out} - \mathbf{R}^\mathsf{T}\mathbf{c}'_f = (\mathbf{D}^\mathsf{T}\mathbf{s} + \mathbf{e}_1) - (\mathbf{D}^\mathsf{T}\mathbf{s} + \mathbf{R}^\mathsf{T}\mathbf{e}') = \mathbf{e}_1 - \mathbf{R}^\mathsf{T}\mathbf{e}' \ .$$

Finally,

$$\|\mathbf{e}_1 - \mathbf{R}^\mathsf{T}\mathbf{e}'\| \leq \chi_{\max} + (k+1)m\sigma_k \cdot (\alpha_\mathcal{F} + 1)\chi_{\max} \leq (k+2)\alpha_\mathcal{F}^2 \cdot \chi_{\max} \cdot m^{k/2+1}$$

with overwhelming probability. The bound on $\alpha_\mathcal{F}$: $\alpha_\mathcal{F}^2 m^{k/2+1} < \frac{1}{4(k+2)} \cdot (q/\chi_{\max})$ ensures that this quantity is less than $q/4$ thereby ensuring correct decryption of all bits of $\mu \in \{0,1\}^m$.

Security. The security game is the same as the security game for ABE, described in Section 3, except in Game 2 we need to answer delegated key queries. Consider a private key query $\mathsf{sk}_{f_1,\dots,f_k}$, where $f_1, \dots, f_k \in \mathcal{F}$. This query is only allowed when $f_1(\mathbf{x}^*) \neq 0 \vee \dots \vee f_k(\mathbf{x}^*) \neq 0$. Without loss of generality, assume that $f_1(\mathbf{x}^*) = 0 \wedge \dots \wedge f_{k-1}(\mathbf{x}^*) = 0$ and $f_k(\mathbf{x}^*) \neq 0$. Indeed for all other cases, the adversary may ask for the key for a smaller sequence of functions and delegate herself. The key generation oracle for all $i \in [k]$ computes $\mathbf{B}_{f_i} = \mathsf{Eval}_{\mathrm{pk}}\big(f_i,\ (\mathbf{B}_1, \dots, \mathbf{B}_\ell)\big)$ and needs to produce a trapdoor $\mathbf{T}_k \in \mathbb{Z}^{(k+1)m\times(k+1)m}$ for the matrix $(\mathbf{A}|\mathbf{B}_{f_1}|\dots|\mathbf{B}_{f_k}) \in \mathbb{Z}_q^{n\times(k+1)m}$.

To do so the key generation oracle does:

- Run $\mathbf{S}_{f_k} \leftarrow \mathsf{Eval}_{\mathrm{sim}}(f_k,\ \big((x_i^*, \mathbf{S}_i^*)\big)_{i=1}^{\ell},\ \mathbf{A})$ and obtains a low-norm matrix $\mathbf{S}_{f_k} \in \mathbb{Z}_q^{m\times m}$ such that $\mathbf{A}\mathbf{S}_{f_k} - f_k(\mathbf{x}^*)\mathbf{G} = \mathbf{B}_{f_k}$. By definition of $\mathsf{Eval}_{\mathrm{sim}}$ we know that $\|\mathbf{S}_{f_k}\|_2 \le \alpha_{\mathcal{F}}$.

- Let $\mathbf{F} = (\mathbf{A}|\mathbf{B}_{f_1}|\dots|\mathbf{B}_{f_k}) = (\mathbf{A}|\mathbf{B}_{f_1}|\dots|\mathbf{B}_{f_{k-1}}|\mathbf{A}\mathbf{S}_{f_k} - y^*\mathbf{G})$. Because $y^* \neq 0$ the key generation oracle can obtain a trapdoor $\mathbf{T}_{(\mathbf{A}|\mathbf{B}_{f_k})}$ by running

$$\mathbf{T}_{(\mathbf{A}|\mathbf{B}_{f_k})} \leftarrow \mathsf{ExtendLeft}(y^*\mathbf{G}, \mathbf{T}_{\mathbf{G}}, \mathbf{A}, \mathbf{S}_{f_k})$$

