Fully Key-Homomorphic Encryption, Arithmetic Circuit ABE, and Compact Garbled Circuits*

Dan Boneh† Craig Gentry‡ Sergey Gorbunov§ Shai Halevi¶
Valeria Nikolaenko‖ Gil Segev** Vinod Vaikuntanathan††
Dhinakaran Vinayagamurthy‡‡

May 20, 2014

Abstract

We construct the first (key-policy) attribute-based encryption (ABE) system with short secret keys: the size of keys in our system depends only on the depth of the policy circuit, not its size. Our constructions extend naturally to arithmetic circuits with arbitrary fan-in gates thereby further reducing the circuit depth. Building on this ABE system we obtain the first reusable circuit garbling scheme that produces garbled circuits whose size is the same as the original circuit *plus* an additive $\mathsf{poly}(\lambda, d)$ bits, where λ is the security parameter and d is the circuit depth. Save the additive $\mathsf{poly}(\lambda, d)$ factor, this is the best one could hope for. All previous constructions incurred a *multiplicative* $\mathsf{poly}(\lambda)$ blowup. As another application, we obtain (single key secure) functional encryption with short secret keys.

We construct our attribute-based system using a mechanism we call *fully key-homomorphic encryption* which is a public-key system that lets anyone translate a ciphertext encrypted under a public-key $\mathbf{x}$ into a ciphertext encrypted under the public-key $(f(\mathbf{x}), f)$ of the same plaintext, for any efficiently computable f. We show that this mechanism gives an ABE with short keys. Security is based on the subexponential hardness of the learning with errors problem.

We also present a second (key-policy) ABE, using multilinear maps, with short ciphertexts: an encryption to an attribute vector $\mathbf{x}$ is the size of $\mathbf{x}$ plus $\mathsf{poly}(\lambda, d)$ additional bits. This gives a reusable circuit garbling scheme where the size of the garbled input is short, namely the same as that of the original input, *plus* a $\mathsf{poly}(\lambda, d)$ factor.

*This is the full version of a paper that appeared in Eurocrypt 2014 [BGG+14]. This work is a merge of two closely related papers [GGH+13d, BNS13].

†Stanford University, Stanford, CA, USA. Email: dabo@cs.stanford.edu.

‡IBM Research, Yorktown, NY, USA. Email: cbgentry@us.ibm.com.

§MIT, Cambridge, MA, USA. Email: sergeyg@mit.edu. This work was partially done while visiting IBM T. J. Watson Research Center.

¶IBM Research, Yorktown, NY, USA. Email: shaih@alum.mit.edu.

‖Stanford University, Stanford, CA, USA. Email: valerini@cs.stanford.edu.

**Hebrew University, Jerusalem, Israel. Email: segev@cs.huji.ac.il. This work was partially done while the author was visiting Stanford University.

††MIT, Cambridge, MA, USA. Email: vinodv@csail.mit.edu.

‡‡University of Toronto, Toronto, Ontario, Canada. Email: dhinakaran5@cs.toronto.edu.

1 Introduction

(Key-policy) attribute-based encryption [SW05, GPSW06] 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 $\mathbf{x} \in \mathcal{X}$. The encryption of μ can be decrypted using sk_f only if $f(\mathbf{x}) = 0 \in \mathcal{Y}$. Intuitively, the security requirement is collusion resistance: a coalition of users learns nothing about the plaintext message μ if none of their individual keys are authorized to decrypt the ciphertext.

Attribute-based encryption (ABE) is a powerful generalization of identity-based encryption [Sha84, BF03, Coc01] and fuzzy IBE [SW05, ABV$^+$12] and is a special case of functional encryption [BSW11]. It is used as a building-block in applications that demand complex access control to encrypted data [PTMW06], in designing protocols for verifiably outsourcing computations [PRV12], and for single-use functional encryption [GKP$^+$13b]. Here we focus on key-policy ABE where the access policy is embedded in the secret key. The dual notion called ciphertext-policy ABE can be realized from this using universal circuits, as explained in [GPSW06, GGH$^+$13c].

The past few years have seen much progress in constructing secure and efficient ABE schemes from different assumptions and for different settings. The first constructions [GPSW06, LOS$^+$10, OT10, LW12, Wat12, Boy13, HW13] apply to predicates computable by Boolean formulas which are a subclass of log-space computations. More recently, important progress has been made on constructions for the set of all polynomial-size circuits: Gorbunov, Vaikuntanathan, and Wee [GVW13] gave a construction from the Learning With Errors (LWE) problem and Garg, Gentry, Halevi, Sahai, and Waters [GGH$^+$13c] gave a construction using multilinear maps. In both constructions the policy functions are represented as Boolean circuits composed of fan-in 2 gates and the secret key size is proportional to the *size* of the circuit.

Our results. We present two new key-policy ABE systems. Our first system, which is the centerpiece of this paper, is an ABE based on the learning with errors problem [Reg05] that supports functions f represented as arithmetic circuits with large fan-in gates. It has secret keys whose size is proportional to *depth* of the circuit for f, not its size. 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. 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.

Theorem 1.1 (Informal)**.** *Let λ be the security parameter. Assuming subexponential LWE, there is an ABE scheme for the class of functions with depth-d circuits where the size of the secret key for a circuit C is $\mathsf{poly}(\lambda, d)$.*

Our second ABE system, based on multilinear maps ([BS02],[GGH13a]), optimizes the ciphertext size rather than the secret key size. The construction here relies on a generalization of broadcast encryption [FN93, BGW05, BW13] and the attribute-based encryption scheme of [GGH$^+$13c]. Previously, ABE schemes with short ciphertexts were known only for the class of Boolean formulas [ALdP11].

Theorem 1.2 (Informal)**.** *Let λ be the security parameter. Assuming that d-level multilinear maps exist, there is an ABE scheme for the class of functions with depth-d circuits where the size of the encryption of an attribute vector $\mathbf{x}$ is $|\mathbf{x}| + \mathsf{poly}(\lambda, d)$.*

Our ABE schemes result in a number of applications and have many desirable features, which we describe next.

Applications to reusable garbled circuits. Over the years, garbled circuits and variants have found many uses: in two party [Yao86] and multi-party secure protocols [GMW87, BMR90], one-time programs [GKR08], key-dependent message security [BHHI10], verifiable computation [GGP10], homomorphic computations [GHV10] and many others. Classical circuit garbling schemes produced single-use garbled circuits which could only be used in conjunction with one garbled input. Goldwasser et al. [GKP$^+$13b] recently showed the first fully reusable circuit garbling schemes and used them to construct token-based program obfuscation schemes and k-time programs [GKP$^+$13b].

Most known constructions of both single-use and reusable garbled circuits proceed by garbling each gate to produce a garbled truth table, resulting in a *multiplicative* size blowup of $\mathsf{poly}(\lambda)$. A fundamental question regarding garbling schemes is: *How small can the garbled circuit be?*

There are three exceptions to the gate-by-gate garbling method that we are aware of. The first is the "free XOR" optimization for *single-use* garbling schemes introduced by Kolesnikov and Schneider [KS08] where one produces garbled tables only for the AND gates in the circuit C. This still results in a multiplicative $\mathsf{poly}(\lambda)$ overhead but proportional to the number of AND gates (as opposed to the total number of gates). Secondly, Lu and Ostrovsky [LO13] recently showed a *single-use* garbling scheme for RAM programs, where the size of the garbled program grows as $\mathsf{poly}(\lambda)$ times its running time. Finally, Goldwasser et al. [GKP$^+$13a] show how to (reusably) garble non-uniform Turing machines under a non-standard and non-falsifiable assumption and incurring a multiplicative $\mathsf{poly}(\lambda)$ overhead in the size of the non-uniformity of the machine. In short, all known garbling schemes (even in the single-use setting) suffer from a multiplicative overhead of $\mathsf{poly}(\lambda)$ in the circuit size or the running time.

Using our first ABE scheme (based on LWE) in conjunction with the techniques of Goldwasser et al. [GKP$^+$13b], we obtain the first reusable garbled circuits whose size is $|C| + \mathsf{poly}(\lambda, d)$. For large and shallow circuits, such as those that arise from database lookup, search and some machine learning applications, this gives significant bandwidth savings over previous methods (even in the single use setting).

Theorem 1.3 (Informal)**.** *Assuming subexponential LWE, there is a reusable circuit garbling scheme that garbles a depth-d circuit C into a circuit $\hat{C}$ such that $|\hat{C}| = |C| + \mathsf{poly}(\lambda, d)$, and garbles an input x into an encoded input $\hat{x}$ such that $|\hat{x}| = |x| \cdot \mathsf{poly}(\lambda, d)$.*

We next ask if we can obtain short garbled inputs of size $|\hat{\mathbf{x}}| = |\mathbf{x}| + \mathsf{poly}(\lambda, d)$, analogous to what we achieved for the garbled circuit. In a beautiful recent work, Applebaum, Ishai, Kushilevitz and Waters [AIKW13] showed constructions of *single-use* garbled circuits with short garbled inputs of size $|\hat{\mathbf{x}}| = |\mathbf{x}| + \mathsf{poly}(\lambda)$. We remark that while their garbled inputs are short, their garbled circuits still incur a multiplicative $\mathsf{poly}(\lambda)$ overhead.

Using our second ABE scheme (based on multilinear maps) in conjunction with the techniques of Goldwasser et al. [GKP$^+$13b], we obtain the first reusable garbling scheme with garbled inputs of size $|\mathbf{x}| + \mathsf{poly}(\lambda, d)$.

Theorem 1.4 (Informal)**.** *Assuming subexponential LWE and the existence of d-level multilinear maps, there is a reusable circuit garbling scheme that garbles a depth-d circuit C into a circuit $\hat{C}$ such that $|\hat{C}| = |C| \cdot \mathsf{poly}(\lambda, d)$, and garbles an input $\mathbf{x}$ into an encoded input $\hat{x}$ such that $|\hat{\mathbf{x}}| = |\mathbf{x}| + \mathsf{poly}(\lambda, d)$.*

A natural open question is to construct a scheme which produces both short garbled circuits and short garbled inputs. We first focus on describing the ABE schemes and then give details of the garbling scheme.

ABE for arithmetic circuits. For a prime q, our first ABE system (based on LWE) directly handles arithmetic circuits with weighted addition and multiplication gates over $\mathbb{Z}_q$, namely gates of the form

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

where the weights α_i can be arbitrary elements in $\mathbb{Z}_q$. Previous ABE constructions worked with Boolean circuits.

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 $\mathbb{Z}_q$). 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 $[-p, p]$. We discuss the relation between p and q at the end of the section.

We can in turn apply our arithmetic ABE construction to Boolean circuits with large fan-in resulting in potentially large savings over constructions restricted to fan-in two gates. An AND gate can be implemented as $\wedge(x_1, \ldots, x_k) = x_1 \cdots x_k$ and an OR gate as $\vee(x_1, \ldots, x_k) = 1 - (1 - x_1) \cdots (1 - x_k)$. In this setting, the inputs to the gates g_+ and $g_\times$ are naturally small, namely in $\{0, 1\}$. Thus, unbounded fan-in allows us to consider circuits with smaller size and depth, and results in smaller overall parameters.

ABE with key delegation. Our first ABE system also supports key delegation. That is, using the master secret key, 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 if and only if 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 with short keys

Key-homomorphic public-key encryption. We obtain our ABE by constructing a public-key encryption scheme that supports computations on public keys. Basic public keys in our system are vectors $\mathbf{x}$ in $\mathbb{Z}_q^\ell$ for some ℓ. Now, let $\mathbf{x}$ be a tuple in $\mathbb{Z}_q^\ell$ 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 $f(\mathbf{x})$.