  And then produce $\mathbf{T}_{(\mathbf{A}|\mathbf{B}_{f_k}|\mathbf{B}_{f_1}|\dots|\mathbf{B}_{f_{k-1}})}$ by running

$$\mathbf{T}_{(\mathbf{A}|\mathbf{B}_{f_k}|\mathbf{B}_{f_1}|\dots|\mathbf{B}_{f_{k-1}})} \leftarrow \mathsf{ExtendRight}(\mathbf{G}, \mathbf{T}_{\mathbf{G}}, (\mathbf{B}_{f_1}|\dots|\mathbf{B}_{f_{k-1}}))$$

  Now we can switch the rows of the matrix $\mathbf{T}_{(\mathbf{A}|\mathbf{B}_{f_k}|\mathbf{B}_{f_1}|\dots|\mathbf{B}_{f_{k-1}})}$ to get the matrix $\mathbf{T}_F$, which is a trapdoor for $(\mathbf{A}|\mathbf{B}_{f_1}|\dots|\mathbf{B}_{f_k})$. This operation, as well as $\mathsf{ExtendRight}$ function (according to Lemma 2.2, part 2) does not change the Gram-Schmidt norm of the basis, therefore this trapdoor satisfies

$$\|\mathbf{T}_{\mathbf{F}}\|_{\mathsf{GS}} \le \|\mathbf{T}_{\mathbf{G}}\|_{\mathsf{GS}} \cdot \|\mathbf{S}_{f_k}\|_2 \le \sqrt{5}\alpha_{\mathcal{F}}(n)$$

  where the bound on $\|\mathbf{T}_{\mathbf{G}}\|_{\mathsf{GS}}$ is from Lemma 2.2 (part 4).

- Finally, it responds with rerandomized trapdoor $\mathbf{T}_k = \mathsf{RandBasis}(\mathbf{F}, \mathbf{T}_{\mathbf{F}}, \sigma_k)$.
  By definition of $\mathsf{RandBasis}$ we know that $\mathbf{T}_k$ is distributed as $\mathcal{D}_{\sigma_k}(\Lambda_q^{\mathbf{F}}(\mathbf{F}))$ as required. Indeed $\sigma_k = \|\mathbf{T}_{\mathbf{F}}\|_{\mathsf{GS}} \cdot \omega(\sqrt{\log m})$ as needed for algorithm $\mathsf{RandBasis}$ in Lemma 2.4 (part 3).

7 Conclusions and open problems

We presented an ABE for arithmetic circuits with short secret keys whose security is based on the LWE problem. At the heart of our construction is a method for transforming a noisy vector of the form $\mathbf{c} = (\mathbf{A}|x_1\mathbf{G} + \mathbf{B}_1|\cdots|x_\ell\mathbf{G} + \mathbf{B}_\ell)^{\mathsf{T}}\mathbf{s} + \mathbf{e}$ into a vector $(\mathbf{A}|y\mathbf{G} + \mathbf{B}_f)^{\mathsf{T}}\mathbf{s} + \mathbf{e}_f$ where $y = f(x_1, \dots, x_\ell)$ and $\mathbf{e}_f$ is not much longer than $\mathbf{e}$. The short decryption key sk_f provides a way to decrypt when $y = 0$. We refer to this property as a *public-key homomorphism* and expect it to find other applications.

Natural open problems that remain are a way to provide adaptive security from LWE with a polynomial-time reduction. It would also be useful to construct an efficient ABE for arithmetic circuits where multiplication gates can handle inputs as large as the modulus q.

Acknowledgements. This work was supported by NSF, the DARPA PROCEED program, an AFO SR MURI award, a grant from ONR, an IARPA project provided via DoI/NBC, and by Samsung. Opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA or IARPA.

References

[ABB10] S. Agrawal, D. Boneh, and X. Boyen. Efficient lattice (H)IBE in the standard model. In *Advances in Cryptology – EUROCRYPT '10*, pages 553–572, 2010.

[ABV$^+$12] S. Agrawal, X. Boyen, V. Vaikuntanathan, P. Voulgaris, and H. Wee. Functional encryption for threshold functions (or fuzzy ibe) from lattices. In *Public Key Cryptography*, pages 280–297, 2012.

[AFV11] S. Agrawal, D. M. Freeman, and V. Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In *Advances in Cryptology – ASIACRYPT '11*, pages 21–40, 2011.

[Ajt99] M. Ajtai. Generating hard instances of the short basis problem. In *Proceedings of the 26th International Colloquium on Automata, Languages and Programming*, pages 1–9, 1999.