More precisely, suppose $\mathbf{c}$ is an encryption of message μ under public-key $\mathbf{x} \in \mathbb{Z}_q^\ell$. There is a public algorithm $\mathsf{Eval}_{\mathrm{ct}}(f, \mathbf{x}, \mathbf{c}) \longrightarrow \mathbf{c}_f$ that outputs a ciphertext $\mathbf{c}_f$ that is an encryption of μ under the public-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 $\mathbf{c}$ whenever $f(\mathbf{x}) = 0$, as required for ABE. Unfortunately, this ABE is completely insecure! This is 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. A base encryption public-key is a tuple $\mathbf{x} \in \mathbb{Z}_q^\ell$ as before, however $\mathsf{Eval}_{\mathrm{ct}}$ produces ciphertexts encrypted under the public key $(f(\mathbf{x}),\ \langle f \rangle)$ where $f(\mathbf{x}) \in \mathbb{Z}_q$ and $\langle f \rangle$ is an encoding of the circuit computing f. Transforming a ciphertext $\mathbf{c}$ from the public key $\mathbf{x}$ to $(f(\mathbf{x}),\ \langle f \rangle)$ is done using algorithm $\mathsf{Eval}_{\mathrm{ct}}(f, \mathbf{x}, \mathbf{c}) \longrightarrow \mathbf{c}_f$ as before. To simplify the notation we write a public-key $(y, \langle f \rangle)$ as simply (y, f). The precise syntax and security requirements for key-homomorphic public-key encryption are provided in Section 3.

To build an ABE we simply publish the parameters of the key-homomorphic PKE system. A message μ is encrypted with attribute vector $\mathbf{x} = (x_1, \ldots, x_\ell) \in \mathbb{Z}_q^\ell$ that serves as the public key. Let $\mathbf{c}$ be the resulting ciphertext. Given an arithmetic circuit f, the key-homomorphic property lets anyone transform $\mathbf{c}$ into an encryption of μ under key $(f(\mathbf{x}),\ f)$. The point is that now the secret key for the function f can simply be the decryption key for the public-key $(0, f)$. This key enables the decryption of $\mathbf{c}$ when $f(\mathbf{x}) = 0$ as follows: the decryptor first uses $\mathsf{Eval}_{\mathrm{ct}}(f, \mathbf{x}, \mathbf{c}) \longrightarrow \mathbf{c}_f$ to transform the ciphertext to the public key $(f(\mathbf{x}),\ f)$. It can then decrypt c_f using the decryption key it was given whenever $f(\mathbf{x}) = 0$. We show that this results in a secure ABE.

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

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

We obtain a ciphertext $\mathbf{c}_\mathbf{x}$. We note that this encryption algorithm is the same as encryption in the hierarchical IBE system of [ABB10] and 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 $\mathbf{c}_\mathbf{x}$ 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 the matrix $\mathbf{B}_f \in \mathbb{Z}_q^{n \times m}$ serves as the encoding of the circuit for the function f. This $\mathbf{B}_f$ 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.

To illustrate the idea, assume that we have the ciphertext under the public key (x, y): $\mathbf{c}_\mathbf{x} = (\mathbf{c}_0 \mid \mathbf{c}_x \mid \mathbf{c}_y)$. Here $\mathbf{c}_0 = \mathbf{A}^T\mathbf{s} + \mathbf{e}$, $\mathbf{c}_x = (x\mathbf{G} + \mathbf{B}_1)^T\mathbf{s} + \mathbf{e}_1$ and $\mathbf{c}_y = (y\mathbf{G} + \mathbf{B}_2)^T\mathbf{s} + \mathbf{e}_2$. To compute the ciphertext under the public key $(x + y,\ \mathbf{B}_+)$ one takes the sum of the ciphertexts $\mathbf{c}_x$ and $\mathbf{c}_y$.

The result is the encryption under the matrix

$$(x+y)\mathbf{G} + (\mathbf{B}_1 + \mathbf{B}_2) \in \mathbb{Z}_q^{n\times m}$$

where $\mathbf{B}_+ = \mathbf{B}_1 + \mathbf{B}_2$. One of the main contributions of this work is a novel method of multiplying the public keys. Together with addition, described above, this gives full key-homomorphism. To construct the ciphertext under the public key $(xy,\ \mathbf{B}_\times)$, we first compute a small-norm matrix $\mathbf{R} \in \mathbb{Z}_q^{m\times m}$, s.t. $\mathbf{GR} = -\mathbf{B}_1$. With this in mind we compute

$$\begin{aligned}
\mathbf{R}^T\mathbf{c}_y &= \mathbf{R}^T \cdot \left[(y\mathbf{G} + \mathbf{B}_2)^T\mathbf{s} + \mathbf{e}_2\right] \approx (-y\mathbf{B}_1 + \mathbf{B}_2\mathbf{R})^T s, \quad \text{and} \\
y \cdot \mathbf{c}_x &= y\left[(x\mathbf{G} + \mathbf{B}_1)^T\mathbf{s} + \mathbf{e}_1\right] \approx (xy\mathbf{G} + y\mathbf{B}_1)^T\mathbf{s}
\end{aligned}$$

Adding the two expressions above gives us

$$(xy\mathbf{G} + \mathbf{B}_2\mathbf{R})^T\mathbf{s} + \mathbf{noise}$$

which is a ciphertext under the public key $(xy,\ \mathbf{B}_\times)$ where $\mathbf{B}_\times = \mathbf{B}_2\mathbf{R}$. Note that performing this operation requires that we know y. This is the reason why this method gives an ABE and not (private index) predicate encryption. In Section 4.1 we show how to generalize this mechanism to arithmetic circuits with arbitrary fan-in gates.

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). 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.

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 $\mathsf{Eval}_{\mathrm{ct}}(f, \mathbf{x}, \mathbf{c})$ to a ciphertext $\mathbf{c}$ 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 $\mathbf{c}$ 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 rely on the hardness of the learning with errors (LWE) problem, which requires that the ratio q/B is not too large. In particular, the underlying problem is believed to be hard even when q/B is $2^{(n^\epsilon)}$ for some fixed $0 < \epsilon < 1/2$. In our settings $q/B = \Omega\big((p^{k-1}m)^d\big)$. Then to support circuits of depth $t(\lambda)$ for some polynomial $t(\cdot)$ we choose n such that $n \geq t(\lambda)^{(1/\epsilon)} \cdot (2\log_2 n + k\log p)^{1/\epsilon}$, set $q = 2^{(n^\epsilon)}$, $m = \Theta(n \log q)$, and 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 ABE 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, namely the trapdoor for the matrix $(\mathbf{A}|\mathbf{B}_f)$. 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$^+$13c] secret keys grew as $O(d^2 s)$ where s is the number of boolean gates or wires in the circuit.

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$^+$13b] constructed functional encryption (which implies predicate encryption) for all polynomial-size functionalities using indistinguishability obfuscation.

The encryption algorithm in our system is similar to that 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] system 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 (private index) 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 [GGSW13] 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 $\lambda \in \mathbb{N}$.

The *statistical distance* between two random variables X and Y over a finite domain Ω is defined as

$$\mathrm{SD}(X, Y) = \frac{1}{2} \sum_{\omega \in \Omega} |\Pr[X = \omega] - \Pr[Y = \omega]|.$$

Two random variables X and Y are *δ-close* if $\mathrm{SD}(X, Y) \leq \delta$. Two distribution ensembles $\{X_\lambda\}_{\lambda \in \mathbb{N}}$ and $\{Y_\lambda\}_{\lambda \in \mathbb{N}}$ are *statistically indistinguishable* if it holds that $\mathrm{SD}(X_\lambda, Y_\lambda)$ is negligible in λ. Such random variables are *computationally indistinguishable* if for every probabilistic polynomial-time algorithm $\mathcal{A}$ it holds that

$$\left| \Pr_{x \leftarrow X_\lambda} \left[\mathcal{A}(1^\lambda, x) = 1 \right] - \Pr_{y \leftarrow Y_\lambda} \left[\mathcal{A}(1^\lambda, y) = 1 \right] \right|$$

is negligible in λ.

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 $\Pi = (\mathsf{Setup}, \mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec})$ 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; $\mathsf{Enc}(\mathsf{mpk},\ x \in \mathcal{X}_\lambda,\ \mu)$ outputs a ciphertext $\mathbf{c}$, the encryption of message μ labeled with attribute vector x; $\mathsf{Dec}(\mathsf{sk}_f, \mathbf{c})$ outputs a message μ or the special symbol $\perp$. (When clear from the context, we drop the subscript λ from $\mathcal{X}_\lambda$, $\mathcal{Y}_\lambda$ and $\mathcal{F}_\lambda$.)

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

Security. For the most part, we consider the standard notion of selective security for ABE schemes [GPSW06]. Specifically, we consider adversaries that first announce a challenge attribute vector 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 the ciphertexts of two different messages encrypted under the challenge attribute x^*. Formally, security is captured by the following definition.

Definition 2.1 (Selectively-secure ABE)**.** An ABE scheme $\Pi = (\mathsf{Setup}, \mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec})$ 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

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

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 messages μ_0, μ_1
4. $\mathbf{c}^* \leftarrow \mathsf{Enc}(\mathsf{mpk}, x^*, \mu_b)$
5. $b' \leftarrow \mathcal{A}_3^{\mathsf{KG}(\mathsf{msk}, x^*, \cdot)}(\mathbf{c}^*, \mathsf{state}_2)$ // $\mathcal{A}$ 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.

A fully secure ABE scheme is defined similarly, except that the adversary can choose the challenge attribute x^* after seeing the master public key and making polynomially many secret key queries. The following lemma, attributed to [BB11], says that any selectively secure ABE scheme is also fully secure with an exponential loss in parameters.

Lemma 2.2. *For any selectively secure ABE scheme with attribute vectors of length $\ell = \ell(\lambda)$, there is a negligible function $\nu(\lambda)$ such that $\mathbf{Adv}^{\mathsf{full}\text{-}\mathsf{ABE}}_{\Pi,\mathcal{A}}(\lambda) \leq 2^{\ell(\lambda)} \cdot \nu(\lambda)$.*

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 $\{\mathbf{x} \in \mathbb{Z}^m \ : \ \mathbf{Ax} = 0 \text{ in } \mathbb{Z}_q\}$. More generally, for $\mathbf{u} \in \mathbb{Z}_q^n$ we let $\Lambda_q^{\mathbf{u}}(\mathbf{A})$ denote the coset $\{\mathbf{x} \in \mathbb{Z}^m \ : \ \mathbf{Ax} = \mathbf{u} \text{ in } \mathbb{Z}_q\}$.

We note the following elementary fact: if the columns of $\mathbf{T_A} \in \mathbb{Z}^{m \times m}$ are a basis of the lattice $\Lambda_q^{\perp}(\mathbf{A})$, then they are also a basis for the lattice $\Lambda_q^{\perp}(x\mathbf{A})$ for any nonzero $x \in \mathbb{Z}_q$.

Learning with errors (LWE) [Reg05]. Fix integers n, m, a prime integer q and a noise distribution χ over $\mathbb{Z}$. 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 $n^{O(d)}$ 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 $\mathbf{R} \in \mathbb{Z}^{k \times m}$, let $\tilde{\mathbf{R}}$ be the result of applying Gram-Schmidt (GS) orthogonalization to the columns of $\mathbf{R}$. We define three matrix norms:

- $\|\mathbf{R}\|$ denotes the ℓ_2 length of the longest column of $\mathbf{R}$.
- $\|\mathbf{R}\|_{\mathsf{GS}} = \|\tilde{\mathbf{R}}\|$ where $\tilde{\mathbf{R}}$ is the GS orthogonalization of $\mathbf{R}$.
- $\|\mathbf{R}\|_2$ is the operator norm of $\mathbf{R}$ defined as $\|\mathbf{R}\|_2 = \sup_{\|\mathbf{x}\|=1} \|\mathbf{Rx}\|$.