[AP09] J. Alwen and C. Peikert. Generating shorter bases for hard random lattices. In *STACS*, pages 75–86, 2009.

[Bab86] L. Babai. On lovsz lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986.

[BB11] D. Boneh and X. Boyen. Efficient selective identity-based encryption without random oracles. *Journal of Cryptology*, 24(4):659–693, 2011.

[BF03] D. Boneh and M. K. Franklin. Identity-based encryption from the Weil pairing. *SIAM Journal on Computing*, 32(3):586–615, 2003. Preliminary version in *Advances in Cryptology – CRYPTO '01*, pages 213–229, 2001.

[Boy13] X. Boyen. Attribute-based functional encryption on lattices. In *TCC*, pages 122–142, 2013.

[BSW11] D. Boneh, A. Sahai, and B. Waters. Functional encryption: Definitions and challenges. In *Proceedings of the 8th Theory of Cryptography Conference*, pages 253–273, 2011.

[BW07] D. Boneh and B. Waters. Conjunctive, subset, and range queries on encrypted data. In *Proceedings of the 4th Theory of Cryptography Conference*, pages 535–554, 2007.

[CHK$^+$10] D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert. Bonsai trees, or how to delegate a lattice basis. In *Advances in Cryptology – EUROCRYPT '10*, pages 523–552, 2010.

[Coc01] C. Cocks. An identity based encryption scheme based on quadratic residues. In *Proceedings of the 8th IMA International Conference on Cryptography and Coding*, pages 360–363, 2001.

[GGH$^+$13a] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. Cryptology ePrint Archive, Report 2013/451, 2013.

[GGH$^+$13b] S. Garg, C. Gentry, S. Halevi, A. Sahai, and B. Waters. Attribute-based encryption for circuits from multilinear maps. In *CRYPTO (2)*, pages 479–499, 2013.

[GGS$^+$13] S. Garg, C. Gentry, A. Sahai, and B. Waters. Witness encryption and its applications. In *STOC*, pages 467–476, 2013.

[GKP$^+$13] S. Goldwasser, Y. T. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich. Reusable garbled circuits and succinct functional encryption. In *STOC*, pages 555–564, 2013.

[GPS$^+$06] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM Conference on Computer and Communications Security*, pages 89–98, 2006.

[GPV08] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the 40th Annual ACM Symposium on Theory of computing*, pages 197–206, 2008.

[GVW13] S. Gorbunov, V. Vaikuntanathan, and H. Wee. Attribute-based encryption for circuits. In *Proceedings of the 45t Annual ACM Symposium on Theory of Computing*, pages 545–554, 2013.

[HW13] S. Hohenberger and B. Waters. Attribute-based encryption with fast decryption. In *Public Key Cryptography*, pages 162–179, 2013.

[KSW08] J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *Advances in Cryptology – EUROCRYPT '08*, pages 146–162, 2008.

[LOS$^+$10] A. B. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *EUROCRYPT*, pages 62–91, 2010.

[LPR$^+$05] A. Litvak, A. Pajor, M. Rudelson, and N. Tomczak-Jaegermann. Smallest singular value of random matrices and geometry of random polytopes. *Advances in Mathematics*, 195(2):491–523, 2005.

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

[MP12] D. Micciancio and C. Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In D. Pointcheval and T. Johansson, editors, *Advances in Cryptology EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 700–718. Springer Berlin Heidelberg, 2012.

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

[Pei09] C. Peikert. Public-key cryptosystems from the worst-case shortest vector problem. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, pages 333–342, 2009.

[PRV12] B. Parno, M. Raykova, and V. Vaikuntanathan. How to delegate and verify in public: Verifiable computation from attribute-based encryption. In *TCC*, pages 422–439, 2012.

[PTM$^+$06] M. Pirretti, P. Traynor, P. McDaniel, and B. Waters. Secure attribute-based systems. In *ACM Conference on Computer and Communications Security*, pages 99–112, 2006.

[Reg05] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 84–93, 2005.

[Sha84] A. Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology – CRYPTO '84*, pages 47–53, 1984.

[Sho08] V. Shoup. A Computational Introduction to Number Theory and Algebra, second edition. Cambridge University Press, 2008.

[SW05] A. Sahai and B. Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, pages 457–473, 2005.