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

We will use the following algorithm, throughout our paper:

$\mathsf{BD}(\mathbf{A}) \longrightarrow \mathbf{R}$ where $m = n\lceil \log q \rceil$: a deterministic algorithm that takes in a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and outputs a matrix $\mathbf{R} \in \mathbb{Z}_q^{m \times m}$, where each element $a \in \mathbb{Z}_q$ that belongs to the matrix $\mathbf{A}$ gets transformed into a column vector $\mathbf{r} \in \mathbb{Z}_q^{\lceil \log q \rceil}$, $\mathbf{r} = [a_0, ..., a_{\lceil \log q \rceil - 1}]^T$. Here a_i is the i-th bit of the binary decomposition of a ordered from LSB to MSB.

Claim 2.3. *For any matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, matrix $\mathbf{R} = \mathsf{BD}(\mathbf{A})$ has the norm $\|\mathbf{R}\|_2 \leq m$ and $\|\mathbf{R}^T\|_2 \leq m$.*

Trapdoor generators. The following lemma states properties of algorithms for generating short basis of lattices.

Lemma 2.4. *Let $n, m, q > 0$ be integers with q prime. There are polynomial time algorithms with the properties below:*

- $\mathsf{TrapGen}(1^n, 1^m, q) \longrightarrow (\mathbf{A}, \mathbf{T_A})$ ([Ajt99, AP09, MP12])*: 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 $\Lambda_q^\perp(\mathbf{A})$ such that $\mathbf{A}$ is $negl(n)$-close to uniform and $\|\mathbf{T}\|_{\mathtt{GS}} = O(\sqrt{n \log q})$, with all but negligible probability in n.*

- $\mathsf{ExtendRight}(\mathbf{A}, \mathbf{T_A}, \mathbf{B}) \longrightarrow \mathbf{T_{(A|B)}}$ ([CHKP10])*: 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 $\mathbf{T_{(A|B)}}$ of $\Lambda_q^\perp(\mathbf{A}|\mathbf{B})$ such that $\|\mathbf{T_A}\|_{\mathtt{GS}} = \|\mathbf{T_{(A|B)}}\|_{\mathtt{GS}}$.*

- $\mathsf{ExtendLeft}(\mathbf{A}, \mathbf{G}, \mathbf{T_G}, \mathbf{S}) \longrightarrow \mathbf{T_H}$ *where* $\mathbf{H} = (\mathbf{A} \mid \mathbf{G} + \mathbf{AS})$ ([ABB10])*: 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 $\|\mathbf{T_H}\|_{\mathtt{GS}} \leq \|\mathbf{T_G}\|_{\mathtt{GS}} \cdot (1 + \|\mathbf{S}\|_2)$.*

- *For $m = n\lceil \log q \rceil$ there is a fixed full-rank matrix $\mathbf{G} \in \mathbb{Z}_q^{n \times m}$ s.t. the lattice $\Lambda_q^\perp(\mathbf{G})$ has a publicly known basis $\mathbf{T_G} \in \mathbb{Z}^{m \times m}$ with $\|\mathbf{T_G}\|_{\mathtt{GS}} \leq \sqrt{5}$. The matrix $\mathbf{G}$ is such that for any matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, $\mathbf{G} \cdot \mathsf{BD}(\mathbf{A}) = \mathbf{A}$.*

To simplify the notation we will always assume that the matrix $\mathbf{G}$ from part 4 of Lemma 2.4 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 $\sigma = \tilde{\Omega}(\sqrt{n})$, 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 $1 - \mathrm{negl}(m)$.

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 $\mathcal{D}_\sigma(\Lambda_q^{\mathbf{u}_i}(\mathbf{A}))$ independently 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 2.5. *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*

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

with overwhelming probability in m.

Proof. For the $\{\pm 1\}$ matrix $\mathbf{S}$ the lemma follows from Litvak et al. [LPRTJ05] (Fact 2.4). For the matrix $\mathbf{R}$ the lemma follow from the fact that $\|\mathbf{R}^\mathsf{T}\|_2 \leq \sqrt{k} \cdot \|\mathbf{R}\| < \sqrt{k}(\sigma\sqrt{m})$. □

Solving $\mathbf{AX} = \mathbf{U}$. We review algorithms for finding a low-norm matrix $\mathbf{X} \in \mathbb{Z}^{m \times k}$ such that $\mathbf{AX} = \mathbf{U}$.

Lemma 2.6. *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 $\mathbf{AX} = \mathbf{U}$ with the properties below:*

- $\mathsf{SampleD}(\mathbf{A}, \mathbf{T_A}, \mathbf{U}, \sigma) \longrightarrow \mathbf{X}$ ([GPV08])*: a randomized algorithm that, when $\sigma = \|\mathbf{T}_A\|_{\mathtt{GS}} \cdot \omega(\sqrt{\log m})$, outputs a random sample $\mathbf{X}$ from a distribution that is statistically close to $\mathcal{D}_\sigma(\Lambda_q^{\mathbf{U}}(\mathbf{A}))$.*
- $\mathsf{RandBasis}(\mathbf{A}, \mathbf{T_A}, \sigma) \longrightarrow \mathbf{T'_A}$ ([CHKP10])*: a randomized algorithm that, when $\sigma = \|\mathbf{T}_A\|_{\mathtt{GS}} \cdot \omega(\sqrt{\log m})$, outputs a basis $\mathbf{T'_A}$ of $\Lambda_q^\perp(\mathbf{A})$ sampled from a distribution that is statistically close to $(\mathcal{D}_\sigma(\Lambda_q^\perp(\mathbf{A})))^m$. Note that $\|\mathbf{T'_A}\|_{\mathtt{GS}} < \sigma\sqrt{m}$ with all but negligible probability.*

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

Lemma 2.7. *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.

Additional algorithms Throughout the paper we will use the following algorithms:

Lemma 2.8.

- $\mathsf{SampleRight}(\mathbf{A}, \mathbf{T_A}, \mathbf{B}, \mathbf{U}, \sigma)$ *: a randomized algorithm that given full-rank matrices $\mathbf{A}, \mathbf{B} \in \mathbb{Z}_q^{n \times m}$, matrix $\mathbf{U} \in \mathbb{Z}_q^{n \times m}$, a basis $\mathbf{T_A}$ of $\Lambda_q^\perp(\mathbf{A})$ and $\sigma = \|\mathbf{T}_A\|_{\mathtt{GS}} \cdot \omega(\sqrt{\log m})$, outputs a random sample $\mathbf{X} \in \mathbb{Z}_q^{2m \times m}$ from a distribution that is statistically close to $\mathcal{D}_\sigma(\Lambda_q^{\mathbf{U}}((\mathbf{A}|\mathbf{B})))$. This algorithm is the composition of two algorithms:* $\mathsf{ExtendRight}(\mathbf{A}, \mathbf{T_A}, \mathbf{B}) \longrightarrow \mathbf{T}_{(\mathbf{A}|\mathbf{B})}$ *and* $\mathsf{SampleD}((\mathbf{A}|\mathbf{B}), \mathbf{T}_{(\mathbf{A}|\mathbf{B})}, \mathbf{U}, \sigma) \longrightarrow \mathbf{X}$.
- $\mathsf{SampleLeft}(\mathbf{A}, \mathbf{S}, y, \mathbf{U}, \sigma)$ *: a randomized algorithm that given full-rank matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, matrices $\mathbf{S}, \mathbf{U} \in \mathbb{Z}_q^{n \times m}$, $y \neq 0 \in \mathbb{Z}_q$ and $\sigma = \sqrt{5} \cdot (1 + \|\mathbf{S}\|_2) \cdot \omega(\sqrt{\log m})$, outputs a random sample $\mathbf{X} \in \mathbb{Z}_q^{2m \times m}$ from a distribution that is statistically close to $\mathcal{D}_\sigma(\Lambda_q^{\mathbf{U}}((\mathbf{A}|y\mathbf{G} + \mathbf{AS})))$, where $\mathbf{G}$ is the matrix from Lemma 2.4, part 4. This algorithm is the composition of two algorithms:* $\mathsf{ExtendLeft}(\mathbf{A}, y\mathbf{G}, \mathbf{T_G}, \mathbf{S}) \longrightarrow \mathbf{T}_{(\mathbf{A}|y\mathbf{G}+\mathbf{AS})}$ *and* $\mathsf{SampleD}((\mathbf{A}|y\mathbf{G}+\mathbf{AS}), \mathbf{T}_{(\mathbf{A}|y\mathbf{G}+\mathbf{AS})}, \mathbf{U}, \sigma) \longrightarrow \mathbf{X}$.

2.3 Multilinear Maps

Assume there exists a group generator $\mathcal{G}$ that takes the security parameter 1^λ and the pairing bound k and outputs groups $G_1, \ldots, G_k$ each of large prime order $q > 2^\lambda$. Let g_i be the generator of group G_i and let $g = g_1$. In addition, the algorithm outputs a description of a set of bilinear maps:

$$\{e_{ij} \ : \ G_i \times G_j \to G_{i+j} \mid i, j \geq 1, i + j \leq k\}$$

satisfying $e_{ij}(g_i^a, g_j^b) = g_{i+j}^{ab}$ for all $a, b \in \mathbb{Z}_q$. We sometimes omit writing e_{ij} and for convince simply use e as the map descriptor.

Definition 2.9. [(k,ℓ)-Multilinear Diffie-Hellman Exponent Assumption] The challenger runs $\mathcal{G}(1^\lambda, k)$ to generate groups $G_1, \ldots, G_k$, generators $g_1, \ldots, g_k$ and the map descriptions e_{ij}. Next, it picks $c_1, c_2, \ldots, c_k \in \mathbb{Z}_q$ at random. The (k,ℓ)-MDHE problem is hard if no adversary can distinguish between the following two experiments with better than negligible advantage in λ:

$$\left(g^{c_1}, \ldots, g^{c_1^\ell}, \ldots, g^{c_1^{\ell+2}}, \ldots, g^{c_1^{2\ell}}, g^{c_2}, \ldots, g^{c_k}, \beta = g_k^{c_1^{\ell+1} \prod_{2 \le i \le k} c_i}\right)$$

and

$$\left(g^{c_1}, \ldots, g^{c_1^\ell}, \ldots, g^{c_1^{\ell+2}}, \ldots, g^{c_1^{2\ell}}, g^{c_2}, \ldots, g^{c_k}, \beta\right)$$

where β is a randomly chosen element in G_k.

We note that if $k = 2$, then this corresponds exactly to the bilinear Diffie-Hellman Exponent Assumption (BDHE). Also, is easy to compute $g_k^{c_1^{\ell+1} \prod_{2 \le i \le k-1} c_i}$ by repeated pairing of the challenge components.

3 Fully Key-Homomorphic PKE (FKHE)

Our new ABE constructions are a direct application of fully key-homomorphic public-key encryption (FKHE), a notion that we introduce. Such systems are public-key encryption schemes that are homomorphic with respect to the public encryption key. We begin by precisely defining FKHE and then show that a key-policy ABE with short keys arises naturally from such a system.

Let $\{\mathcal{X}_\lambda\}_{\lambda \in \mathbb{N}}$ and $\{\mathcal{Y}_\lambda\}_{\lambda \in \mathbb{N}}$ be sequences of finite sets. Let $\{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$ be a sequence of sets of functions, namely $\mathcal{F}_\lambda = \{f : \mathcal{X}_\lambda^\ell \to \mathcal{Y}_\lambda\}$ for some $\ell > 0$. Public keys in an FKHE scheme are pairs $(x, f) \in \mathcal{Y}_\lambda \times \mathcal{F}_\lambda$. We call x the "value" and f the associated function. All such pairs are valid public keys. We also allow tuples $\mathbf{x} \in \mathcal{X}_\lambda^\ell$ to function as public keys. To simplify the notation we often drop the subscript λ and simply refer to sets $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{F}$.