[Wat12] B. Waters. Functional encryption for regular languages. In *Advances in Cryptology – CRYPTO '12*, pages 218–235, 2012.

A Key-Homomorphic Public-Key Encryption (KH-PKE): Definition

To help explain our ABE construction for arithmetic circuits we first describe a slightly more general mechanism we call *key-homomorphic public-key encryption* (KH-PKE). Such systems are public-key encryption schemes that are homomorphic with respect to the public encryption key. We show below that a key-policy ABE arises trivially from such a system.

Let $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$ and $\mathsf{PKT} = \{\mathsf{PKT}_\lambda\}_{\lambda\in\mathbb{N}}$ be sequences of finite sets. Public keys in a KH-PKE are pairs $(x, \mathsf{pkt}) \in \mathcal{X}_\lambda \times \mathsf{PKT}_\lambda$. We call x the "value" and pkt the "public-key tag." All such pairs are valid public keys as are tuples of pairs $((x_1, \mathsf{pkt}_1), \ldots, (x_\ell, \mathsf{pkt}_\ell))$ for some $\ell > 0$. To simplify the notation we often drop the subscript λ and simply refer to sets $\mathcal{X}$ and PKT.

A KH-PKE is defined with respect to some family of multi-variate functions $\mathcal{F} = \{f : \mathcal{X}^\ell \to \mathcal{X}\}$ for some $\ell > 0$. In our ABE construction we set $\mathcal{X} = \mathbb{Z}/q\mathbb{Z}$ for some q and let $\mathcal{F}$ be the set of functions on $\mathbb{Z}_q$ computable by polynomial size arithmetic circuits.

Now, A KH-PKE for a family of functions $\mathcal{F}$ comprises five PPT algorithms:

$\mathsf{Setup}_{\mathsf{KH}}(1^\lambda) \to (\mathsf{mpk}_{\mathrm{KH}}, \mathsf{msk}_{\mathrm{KH}})$: outputs a master secret key $\mathsf{msk}_{\mathrm{KH}}$ and public parameters $\mathsf{mpk}_{\mathrm{KH}}$.

$\mathsf{KeyGen}_{\mathsf{KH}}\big(\mathsf{msk}_{\mathrm{KH}},\ (x, \mathsf{pkt})\big) \to \mathsf{sk}$: outputs a decryption key for the public key $(x, \mathsf{pkt}) \in \mathcal{X} \times \mathsf{PKT}$.

$\mathsf{E}_{\mathsf{KH}}\big(\mathsf{mpk}_{\mathrm{KH}},\ \{(x_i, \mathsf{pkt}_i)\}_{i=1}^{\ell},\ \mu\big) \longrightarrow c$: encrypts a message μ to the public-key $\{(x_i, \mathsf{pkt}_i)\}_{i=1}^{\ell}$.

$\mathsf{D}_{\mathsf{KH}}(\mathsf{sk}, c)$: decrypts a ciphertext c with key sk.

Eval : key-homomorphism is captured by two *deterministic* algorithms: $\mathsf{Eval}_{\mathrm{pk}}$ and $\mathsf{Eval}_{\mathrm{ct}}$. For $x_1, \ldots, x_\ell \in \mathcal{X}$ and $\mathsf{pkt}_1, \ldots, \mathsf{pkt}_\ell \in \mathsf{PKT}$, ciphertext c, and function $f : \mathcal{X}^\ell \to \mathcal{X} \in \mathcal{F}$ these algorithms do:

$$\mathsf{Eval}_{\mathrm{pk}}\big(f,\ (\mathsf{pkt}_i)_{i=1}^{\ell}\big) \longrightarrow \mathsf{pkt}_f \qquad ; \qquad \mathsf{Eval}_{\mathrm{ct}}\big(f,\ \{(x_i, \mathsf{pkt}_i)\}_{i=1}^{\ell},\ c\big) \longrightarrow c_f$$

If c is an encryption of message μ under public-key $\{(x_i, \mathsf{pkt}_i)\}_{i=1}^{\ell}$ then c_f is an encryption of μ under public key (y, pkt_f) where $y = f(x_1, \ldots, x_\ell)$.