In our constructions we set $\mathcal{X} = \mathbb{Z}_q$ for some q and let $\mathcal{F}$ be the set of ℓ-variate functions on $\mathbb{Z}_q$ computable by polynomial size arithmetic circuits.

Now, an FKHE scheme for the family of functions $\mathcal{F}$ consists of five PPT algorithms:

- $\mathsf{Setup}_{\mathsf{FKHE}}(1^\lambda) \to (\mathsf{mpk}_{\text{FKHE}}, \mathsf{msk}_{\text{FKHE}})$: outputs a master secret key $\mathsf{msk}_{\text{FKHE}}$ and public parameters $\mathsf{mpk}_{\text{FKHE}}$.
- $\mathsf{KeyGen}_{\mathsf{FKHE}}\big(\mathsf{msk}_{\text{FKHE}},\ (y, f)\big) \to \mathsf{sk}_{y,f}$: outputs a decryption key for the public key $(y, f) \in \mathcal{Y} \times \mathcal{F}$.
- $\mathsf{E}_{\mathsf{FKHE}}\big(\mathsf{mpk}_{\text{FKHE}},\ \mathbf{x} \in \mathcal{X}^\ell,\ \mu\big) \longrightarrow \mathbf{c}_{\mathbf{x}}$: encrypts message μ under the public key $\mathbf{x}$.
- Eval : a *deterministic* algorithm that implements key-homomorphism. Let $\mathbf{c}$ be an encryption of message μ under public key $\mathbf{x} \in \mathcal{X}^\ell$. For a function $f : \mathcal{X}^\ell \to \mathcal{Y} \in \mathcal{F}$ the algorithm does:

  $$\mathsf{Eval}\big(f,\ \mathbf{x},\ \mathbf{c}\big) \longrightarrow \mathbf{c}_f$$

  where if $y = f(x_1, \ldots, x_\ell)$ then $\mathbf{c}_f$ is an encryption of message μ under public-key (y, f).
- $\mathsf{D}_{\mathsf{FKHE}}(\mathsf{sk}_{y,f},\ \mathbf{c})$: decrypts a ciphertext $\mathbf{c}$ with key $\mathsf{sk}_{y,f}$. If $\mathbf{c}$ is an encryption of μ under public key (x, g) then decryption succeeds only when $x = y$ and f and g are identical arithmetic circuits.

Algorithm Eval captures the key-homomorphic property of the system: ciphertext $\mathbf{c}$ encrypted with key $\mathbf{x} = (x_1, \ldots, x_\ell)$ is transformed to a ciphertext $\mathbf{c}_f$ encrypted under key $\big(f(x_1, \ldots, x_\ell),\ f\big)$.

Correctness. The key-homomorphic property is stated formally in the following requirement: For all $(\mathsf{mpk}_{\text{FKHE}}, \mathsf{msk}_{\text{FKHE}})$ output by Setup, all messages μ, all $f \in \mathcal{F}$, and $\mathbf{x} = (x_1, \ldots, x_\ell) \in \mathcal{X}^\ell$:

If $\quad \mathbf{c} \leftarrow \mathsf{E}_{\mathsf{FKHE}}\big(\mathsf{mpk}_{\text{FKHE}},\ \mathbf{x} \in \mathcal{X}^\ell,\ \mu\big), \quad y = f(x_1, \ldots, x_\ell),$

$\mathbf{c}_f = \mathsf{Eval}\big(f,\ \mathbf{x},\ \mathbf{c}\big), \quad \mathsf{sk} \leftarrow \mathsf{KeyGen}_{\mathsf{FKHE}}(\mathsf{msk}_{\text{FKHE}}, (y, f))$

Then $\mathsf{D}_{\mathsf{FKHE}}(\mathsf{sk}, \mathbf{c}_f) = \mu$.

An ABE from a FKHE. A FKHE for a family of functions $\mathcal{F} = \{f : \mathcal{X}^\ell \to \mathcal{Y}\}$ immediately gives a key-policy ABE. Attribute vectors 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 $\mathsf{Setup}_{\mathsf{FKHE}}(1^\lambda)$ to get public parameters mpk and master secret msk. These function as the ABE public parameters and master secret.
- $\mathsf{Keygen}(\mathsf{msk}, f)$: Output $\mathsf{sk}_f \leftarrow \mathsf{KeyGen}_{\mathsf{FKHE}}\big(\mathsf{msk}_{\text{FKHE}},\ (0, f)\big)$.
  Jumping ahead, we remark that in our FKHE instantiation (in Section 4), the number of bits needed to encode the function f in sk_f depends only on the depth of the circuit computing f, not its size. Therefore, the size of sk_f depends only on the depth complexity of f.
- $\mathsf{Enc}(\mathsf{mpk},\ \mathbf{x} \in \mathcal{X}^\ell,\ \mu)$: output $(\mathbf{x}, \mathbf{c})$ where $\mathbf{c} \leftarrow \mathsf{E}_{\mathsf{FKHE}}(\mathsf{mpk}_{\text{FKHE}},\ \mathbf{x},\ \mu\big)$.
- $\mathsf{Dec}\big(\mathsf{sk}_f,\ (\mathbf{x}, \mathbf{c})\big)$: if $f(\mathbf{x}) = 0$ set $\mathbf{c}_f = \mathsf{Eval}\big(f,\ \mathbf{x},\ \mathbf{c}\big)$ and output the decrypted answer $\mathsf{D}_{\mathsf{FKHE}}(\mathsf{sk}_f, \mathbf{c}_f)$.
  Note that c_f is the encryption of the plaintext under the public key $(f(\mathbf{x}),\ f)$. Since sk_f is the decryption key for the public key $(0, f)$, decryption will succeed whenever $f(\mathbf{x}) = 0$ as required.

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

Definition 3.1 (Selectively-secure FKHE)**.** A fully key homomorphic encryption scheme $\Pi = (\mathsf{Setup}_{\mathsf{FKHE}}, \mathsf{KeyGen}_{\mathsf{FKHE}}, \mathsf{E}_{\mathsf{FKHE}}, \mathsf{Eval})$ for a class of functions $\mathcal{F}_\lambda = \{f : \mathcal{X}_\lambda^{\ell(\lambda)} \to \mathcal{Y}_\lambda\}$ is *selectively secure* if for all p.p.t. adversaries $\mathcal{A}$ where $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$, there is a negligible function $\nu(\lambda)$ such that

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

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

1. $\big(\mathbf{x}^* \in \mathcal{X}_\lambda^{\ell(\lambda)},\ \mathsf{state}_1\big) \leftarrow \mathcal{A}_1(\lambda)$
2. $(\mathsf{mpk}_{\text{FKHE}}, \mathsf{msk}_{\text{FKHE}}) \leftarrow \mathsf{Setup}_{\mathsf{FKHE}}(\lambda)$
3. $(\mu_0, \mu_1,\ \mathsf{state}_2) \leftarrow \mathcal{A}_2^{\mathsf{KG}_{\mathsf{KH}}(\mathsf{msk}_{\text{FKHE}}, x^*, \cdot, \cdot)}(\mathsf{mpk}_{\text{FKHE}},\ \mathsf{state}_1)$

4. $\mathbf{c}^* \leftarrow \mathsf{E}_{\mathsf{FKHE}}(\mathsf{mpk}_{\text{FKHE}},\ \mathbf{x}^*,\ \mu_b)$

5. $b' \leftarrow \mathcal{A}_3^{\mathsf{KG}_{\mathsf{KH}}(\mathsf{msk}_{\text{FKHE}}, x^*, \cdot, \cdot)}(\mathbf{c}^*, \mathsf{state}_2)$ // $\mathcal{A}$ outputs a guess b' for b

6. output $b' \in \{0,1\}$

where $\mathsf{KG}_{\mathsf{KH}}(\mathsf{msk}_{\text{FKHE}}, x^*, y, f)$ is an oracle that on input $f \in \mathcal{F}$ and $y \in \mathcal{Y}_\lambda$, returns $\perp$ whenever $f(\mathbf{x}^*) = y$, and otherwise returns $\mathsf{KeyGen}_{\mathsf{FKHE}}\big(\mathsf{msk}_{\text{FKHE}}, (y, f)\big)$.

With Definition 3.1 the following theorem is now immediate.

Theorem 3.2. *The ABE system above is selectively secure provided the underlying FKHE is selectively secure.*

4 An ABE and FKHE for arithmetic circuits from LWE

We now turn to building an FKHE for arithmetic circuits from the learning with errors (LWE) problem. This directly gives an ABE with short private keys as explained in Section 3. Our construction follows the key-homomorphism paradigm outlined in the introduction.

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 2.4 (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

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

For now we will assume the existence of three efficient *deterministic* algorithms $\mathsf{Eval}_{\text{pk}}, \mathsf{Eval}_{\text{ct}}, \mathsf{Eval}_{\text{sim}}$ 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 $\alpha_{\mathcal{F}} : \mathbb{Z} \to \mathbb{Z}$.

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

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

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

  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 $\{x_i\}_{i=1}^{\ell}$ into a ciphertext $\mathbf{c}_f$ encrypted under public-key $(f(\mathbf{x}), f)$.

- $\mathsf{Eval}_{\text{sim}}(\ f \in \mathcal{F},\ \ \big((x_i^*, \mathbf{S}_i)\big)_{i=1}^{\ell},\ \ \mathbf{A}) \longrightarrow \mathbf{S}_f \in \mathbb{Z}_q^{m \times m}$. 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

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

  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 4.1. The deterministic algorithms $(\mathsf{Eval}_{\rm pk}, \mathsf{Eval}_{\rm ct}, \mathsf{Eval}_{\rm sim})$ are $\alpha_{\mathcal{F}}$-*FKHE 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 $\alpha_{\mathcal{F}}$-FKHE 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 FKHE system. Given *FKHE-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 FKHE 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 $\alpha_{\mathcal{F}}$-*FKHE 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 Appendix 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 $\alpha_{\mathcal{F}} > \sqrt{n \log m}$.

- $\mathsf{Setup}_{\mathsf{FKHE}}(1^\lambda) \to (\mathsf{mpk}_{\rm FKHE}, \mathsf{msk}_{\rm FKHE})$: Run algorithm $\mathsf{TrapGen}(1^n, 1^m, q)$ from Lemma 2.4 (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 $\mathbf{D}, \mathbf{B}_1, \ldots, \mathbf{B}_\ell \in \mathbb{Z}_q^{n \times m}$ and output a master secret key $\mathsf{msk}_{\rm FKHE}$ and public parameters $\mathsf{mpk}_{\rm FKHE}$:

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

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

  To construct $\mathbf{R}_f$ run algorithm $\mathsf{SampleRight}(\mathbf{A}, \mathbf{T_A}, y\mathbf{G} + \mathbf{B}_f, \mathbf{D}, \sigma)$ from Lemma 2.8, part 1. Here σ is sufficiently large for algorithm $\mathsf{SampleRight}$ since $\sigma = \|\mathbf{T_A}\|_{\rm GS} \cdot \omega(\sqrt{\log m})$, where $\|\mathbf{T_A}\|_{\rm GS} = O(\sqrt{n \log q})$.

  Note that the secret key $\mathsf{sk}_{y,f}$ is always in $\mathbb{Z}^{2m \times m}$ independent of the complexity of the function f. We assume $\mathsf{sk}_{y,f}$ also implicitly includes $\mathsf{mpk}_{\rm FKHE}$.

- $\mathsf{E}_{\mathsf{FKHE}}\big(\mathsf{mpk}_{\rm FKHE},\ \mathbf{x} \in \mathcal{X}^\ell,\ \mu\big) \longrightarrow \mathbf{c_x}$: 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

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

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

- $\mathsf{D}_{\mathsf{FKHE}}(\mathsf{sk}_{y,f},\ \mathbf{c})$: Let $\mathbf{c}$ be the encryption of μ under public key (x, g). If $x \neq y$ or f and g are not identical arithmetic circuits, output $\perp$. Otherwise, let $\mathbf{c} = (\mathbf{c}_{in}, \mathbf{c}_1, \ldots, \mathbf{c}_\ell, \mathbf{c}_{out}) \in \mathbb{Z}_q^{(\ell+2)m}$.

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

  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 $f(\mathbf{x}) = y$ we know by the requirement on $\mathsf{Eval}_{\mathrm{ct}}$ that $\mathbf{c}_f$ is in $E_{\mathbf{s},\Delta}(y, \mathbf{B}_f)$ so that $\mathbf{c}_f = y\mathbf{G} + \mathbf{B}_f^{\mathsf{T}}\mathbf{s} + \mathbf{e}$ with $\|\mathbf{e}\| < \Delta$. Consequently,

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

Since $\mathbf{R}_f \in \mathbb{Z}^{2m\times m}$ is sampled from the distribution $\mathcal{D}_\sigma(\Lambda_q^{\mathbf{D}}(\mathbf{A}|y\mathbf{G} + \mathbf{B}_f))$ we know that $(\mathbf{A}|y\mathbf{G} + \mathbf{B}_f) \cdot \mathbf{R}_f = \mathbf{D}$ and, by Lemma 2.5, $\|\mathbf{R}_f^{\mathsf{T}}\|_2 < 2m\sigma$ 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 FKHE 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 *FKHE-enabling*.

Theorem 4.2. *Given the three algorithms* $(\mathsf{Eval}_{pk}, \mathsf{Eval}_{ct}, \mathsf{Eval}_{sim})$ *for the family of functions* $\mathcal{F}$*, the FKHE 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 FKHE.*

Proof idea. Before giving the complete proof we first briefly sketch the main proof idea which hinges on the properties of algorithms $(\mathsf{Eval}_{\mathrm{pk}}, \mathsf{Eval}_{\mathrm{ct}}, \mathsf{Eval}_{\mathrm{sim}})$ and also employs ideas from [CHKP10, ABB10]. We build an LWE algorithm $\mathcal{B}$ that uses a selective FKHE attacker $\mathcal{A}$ to solve LWE. $\mathcal{B}$ 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 FKHE 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 $\mathsf{mpk}_{\mathrm{FKHE}} = (\mathbf{A}, \mathbf{D}, \mathbf{B}_1, \ldots, \mathbf{B}_\ell)$. 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 $f \in \mathcal{F}$ and attribute $y \in \mathbb{Z}_q$. Only functions f and attributes y for which $y^* = f(x_1^*, \ldots, x_\ell^*) \neq y$ 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 $(\mathbf{A}|\mathbf{B}_f)\cdot\mathbf{R}_f =$

$\mathbf{D}$. To do so $\mathcal{B}$ needs a recoding matrix from the lattice $\Lambda_q^\perp(\mathbf{F})$ where $\mathbf{F} = (\mathbf{A}|\mathbf{B}_f)$ to the lattice $\Lambda_q^\perp(\mathbf{D})$. In the real key generation algorithm this short basis is derived from a short basis for $\Lambda_q^\perp(\mathbf{A})$ using algorithm $\mathsf{SampleRight}$. 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 $\mathbf{B}_f = \mathbf{A}\mathbf{S}_f - y^*\mathbf{G}$. Because $y^* \neq y$, algorithm $\mathcal{B}$ can construct the required key as $\mathbf{R}_f \leftarrow \mathsf{SampleLeft}(\mathbf{A}, \mathbf{S}_f, (y - y^*), \mathbf{D}, \sigma)$.

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 do so $\mathcal{B}$ uses $\mathsf{Eval}_{\mathrm{sim}}$ giving it the secret matrices $\mathbf{S}_i^*$. More precisely, $\mathcal{B}$ runs $\mathsf{Eval}_{\mathrm{sim}}(f,\ \big((x_i^*, \mathbf{S}_i^*)\big)_{i=1}^{\ell},\ \mathbf{A})$ 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 decrypts $\mathbf{c}^*$. The reason is that it cannot build a secret key $\mathsf{sk}_{y,f}$ for functions where $f(\mathbf{x}^*) = y$ and these are the only keys that will decrypt $\mathbf{c}^*$.

Proof of Theorem 4.2. 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 $\mathbf{B}_1, \ldots, \mathbf{B}_\ell$ 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 $\mathbf{S}_1^*, \ldots, \mathbf{S}_\ell^* \in \{-1, 1\}^{m\times m}$ denote the random matrices generated for the creation of $\mathbf{c}^*$ in the encryption algorithm Enc.

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

$$\mathbf{B}_i := \mathbf{A}\,\mathbf{S}_i^* - x_i^*\mathbf{G} \tag{2}$$

The remainder of the game is unchanged.

We show that Game 0 is statistically indistinguishable from Game 1 by Lemma 2.7. 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 2.7 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 (2) 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 $\mathbf{B}_1, \ldots, \mathbf{B}_\ell$ 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 $f \in \mathcal{F}$ and element $y \in \mathcal{Y}$. Only

f such that $y^* = f(x_1^*, \ldots, x_\ell^*) \neq y$ are allowed. To respond, the key generation oracle computes $\mathbf{B}_f = \mathsf{Eval}_{\mathrm{pk}}\big(f,\ (\mathbf{B}_1, \ldots, \mathbf{B}_\ell)\big)$ and needs to produce a matrix $\mathbf{R}_f$ in $\mathbb{Z}^{2m \times m}$ satisfying

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

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 $\|\mathbf{S}_f\|_2 \leq \alpha_{\mathcal{F}}$.
- Finally, it responds with $\mathbf{R}_f = \mathsf{SampleLeft}(\mathbf{A}, \mathbf{S}_f, y, \mathbf{D}, \sigma)$. By definition of $\mathsf{SampleLeft}$ we know that $\mathbf{R}_f$ is distributed as required. Indeed because $\|\mathbf{S}_f\|_2 \leq \alpha_{\mathcal{F}}(n)$, $\sigma = \sqrt{5} \cdot (1 + \|\mathbf{S}_f\|_2) \cdot \omega(\sqrt{\log m})$ as needed for algorithm $\mathsf{SampleLeft}$ in Lemma 2.8, 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

$$\mathbf{c}_{in} = \mathbf{A}^\mathsf{T}\mathbf{s} + \mathbf{e}_0 \quad \text{and} \quad \mathbf{c}_{out} = \mathbf{D}^\mathsf{T}\mathbf{s} + \mathbf{e}_1 \tag{3}$$

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

$$\mathbf{c}_0^* \quad = \quad (\mathbf{I}_m|\mathbf{S}_1^*|\ldots|\mathbf{S}_\ell^*)^\mathsf{T} \cdot \mathbf{c}_{in} \quad \in \mathbb{Z}_q^{(\ell+1)m} \tag{4}$$

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) 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 Enc 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 (4) 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 Enc. 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 (4) 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. ■

Remark 4.3. We note that the matrix $\mathbf{R}_f$ in $\mathsf{KeyGen}_{\mathsf{FKHE}}$ can alternatively be generated using a sampling method from [MP12]. To do so we choose FKHE public parameters as we do in the security proof by choosing random matrices $\mathbf{S}_i, \ldots, \mathbf{S}_\ell$ in $\{\pm 1\}^{m \times m}$ and setting $\mathbf{B}_i = \mathbf{A}\,\mathbf{S}_i$. We then define the matrix $\mathbf{B}_f$ as $\mathbf{B}_f := \mathbf{A}\mathbf{S}_f$ where $\mathbf{S}_f = \mathsf{Eval}_{\mathrm{sim}}(f,\ ((0, \mathbf{S}_i))_{i=1}^{\ell},\ \mathbf{A})$. We could then build the secret key matrix $\mathsf{sk}_{y,f} = \mathbf{R}_f$ satisfying $(\mathbf{A} | y\mathbf{G} + \mathbf{B}_f) \cdot \mathbf{R}_f = \mathbf{D}$ directly from the bit decomposition of $\mathbf{D}/y$. Adding suitable low-norm noise to the result will ensure that $\mathsf{sk}_{y,f}$ is distributed as in the simulation in the security proof. Note that this approach can only be used to build secret keys $\mathsf{sk}_{y,f}$ when $y \neq 0$ where as the method in $\mathsf{KeyGen}_{\mathsf{FKHE}}$ works for all y.

4.1 Evaluation Algorithms for Arithmetic Circuits

In this section we build the *FKHE-enabling* algorithms $(\mathsf{Eval}_{\mathrm{pk}}, \mathsf{Eval}_{\mathrm{ct}}, \mathsf{Eval}_{\mathrm{sim}})$ that are at the heart of the FKHE construction in Section 4. We do so for the family of polynomial depth, unbounded fan-in arithmetic circuits.

4.2 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:

$$\mathcal{G} = \bigcup_{\alpha,\alpha_1,\ldots,\alpha_k \in \mathbb{Z}_q} \left\{ g \mid g : \mathbb{Z}_q^k \to \mathbb{Z}_q, \quad \begin{array}{c} g(x_1,\ldots,x_k) = \alpha_1 x_1 + \alpha_2 x_2 + \ldots + \alpha_k x_k \\ \text{or} \\ g(x_1,\ldots,x_k) = \alpha \cdot x_1 \cdot x_2 \cdot \ldots \cdot x_k \end{array} \right\} \tag{5}$$

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:

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

- 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

  $$\mathbf{G}\mathbf{R}_i = \alpha_i \mathbf{G}\ :\ \mathbf{R}_i = \mathsf{BD}(\alpha_i \mathbf{G}) \qquad \text{(as in Lemma 2.4 part 4).} \tag{6}$$

  Output the following matrices and the ciphertext:

  $$\mathbf{B}_g = \sum_{i=1}^k \mathbf{B}_i \mathbf{R}_i, \qquad \mathbf{S}_g = \sum_{i=1}^k \mathbf{S}_i \mathbf{R}_i, \qquad \mathbf{c}_g = \sum_{i=1}^k \mathbf{R}_i^T \mathbf{c}_i \tag{7}$$

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

  $$\mathbf{G}\mathbf{R}_1 = \alpha\mathbf{G}\ :\ \mathbf{R}_1 = \mathsf{BD}(\alpha\mathbf{G}) \tag{8}$$
  $$\mathbf{G}\mathbf{R}_i = -\mathbf{B}_{i-1}\mathbf{R}_{i-1}\ :\ \mathbf{R}_i = \mathsf{BD}(-\mathbf{B}_{i-1}\mathbf{R}_{i-1}) \quad \text{for all } i \in \{2,3,\ldots,k\} \tag{9}$$

  Output the following matrices and the ciphertext:

  $$\mathbf{B}_g = \mathbf{B}_k\mathbf{R}_k, \qquad \mathbf{S}_g = \sum_{j=1}^k \left(\prod_{i=j+1}^k x_i^*\right) \mathbf{S}_j\mathbf{R}_j, \qquad \mathbf{c}_g = \sum_{j=1}^k \left(\prod_{i=j+1}^k x_i\right) \mathbf{R}_j^T\mathbf{c}_j \tag{10}$$

  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 (10) 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 *FKHE-enabling*.

Lemma 4.4. *Let $\beta_g(m) = km$. 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* $\Delta \leq \beta_g(m) \cdot \delta$ *and* $\mathbf{B}_g = \mathsf{Eval}_{pk}(g,\ (\mathbf{B}_1, \ldots, \mathbf{B}_k))$.

2. *The output* $\mathbf{S}_g$ *satisfies* $\mathbf{A}\mathbf{S}_g - g(\mathbf{x}^*)\mathbf{G} = \mathbf{B}_g$ *where* $\|\mathbf{S}_g\|_2 \leq \beta_g(m) \cdot \max_{i\in[k]} \|\mathbf{S}_i\|_2$ *and* $\mathbf{B}_g = \mathsf{Eval}_{pk}\big(g,\ (\mathbf{A}\mathbf{S}_1 - x_1^*\mathbf{G}, \ldots, \mathbf{A}\mathbf{S}_k - x_k^*\mathbf{G})\big)$.