Algorithm $\mathsf{Eval}_{\mathrm{ct}}$ captures the key-homomorphic property of the system: a ciphertext c encrypted with key $\{(x_i, \mathsf{pkt}_i)\}_{i=1}^{\ell}$ is translated to a ciphertext c_f encrypted under key $\big(f(x_1, \ldots, x_\ell),\ \mathsf{pkt}_f\big)$.

Correctness. The key-homomorphic property is stated formally in the following requirement: For all $(\mathsf{mpk}_{\mathrm{KH}}, \mathsf{msk}_{\mathrm{KH}})$ output by Setup, all messages μ, and all $f \in \mathcal{F}$:

If $\quad c \leftarrow \mathsf{E}_{\mathsf{KH}}\big(\mathsf{mpk}_{\mathrm{KH}},\ \{(x_i, \mathsf{pkt}_i)\}_{i=1}^{\ell},\ \mu\big), \quad y = f(x_1, \ldots, x_\ell), \quad \mathsf{pkt}_f = \mathsf{Eval}_{\mathrm{pk}}\big(f,\ \{\mathsf{pkt}_i\}_{i=1}^{\ell}\big),$

$\quad c_f = \mathsf{Eval}_{\mathrm{ct}}\big(f,\ \{(x_i, \mathsf{pkt}_i)\}_{i=1}^{\ell},\ c\big), \quad \mathsf{sk} \leftarrow \mathsf{KeyGen}_{\mathsf{KH}}(\mathsf{msk}_{\mathrm{KH}}, (y, \mathsf{pkt}_f))$

Then $\mathsf{D}_{\mathsf{KH}}(\mathsf{sk}, c_f) = \mu$.

An ABE from a KH-PKE. A KH-PKE for a family of functions $\mathcal{F} = \{f : \mathcal{X}^\ell \to \mathcal{X}\}$ immediately gives a key-policy ABE. Indexes for the ABE are ℓ-tuples over $\mathcal{X}$ and the supported key-policies are functions in $\mathcal{F}$. The ABE system works as follows:

$\mathsf{Setup}(1^\lambda, \ell)$: Run $\mathsf{Setup}_{\mathsf{KH}}(1^\lambda)$ to get public parameters $\mathsf{mpk}_{\mathrm{KH}}$ and master secret $\mathsf{msk}_{\mathrm{KH}}$. Choose ℓ random tags $\mathsf{pkt}_1, \ldots, \mathsf{pkt}_\ell$ in PKT and output the ABE public parameters and master secret:

$$\mathsf{mpk} = \big(\mathsf{mpk}_{\mathrm{KH}},\ (\mathsf{pkt}_1^*, \ldots, \mathsf{pkt}_\ell^*)\ \big) \quad ; \quad \mathsf{msk} = \big(\mathsf{msk}_{\mathrm{KH}},\ (\mathsf{pkt}_1^*, \ldots, \mathsf{pkt}_\ell^*)\ \big)$$

$\mathsf{KeyGen}(\mathsf{msk}, f)$: Let $\mathsf{pkt}_f = \mathsf{Eval}_{\mathrm{pk}}\big(f,\ (\mathsf{pkt}_i^*)_{i=1}^\ell\big)$. output $\mathsf{sk}_f \leftarrow \mathsf{KeyGen}_{\mathsf{KH}}\big(\mathsf{msk}_{\mathrm{KH}},\ (0, \mathsf{pkt}_f)\big)$. We assume sk_f also implicitly includes mpk. Note that the size of the secret key sk_f is independent of the complexity of the function f.

$\mathsf{E}(\mathsf{mpk},\ \mathbf{x} \in \mathcal{X}^\ell,\ \mu)$: output $(\mathbf{x}, c)$ where $c \leftarrow \mathsf{E}_{\mathsf{KH}}\big(\mathsf{mpk}_{\mathrm{KH}},\ \{(x_i, \mathsf{pkt}_i^*)\}_{i=1}^\ell,\ \mu\big)$ and $\mathbf{x} = (x_1, \ldots, x_\ell)$.