Proof. By Eq. 7 the output ciphertext is computed as follows:

$$\begin{aligned}
\mathbf{c}_g &= \sum_{i=1}^{k} \mathbf{R}_i^T \cdot \mathbf{c}_i = \sum_{i=1}^{k} \mathbf{R}_i^T \cdot \Big((x_i\mathbf{G} + \mathbf{B}_i)^T \mathbf{s} + \mathbf{e}_i\Big) = \quad \text{// substitute for } \mathbf{c}_i = (x_i\mathbf{G} + \mathbf{B}_i)^T\mathbf{s} + \mathbf{e}_i \\
&= \sum_{i=1}^{k} (x_i\mathbf{G}\mathbf{R}_i)^T\mathbf{s} + \sum_{i=1}^{k} (\mathbf{B}_i\mathbf{R}_i)^T\mathbf{s} + \sum_{i=1}^{k} (\mathbf{R}_i^T\mathbf{e}_i) = \quad \text{// break the product into components} \\
&= \left(\sum_{i=1}^{k} \alpha_i x_i\right) \mathbf{G}^T\mathbf{s} + \mathbf{B}_g^T\mathbf{s} + \mathbf{e}_g = \quad \text{// } \mathbf{G}\mathbf{R}_i = \alpha_i\mathbf{R}_i \text{ from Eq. 6 and } \mathbf{B}_g = \sum_{i=1}^{k} \mathbf{B}_i\mathbf{R}_i \text{ from Eq. 7} \\
&= [g(\mathbf{x})\mathbf{G} + \mathbf{B}_g]^T \mathbf{s} + \mathbf{e}_g
\end{aligned}$$

The noise bound is: $\|\mathbf{e}_g\| = \|\mathbf{R}_1^T\mathbf{e}_1 + \cdots + \mathbf{R}_k^T\mathbf{e}_k\| \leq k \cdot \max_{j\in[k]} \left(\|\mathbf{R}_j^T\|_2 \cdot \|\mathbf{e}_j\|\right) \overset{\text{Lemma 2.4,part 4}}{\leq} km \cdot \delta$. This completes the proof of the first part of the lemma.

In the second part of the lemma, by Eq. 7 the output matrix $\mathbf{B}_g$ is computed as follows:

$$\begin{aligned}
\mathbf{B}_g &= \sum_{i=1}^{k} (\mathbf{A}\mathbf{S}_i - x_i^*\mathbf{G})\mathbf{R}_i = \quad \text{// plug-in matrices given in the lemma into Eq. 7} \\
&= \mathbf{A}\sum_{i=1}^{k} \mathbf{S}_i\mathbf{R}_i - \sum_{i=1}^{k} \alpha_i x_i^*\mathbf{G} = \mathbf{A}\mathbf{S}_g - g(x^*)\mathbf{G} \quad \text{// } \mathbf{G}\mathbf{R}_i = \alpha_i\mathbf{R}_i \text{ from Eq. 6}
\end{aligned}$$

Then $\|\mathbf{S}_g\|_2 = \|\sum_{i=1}^{k} \mathbf{S}_i\mathbf{R}_i\|_2 \leq k \cdot \max_{i\in[k]} \left(\|\mathbf{S}_i\|_2 \cdot \|\mathbf{R}_i\|_2\right) \overset{\text{Lemma 2.4,part 4}}{\leq} km \cdot \max_{i\in[k]} \left(\|\mathbf{S}_i\|_2\right)$ as required. □

The next Lemma proves similar bounds for a multiplication gate.

Lemma 4.5. *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 4.4 with* $\beta_g(m) = \frac{p^k - 1}{p - 1} m$.

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:

$$\begin{aligned}
\mathbf{c}_g &= \sum_{j=1}^{k} \left(\prod_{i=j+1}^{k} x_i\right) \mathbf{R}_j^T \mathbf{c}_j = \sum_{j=1}^{k} \left(\prod_{i=j+1}^{k} x_i\right) \mathbf{R}_j^T \left((x_j \mathbf{G} + \mathbf{B}_j)^T \mathbf{s} + \mathbf{e}_j\right) = \quad \text{// substitute for } \mathbf{c}_j \\
&= \left[\left(\prod_{i=1}^{k} x_i\right) \mathbf{G}\mathbf{R}_1 + \sum_{j=2}^{k} \left(\prod_{i=j}^{k} x_i\right) \cancel{(\mathbf{G}\mathbf{R}_j + \mathbf{B}_{j-1}\mathbf{R}_{j-1})} + \mathbf{B}_k \mathbf{R}_k\right]^T \mathbf{s} + \mathbf{e}_g = \quad \text{// regroup} \\
&= \left[\left(\prod_{i=1}^{k} x_i\right) \mathbf{G}\mathbf{R}_1 + \mathbf{B}_k \mathbf{R}_k\right]^T \mathbf{s} + \mathbf{e}_g = \quad \text{// use Eq. 9 to cancel terms} \\
&= [g(\mathbf{x})\mathbf{G} + \mathbf{B}_g]^T \mathbf{s} + \mathbf{e}_g \quad \text{// use the facts } \mathbf{G}\mathbf{R}_1 = \alpha\mathbf{G} \text{ (Eq. 8), } \mathbf{B}_g = \mathbf{B}_k\mathbf{R}_k \text{ (Eq. 10)}
\end{aligned}$$

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

$$\|\mathbf{e}_g\| = \left\| \sum_{j=1}^{k} \left(\prod_{i=j+1}^{k} x_i\right) \mathbf{R}_j^T \mathbf{e}_j \right\| \leq \left(1 + p + \ldots + p^{k-1}\right) \cdot \max_{j \in [k]} \left[\|\mathbf{R}_j^T\|_2 \cdot \|\mathbf{e}_j\|\right] \overset{\text{Lemma. 2.3}}{\leq} \frac{p^k - 1}{p - 1} m \cdot \delta$$

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:

$$\begin{aligned}
\mathbf{B}_g &= (\mathbf{A}\mathbf{S}_k - x_k \mathbf{G})\mathbf{R}_k \overset{\text{Eq. 9}}{=} \quad \text{// by (9) we have } \mathbf{G}\mathbf{R}_k = -(\mathbf{A}\mathbf{S}_{k-1} - x_{k-1}\mathbf{G})\mathbf{R}_{k-1} \\
&= (\mathbf{A}\mathbf{S}_k\mathbf{R}_k + x_k \mathbf{A}\mathbf{S}_{k-1}\mathbf{R}_{k-1} - x_k \cdot x_{k-1}\mathbf{G}\mathbf{R}_{k-1}) \overset{\text{Eq. 9}}{=} \ldots \overset{\text{Eq. 9}}{=} \\
&= (\mathbf{A}\mathbf{S}_k\mathbf{R}_k + x_k \mathbf{A}\mathbf{S}_{k-1}\mathbf{R}_{k-1} + x_k \cdot x_{k-1}\mathbf{A}\mathbf{S}_{k-2}\mathbf{R}_{k-2} + \ldots + (-x_1 \cdots x_k \mathbf{G}\mathbf{R}_1)) \overset{\text{Eq. 8}}{=} \\
&= (\mathbf{A}\mathbf{S}_g - \alpha x_1 \cdots x_k \mathbf{G}) = (\mathbf{A}\mathbf{S}_g - g(\mathbf{x})\mathbf{G})
\end{aligned}$$

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

$$\begin{aligned}
\|\mathbf{S}_g\|_2 &= \left\| \sum_{j=1}^{k} \left(\prod_{i=j+1}^{k} x_i\right) \mathbf{S}_j \mathbf{R}_j \right\|_2 \\
&\leq \left(1 + p + \ldots + p^{k-1}\right) \max_{i \in [k]} \left(\|\mathbf{S}_i\|_2 \cdot \|\mathbf{R}_i\|_2\right) \overset{\text{Lemma. 2.3}}{\leq} \frac{p^k - 1}{p - 1} m \cdot \max_{i \in [k]} \left(\|\mathbf{S}_i\|_2\right)
\end{aligned}$$

as required. □

4.3 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 $\ell + 1, \ell + 2, \ldots, |\mathcal{C}| - 1$ and the output wire has index $|\mathcal{C}|$, which also denotes the size of the circuit. Every gate $g_w : \mathbb{Z}_q^{k_w} \to \mathbb{Z}_q$ (in $\mathcal{G}$ as per 5) is indexed as a tuple $(w_1, \ldots, w_{k_w}, w)$

where k_w is the fan-in of the gate. 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.

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

Let f be computed by some circuit $\mathcal{C} \in \mathcal{C}_\lambda$, that has ℓ input wires. We construct the required matrices inductively input to output gate-by-gate.

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

Using Eval algorithms described in Section 4.2, compute

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

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 4.6. *Let* $\beta(m) = \frac{p^k-1}{p-1}m$. *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* $\mathbf{B}_f = \mathsf{Eval}_{pk}(f,\ (\mathbf{B}_1, \ldots, \mathbf{B}_\ell))$.

Proof. By Lemma 4.4 and 4.5, 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 4.7. *Let* $\beta(m)$ *be as defined in Lemma 4.6. If* $\mathbf{S}_1, \ldots, \mathbf{S}_\ell$ *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* $\mathbf{B}_f = \mathsf{Eval}_{pk}\big(f,\ (\mathbf{A}\mathbf{S}_1 - x_1^*\mathbf{G}, \ldots, \mathbf{A}\mathbf{S}_\ell - x_\ell^*\mathbf{G})\big)$.

Proof. Since the input $\mathbf{S}_i$ for $i \in [\ell]$ are random matrices in $\{\pm 1\}^{m\times m}$, by Lemma 2.5 for all $i \in [\ell]$, $\|\mathbf{S}_i\|_2 < 20\sqrt{m}$. By Lemma 4.4 and 4.5, 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 $\alpha_{\mathcal{F}}$-FKHE enabling for

$$\alpha_{\mathcal{F}}(n) = (\beta(m))^d \cdot 20\sqrt{m} = O\big((p^{k-1}m)^d\sqrt{m}\big), \quad \text{where} \quad m = \Theta(n \log q). \tag{11}$$

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

4.4 ABE with Short Secret Keys for Arithmetic Circuits from LWE

The FKHE for a family of functions $\mathcal{F} = \{f : (\mathbb{Z}_q)^\ell \to \mathbb{Z}_q\}$ we constructed in Section 4 immediately gives a key-policy ABE as discussed in Section 3. For completeness we briefly describe the resulting ABE system.

Given *FKHE-enabling* algorithms $(\mathsf{Eval}_{\mathrm{pk}}, \mathsf{Eval}_{\mathrm{ct}}, \mathsf{Eval}_{\mathrm{sim}})$ for a family of functions $\mathcal{F}$ from Section 4.1, the ABE system works as follows:

- $\mathsf{Setup}(1^\lambda, \ell)$: Choose n, q, χ, m and σ as in "Parameters" in Section 4.
  Run algorithm $\mathsf{TrapGen}(1^n, 1^m, q)$ (Lemma 2.4, part 1) to generate $(\mathbf{A}, \mathbf{T}_\mathbf{A})$.
  Choose random matrices $\mathbf{D}, \mathbf{B}_1, \ldots, \mathbf{B}_\ell \in \mathbb{Z}_q^{n \times m}$ and output the keys:

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

- $\mathsf{Keygen}(\mathsf{msk}, f)$: Let $\mathbf{B}_f = \mathsf{Eval}_{\mathrm{pk}}(f,\ (\mathbf{B}_1, \ldots, \mathbf{B}_\ell))$.
  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 $\mathbf{R}_f$ run algorithm $\mathsf{SampleRight}(\mathbf{A}, \mathbf{T}_\mathbf{A}, y\mathbf{G} + \mathbf{B}_f, \mathbf{D}, \sigma)$ from Lemma 2.8, part 1.

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

- $\mathsf{Enc}(\mathsf{mpk},\ \mathbf{x} \in \mathbb{Z}_q^\ell,\ \mu \in \{0,1\}^m)$: Choose a random 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

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

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

- $\mathsf{Dec}\big(\mathsf{sk}_f,\ (\mathbf{x}, \mathbf{c})\big)$: If $f(\mathbf{x}) \neq 0$ output $\perp$. Otherwise, let the ciphertext $\mathbf{c} = (\mathbf{c}_{in}, \mathbf{c}_1, \ldots, \mathbf{c}_\ell, \mathbf{c}_{out}) \in \mathbb{Z}_q^{(\ell+2)m}$, set $\mathbf{c}_f = \mathsf{Eval}_{\mathrm{ct}}\big(f,\ \{(x_i, \mathbf{B}_i, \mathbf{c}_i)\}_{i=1}^\ell\big) \in \mathbb{Z}_q^m$.

  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. The proof of the following security theorem follows from Theorems 4.2 and 3.2.

Theorem 4.8. *For* FKHE-enabling *algorithms* $(\mathsf{Eval}_{pk}, \mathsf{Eval}_{ct}, \mathsf{Eval}_{sim})$ *for the family of functions* $\mathcal{F}$*, the ABE system above is correct and selectively-secure with respect to* $\mathcal{F}$*, assuming the* (n, q, χ)*-LWE assumption holds where* n, q, χ *are the parameters for the* FKHE-enabling *algorithms.*

5 Extensions

5.1 Key Delegation

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 4.4, 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) $\mathbf{T_F}$ for the matrix $\mathbf{F} = (\mathbf{A}|\mathbf{B}_f)$. The decryptor could use $\mathbf{T_F}$ to generate the matrix $\mathbf{R}_f$ herself using algorithm $\mathsf{SampleD}$. 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 $(\mathsf{Setup}, \mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec})$ 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 $\mu = \mathsf{Dec}(\mathsf{sk}_{f_1,\ldots,f_k}, (\mathbf{x}, c))$ with an overwhelming probability over the choice of $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \ell)$, $c \leftarrow \mathsf{Enc}(\mathsf{mpk}, \ \mathbf{x} \in \mathcal{X}^\ell, \ \mu)$, $\mathsf{sk}_{f_1} \leftarrow \mathsf{Keygen}(\mathsf{msk}, f_1)$ 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 5.1 (Selectively-secure DABE)**.** A DABE scheme $\Pi = (\mathsf{Setup}, \mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Delegate})$ 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

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

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. $c^* \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathbf{x}^*, \mu_b)$.
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 $i \in \{2, \ldots, k\}$ $\mathsf{sk}_{f_1,\ldots,f_i} = \mathsf{Delegate}(\mathsf{mpk}, \mathsf{sk}_{f_1,\ldots,f_{i-1}}, f_i)$.

5.1.1 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 $\mathbf{B}_f = \mathsf{Eval}_{\mathrm{pk}}(f,\ (\mathbf{B}_1, \ldots, \mathbf{B}_\ell))$.
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 $\sigma = \|\mathbf{T}_\mathbf{A}\|_{\mathrm{GS}} \cdot \omega(\sqrt{\log m})$ (we will set σ as before: $\sigma = \omega(\alpha_\mathcal{F} \cdot \sqrt{\log m})$).
Output $\mathsf{sk}_f := \mathbf{T}_f$.

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

Parse the secret key $\mathsf{sk}_{f_1,\ldots,f_k}$ as a matrix $\mathbf{T}_k \in \mathbb{Z}_q^{(k+1)m \times (k+1)m}$ which is a trapdoor for the matrix $(\mathbf{A}|\mathbf{B}_{f_1}|\ldots|\mathbf{B}_{f_k}) \in \mathbb{Z}_q^{n \times (k+1)m}$.
Let $\mathbf{B}_g = \mathsf{Eval}_{\mathrm{pk}}(g,\ (\mathbf{B}_1, \ldots, \mathbf{B}_\ell))$.
Build the basis for matrix $\mathbf{F} = (\mathbf{A}|\mathbf{B}_{f_1}|\ldots|\mathbf{B}_{f_k}|\mathbf{B}_g) \in \mathbb{Z}_q^{n \times (k+2)m}$:

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

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

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

Otherwise parse the secret key $\mathsf{sk}_{f_1,\ldots,f_k}$ as a matrix $\mathbf{T}_k \in \mathbb{Z}_q^{(k+1)m \times (k+1)m}$ which is a trapdoor for the matrix $(\mathbf{A}|\mathbf{B}_{f_1}|\ldots|\mathbf{B}_{f_k})$.
Run $\mathbf{R} \leftarrow \mathsf{SampleD}(\ (\mathbf{A}|\mathbf{B}_{f_1}|\ldots|\mathbf{B}_{f_k}),\ \mathbf{T}_k,\ \mathbf{D},\ \sigma_k)$ to generate a low-norm matrix $\mathbf{R} \in \mathbb{Z}_q^{(k+1)m \times m}$ such that $(\mathbf{A}|\mathbf{B}_{f_1}|\ldots|\mathbf{B}_{f_k}) \cdot \mathbf{R} = \mathbf{D}$.
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 $\mathbf{c}' = (\mathbf{c}_{in}|\mathbf{c}_1|\ldots|\mathbf{c}_k) \in \mathbb{Z}_q^{(k+1)m}$. Output $\mu = \mathsf{Round}(\mathbf{c}_{out} - \mathbf{R}^T\mathbf{c}')$.

Correctness. To show correctness, note that when $f_1(\mathbf{x}) = 0 \wedge \ldots \wedge f_k(\mathbf{x}) = 0$ 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,

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

We know that $(\mathbf{A}|\mathbf{B}_{f_1}|\ldots|\mathbf{B}_{f_k}) \cdot \mathbf{R} = \mathbf{D}$ and $||\mathbf{R}^T||_2 < (k+1)m\sigma_k$ with overwhelming probability by Lemma 2.5. 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 similar to the security game for FKHE, described in Section 4, except in Game 2 we need to answer delegated key queries. Consider a private key query $\mathsf{sk}_{f_1,\ldots,f_k}$, where $f_1, \ldots, f_k \in \mathcal{F}$. This query is only allowed when $f_1(\mathbf{x}^*) \neq 0 \vee \ldots \vee f_k(\mathbf{x}^*) \neq 0$. Without loss of generality, assume that $f_1(\mathbf{x}^*) = 0 \wedge \ldots \wedge f_{k-1}(\mathbf{x}^*) = 0$ 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 $\mathbf{B}_{f_i} = \mathsf{Eval}_{\mathrm{pk}}\big(f_i,\ (\mathbf{B}_1, \ldots, \mathbf{B}_\ell)\big)$ and needs to produce a trapdoor $\mathbf{T}_k \in \mathbb{Z}^{(k+1)m \times (k+1)m}$ for the matrix $(\mathbf{A}|\mathbf{B}_{f_1}|\ldots|\mathbf{B}_{f_k}) \in \mathbb{Z}_q^{n \times (k+1)m}$.

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 $\|\mathbf{S}_{f_k}\|_2 \leq \alpha_{\mathcal{F}}$.
- Let $\mathbf{F} = (\mathbf{A}|\mathbf{B}_{f_1}|\ldots|\mathbf{B}_{f_k}) = (\mathbf{A}|\mathbf{B}_{f_1}|\ldots|\mathbf{B}_{f_{k-1}}|\mathbf{A}\mathbf{S}_{f_k} - y^*\mathbf{G})$. 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 $\mathbf{T}_{(\mathbf{A}|\mathbf{B}_{f_k}|\mathbf{B}_{f_1}|\ldots|\mathbf{B}_{f_{k-1}})}$ by running

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

  Now we can switch the rows of the matrix $\mathbf{T}_{(\mathbf{A}|\mathbf{B}_{f_k}|\mathbf{B}_{f_1}|\ldots|\mathbf{B}_{f_{k-1}})}$ to get the matrix $\mathbf{T}_F$, which is a trapdoor for $(\mathbf{A}|\mathbf{B}_{f_1}|\ldots|\mathbf{B}_{f_k})$. This operation, as well as $\mathsf{ExtendRight}$ function (according to Lemma 2.4, part 2) does not change the Gram-Schmidt norm of the basis, therefore this trapdoor satisfies

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

  where the bound on $\|\mathbf{T}_{\mathbf{G}}\|_{\mathtt{GS}}$ is from Lemma 2.4 (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 $\sigma_k = \|\mathbf{T}_{\mathbf{F}}\|_{\mathtt{GS}} \cdot \omega(\sqrt{\log m})$ as needed for algorithm $\mathsf{RandBasis}$ in Lemma 2.6 (part 3).

5.2 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 5.2. 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:

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

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 4.3) 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 $\mathbf{R} = \mathsf{BD}(-\mathbf{B}_u)$. 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 5.3. *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 $\beta_d = (1 + p + \ldots + p^d) \cdot m$ 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 $\mathbf{A}\mathbf{S}_f - f(\mathbf{x})\mathbf{G} = \mathbf{B}_f$ where $\mathbf{B}_f = \mathsf{Eval}_{pk}\big(f,\ (\mathbf{A}\mathbf{S}_1 - x_1\mathbf{G}, \ldots, \mathbf{A}\mathbf{S}_\ell - x_\ell\mathbf{G})\big)$ 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 $||\mathbf{S}_{w_i}||_2 \leq \beta_{i-1}(m) \cdot \gamma$.
  - $\mathbf{c}_w = \mathbf{c}_{w_1} + \mathbf{c}_{w_2} = (x_{w_1}\mathbf{G} + \mathbf{B}_{w_1})^T s + \mathbf{e}_{w_1} + (x_{w_2}\mathbf{G} + \mathbf{B}_{w_2})^T s + \mathbf{e}_{w_2} = (x_w\mathbf{G} + \mathbf{B}_w)^T s + \mathbf{e}_w$
  - $||\mathbf{e}_w|| = ||\mathbf{e}_{w_1} + \mathbf{e}_{w_2}|| \leq ||\mathbf{e}_{w_1}|| + ||\mathbf{e}_{w_2}|| \leq (\beta_{i-1}(m) + \beta_{i-1}(m))\delta \leq \beta_i(m)\delta$
  - $\mathbf{B}_w = \mathbf{B}_{w_1} + \mathbf{B}_{w_2} = (\mathbf{A}\mathbf{S}_{w_1} - x_{w_1}\mathbf{G}) + (\mathbf{A}\mathbf{S}_{w_2} - x_{w_2}\mathbf{G}) = \mathbf{A}(\mathbf{S}_{w_1} + \mathbf{S}_{w_2}) - (x_{w_1} + x_{w_2})\mathbf{G} = \mathbf{A}\mathbf{S}_w - x_w\mathbf{G}$
  - $||\mathbf{S}_w||_2 = ||\mathbf{S}_{w_1} + \mathbf{S}_{w_2}||_2 \leq ||\mathbf{S}_{w_1}||_2 + ||\mathbf{S}_{w_2}||_2 \leq (\beta_{i-1}(m) + \beta_{i-1}(m)) \cdot \gamma \leq \beta_i(m) \cdot \gamma$.
- 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
  - $\mathbf{c}_w = x_i\mathbf{c}_u + \mathbf{R}^T\mathbf{c}_i = x_i(x_u\mathbf{G} + \mathbf{B}_u)^T s + x_i\mathbf{e}_u + \mathbf{R}^T(x_i\mathbf{G} + \mathbf{B}_i)^T s + \mathbf{R}^T\mathbf{e}_i = (x_w\mathbf{G} + x_i(\cancel{\mathbf{B}_g + \mathbf{G}\mathbf{R}}) + \mathbf{B}_w)^T s + \mathbf{e}_w$
  - $||\mathbf{e}_w||_2 = ||x_i\mathbf{e}_u + \mathbf{R}^T\mathbf{e}_i||_2 \leq p||\mathbf{e}_u||_2 + m||\mathbf{e}_i||_2 \leq (p\beta_{i-1}(m) + m)\delta \leq \beta_i(m) \cdot \delta$
  - $\mathbf{B}_w = \mathbf{B}_i\mathbf{R} = (\mathbf{A}\mathbf{S}_i - x_i\mathbf{G})\mathbf{R} = \mathbf{A}\mathbf{S}_i\mathbf{R} + x_i\mathbf{B}_u = \mathbf{A}\mathbf{S}_i\mathbf{R} + x_i(\mathbf{A}\mathbf{S}_u - x_u\mathbf{G}) = \mathbf{A}(x_i\mathbf{S}_u + \mathbf{S}_i\mathbf{R}) - (x_i x_u)\mathbf{G} = \mathbf{A}\mathbf{S}_w - x_w\mathbf{G}$
  - $||\mathbf{S}_w||_2 = ||x_i\mathbf{S}_u + \mathbf{S}_i\mathbf{R}||_2 \leq (p\beta_{i-1}(m) + m) \cdot \gamma \leq \beta_i(m) \cdot \gamma$.

as required. □

Now combining Lemma 5.3 and lemmas analogous to Lemmas 4.6, 4.7 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 $O(g \cdot |C|)$.