$\mathsf{D}\big(\mathsf{sk}_f,\ (\mathbf{x}, c)\big)$: if $f(\mathbf{x}) = 0$ set $c_f = \mathsf{Eval}_{\mathrm{ct}}\big(f,\ \{(x_i, \mathsf{pkt}_i^*)\}_{i=1}^\ell,\ c\big)$ and output $\mathsf{D}_{\mathsf{KH}}(\mathsf{sk}_f, c_f)$. Note that c_f is the encryption of the plaintext under the public key $(f(\mathbf{x}),\ \mathsf{pkt}_f)$. Since sk_f is the decryption key for the public key $(0, \mathsf{pkt}_f)$, decryption will succeed whenever $f(\mathbf{x}) = 0$ as required.

The security of KH-PKE systems. Security for a key-homomorphic PKE is defined so as to make the ABE system above secure. More precisely, we define security as follows.

Definition A.1 (Selectively-secure KH-PKE)**.** A KH-PKE scheme $\Pi = (\mathsf{Setup}_{\mathsf{KH}}, \mathsf{KeyGen}_{\mathsf{KH}}, \mathsf{E}_{\mathsf{KH}}, \mathsf{Eval}_{\mathrm{pk}}, \mathsf{Eval}_{\mathrm{ct}})$ for a class of functions $\mathcal{F}_\lambda = \{f : \mathcal{X}_\lambda^{\ell(\lambda)} \to \mathcal{Y}_\lambda\}$ is *selectively secure* if for all probabilistic polynomial-time adversaries $\mathcal{A}$ where $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$, there is a negligible function $\nu(\lambda)$ such that

$$\mathbf{Adv}_{\Pi,\mathcal{A}}^{\mathsf{KHPKE}}(\lambda) \stackrel{\mathsf{def}}{=} \Big|\Pr\Big[\mathrm{EXP}_{\mathsf{KHPKE},\Pi,\mathcal{A}}^{(0)}(\lambda) = 1\Big] - \Pr\Big[\mathrm{EXP}_{\mathsf{KHPKE},\Pi,\mathcal{A}}^{(1)}(\lambda) = 1\Big]\Big| \le \nu(\lambda),$$

where for each $b \in \{0,1\}$ and $\lambda \in \mathbb{N}$ the experiment $\mathrm{EXP}_{\mathsf{KHPKE},\Pi,\mathcal{A}}^{(b)}(\lambda)$ is defined as follows:

1. $\big(\mathbf{x}^* \in \mathcal{X}_\lambda^{\ell(\lambda)},\ \mathsf{state}_1\big) \leftarrow \mathcal{A}_1(\lambda)$
2. $(\mathsf{mpk}_{\mathrm{KH}}, \mathsf{msk}_{\mathrm{KH}}) \leftarrow \mathsf{Setup}_{\mathsf{KH}}(\lambda) \quad , \quad \mathsf{Pkt} \leftarrow \mathsf{PKT}^{\ell(\lambda)}$
3. $(\mu_0, \mu_1,\ \mathsf{state}_2) \leftarrow \mathcal{A}_2^{\mathsf{KG}_{\mathsf{KH}}(\mathsf{msk}_{\mathrm{KH}}, x^*, \cdot)}(\mathsf{mpk}_{\mathrm{KH}},\ \mathsf{Pkt},\ \mathsf{state}_1)$
4. $c^* \leftarrow \mathsf{E}_{\mathsf{KH}}(\mathsf{mpk}_{\mathrm{KH}},\ (\mathbf{x}^*, \mathsf{Pkt}),\ \mu_b)$
5. $b' \leftarrow \mathcal{A}_3^{\mathsf{KG}_{\mathsf{KH}}(\mathsf{msk}_{\mathrm{KH}}, x^*, \cdot)}(c^*, \mathsf{state}_2)$ // $\mathcal{A}$ outputs a guess b' for b
6. Output $b' \in \{0,1\}$

where $\mathsf{KG}_{\mathsf{KH}}(\mathsf{msk}_{\mathrm{KH}}, x^*, f)$ is an oracle that on input $f \in \mathcal{F}$ with $f(\mathbf{x}^*) = 0$ returns $\perp$ and otherwise with $\mathsf{pkt} \leftarrow \mathsf{Eval}_{\mathrm{pk}}(f, \mathsf{Pkt})$ returns $\mathsf{KeyGen}_{\mathsf{KH}}\big(\mathsf{msk}_{\mathrm{KH}}, (0, \mathsf{pkt})\big)$.

With Definition A.1 the following theorem is now immediate.

Theorem A.2. *The ABE system above is selectively secure provided the underlying KH-PKE is selectively secure.*