5.2.1 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 5.3, 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 4.3, 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:

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

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 5.3, 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 4.3 will be proportional to $(pm)^{O(\ell)}$ which is much worse.

6 ABE with Short Ciphertexts from Multi-linear Maps

We assume familiarity with multi-linear maps [BS02, GGH13a], which we overview in Section 2.3.

Intuition. We assume that the circuits consist of AND and OR gates. To handle general circuits (with negations), we can apply De Morgan's rule to transform it into a monotone circuit, doubling the number of input attributes (similar to [GGH$^+$13c]).

The inspiration of our construction comes from the beautiful work of Applebaum, Ishai, Kushilevitz and Waters [AIKW13] who show a way to compress the garbled input in a (single use) garbling scheme all the way down to size $|\mathbf{x}| + \mathsf{poly}(\lambda)$. This is useful to us in the context of ABE schemes due to a simple connection between ABE and *reusable* garbled circuits with authenticity observed in [GVW13]. In essence, they observe that the secret key for a function f in an ABE scheme corresponds to the garbled circuit for f, and the ciphertext encrypting an attribute vector $\mathbf{x}$ corresponds

to the garbled input for $\mathbf{x}$ in the reusable garbling scheme. Thus, the problem of compressing ciphertexts down to size $|\mathbf{x}| + \mathsf{poly}(\lambda)$ boils down to the question of generalizing [AIKW13] to the setting of *reusable* garbling schemes. We are able to achieve this using multilinear maps.

Security of the scheme relies on a generalization of the bilinear Diffie-Hellman Exponent Assumption to the multi-linear setting (see Definition 2.9). [1] The bilinear Diffie-Hellman Exponent Assumption was recently used to prove the security of the first broadcast encryption with constant size ciphertexts [BGW05] (which in turn can be thought of as a special case of ABE with short ciphertexts.)

Theorem 6.1 (Selective security)**.** *For all polynomials $d_{\max} = d_{\max}(\lambda)$, there exists a selectively-secure attribute-based encryption with ciphertext size* $\mathrm{poly}(d_{\max})$ *for any family of polynomial-size circuits with depth at most $d_{\max}$ and input size ℓ, assuming hardness of $(d+1,\ell)-$Multilinear Diffie-Hellman Exponent Assumption.*

6.1 Our Construction

- $\mathsf{Params}(1^\lambda, d_{\max})$: The parameters generation algorithm takes the security parameter and the maximum circuit depth. It generates a multi-linear map $\mathcal{G}(1^\lambda, k = d+1)$ that produces groups $(G_1, \ldots, G_k)$ along with a set of generators $g_1, \ldots, g_k$ and map descriptors $\{e_{ij}\}$. It outputs the public parameters $pp = \big(\{G_i, g_i\}_{i\in[k]}, \{e_{ij}\}_{i,j\in[k]}\big)$, which are implicitly known to all of the algorithms below.

- $\mathsf{Setup}(1^\ell)$: For each input bit $i \in \{1, 2, \ldots, \ell\}$, choose a random element q_i in $\mathbb{Z}_p$. Let $g = g_1$ be the generator of the first group. Define $h_i = g^{q_i}$. Also, choose α at random from $\mathbb{Z}_p$ and let $t = g_k^\alpha$. Set the master public key

$$\mathsf{mpk} := (h_1, \ldots, h_\ell, t)$$

  and the master secret key as $\mathsf{msk} := \alpha$.

- $\mathsf{Keygen}(\mathsf{msk}, C)$: The key-generation algorithm takes a circuit C with ℓ input bits and a master secret key msk and outputs a secret key sk_C defined as follows.

  1. Choose randomly $\big((r_1, z_1), \ldots, (r_\ell, z_\ell)\big)$ from $\mathbb{Z}_q^2$ for each input wire of the circuit C. In addition, choose $\big((r_{\ell+1}, a_{\ell+1}, b_{\ell+1}), \ldots, (r_n, a_n, b_n)\big)$ from $\mathbb{Z}_q^3$ randomly for all internal wires of C.
  2. Compute an $\ell \times \ell$ matrix $\tilde{M}$, where all diagonal entries (i,i) are of the form $(h_i)^{z_i} g^{r_i}$ and all non-diagonal entries (i,j) are of the form $(h_i)^{z_j}$. Append g^{-z_i} as the last row of the matrix and call the resulting matrix M.
  3. Consider a gate $\Gamma = (u, v, w)$ where wires u, v are at depth $j-1$ and w is at depth j. If Γ is an OR gate, compute

$$K_\Gamma = \big(K_\Gamma^1 = g^{a_w}, K_\Gamma^2 = g^{b_w}, K_\Gamma^3 = g_j^{r_w - a_w r_u}, K_\Gamma^4 = g_j^{r_w - b_w r_v}\big)$$

     Else if Γ is an AND gate, compute

$$K_\Gamma = \big(K_\Gamma^1 = g^{a_w}, K_\Gamma^2 = g^{b_w}, K_\Gamma^3 = g_j^{r_w - a_w r_u - b_w r_v}\big)$$

[1] Our construction can be converted to multi-linear graded-encodings, recently instantiated by Garg et al. [GGH13a] and Coron et al. [CLT13].

4. Set $\sigma = g_{k-1}^{\alpha - r_n}$
5. Define and output the secret key as

$$\mathsf{sk}_C := \big(C, \{K_\Gamma\}_{\Gamma \in C}, M, \sigma\big)$$

- $\mathsf{Enc}(\mathsf{mpk}, \mathbf{x}, \mu)$: The encryption algorithm takes the master public key mpk, an index $\mathbf{x} \in \{0,1\}^\ell$ and a message $\mu \in \{0,1\}$, and outputs a ciphertext $\mathbf{c_x}$ defined as follows. Choose a random element s in $\mathbb{Z}_q$. Let X be the set of indices i such that $x_i = 1$. Let $\gamma_0 = t^s$ if $\mu = 1$, otherwise let γ_0 be a randomly chosen element from G_k. Output ciphertext as

$$\mathbf{c_x} := \left(\mathbf{x}, \gamma_0,\ g^s,\ \gamma_1 = \big(\prod_{i \in X} h_i\big)^s\right)$$

- $\mathsf{Dec}(\mathsf{sk}_C, \mathbf{c_x})$: The decryption algorithm takes the ciphertext $\mathbf{c_x}$, and secret key sk_C and proceeds as follows. If $C(\mathbf{x}) = 0$, it outputs $\perp$. Otherwise,

1. Let X be the set of indices i such that $x_i = 1$. For each input wire $i \in X$, using the matrix M compute $g^{r_i}\big(\prod_{j \in X} h_j\big)^{z_i}$ and then

$$\begin{aligned} g_2^{r_i s} &= e\left(g^s, g^{r_i}\big(\prod_{j \in X} h_j\big)^{z_i}\right) \cdot e\left(\gamma_1, g^{-z_i}\right) \\ &= e\left(g^s, g^{r_i}\big(\prod_{j \in X} h_j\big)^{z_i}\right) \cdot e\left(\big(\prod_{j \in X} h_j\big)^s, g^{-z_i}\right) \end{aligned}$$

2. Now, for each gate $\Gamma = (u, v, w)$ where w is a wire at level j, (recursively going from the input to the output) compute $g_{j+1}^{r_w s}$ as follows:
   - If Γ is an OR gate, and $C(\mathbf{x})_u = 1$, compute $g_{j+1}^{r_w s} = e\big(K_\Gamma^1, g_j^{r_u s}\big) \cdot e\big(g^s, K_\Gamma^3\big)$.
   - Else if $C(\mathbf{x})_v = 1$, compute $g_{j+1}^{r_w s} = e\big(K_\Gamma^2, g_j^{r_v s}\big) \cdot e\big(g^s, K_\Gamma^4\big)$.
   - Else if Γ is an AND gate, compute $g_{j+1}^{r_w s} = e\big(K_\Gamma^1, g_j^{r_u s}\big) \cdot e\big(K_\Gamma^2, g_j^{r_v s}\big) \cdot e\big(g^s, K_\Gamma^3\big)$.

3. If $C(\mathbf{x}) = 1$, then the user computes $g_k^{r_n s}$ for the output wire. Finally, compute

$$\psi = e\big(g^s, \sigma\big) \cdot g_k^{r_n s} = e\big(g^s, g_{k-1}^{\alpha - r_n}\big) \cdot g_k^{r_n s}$$

4. Output $\mu = 1$ if $\psi = \gamma_0$, otherwise output 0.

6.2 Correctness

Claim 6.2. *For all active wires w at level j (that is, $C(\mathbf{x})_w = 1$) the user holds $g_{i+1}^{s r_w}$.*

Proof. Clearly, the base case is satisfied as shown above. Now consider a gate $\Gamma = (u, v, w)$. If g is an OR gate and assume $C(x)_u = 1$, then

$$\begin{aligned} g_{j+1}^{s r_w} &= e\big(K_\Gamma^1, g_j^{r_u s}\big) \cdot e\big(g^s, K_\Gamma^3\big) \\ &= e\big(g^{a_w}, g_j^{r_u s}\big) \cdot e\big(g^s, g_j^{r_w - a_w r_u}\big) \\ &= e\big(g, g_j\big)^{a_w r_u s} \cdot e\big(g, g_j\big)^{s r_w} \cdot e\big(g, g_j\big)^{-a_w r_u s} \end{aligned}$$

The case when $C(x)_v = 1$ is similar. Also, if g is an AND gate, then

$$\begin{aligned} g_{j+1}^{sr_w} &= e\big(K_\Gamma^1, g_j^{r_u s}\big) \cdot e\big(K_\Gamma^2, g_j^{r_v s}\big) \cdot e\big(g^s, K_\Gamma^3\big) \\ &= e\big(g^{a_w}, g_j^{r_u s}\big) \cdot e\big(g^{b_w}, g_j^{r_v s}\big) \cdot e\big(g^s, g_j^{r_w - a_w r_u - b_w r_v}\big) \\ &= e\big(g, g_j\big)^{a_w r_u s} \cdot e\big(g, g_j\big)^{b_w r_v s} \cdot e\big(g, g_j\big)^{sr_w} \cdot e\big(g, g_j\big)^{-a_w r_u s - b_w r_v s} \\ &= e\big(g, g_j\big)^{a_w r_u s + b_w r_v s} \cdot e\big(g, g_j\big)^{sr_w} \cdot e\big(g, g_j\big)^{-a_w r_u s - b_w r_v s} \end{aligned}$$

Hence, if $C(x) = 1$, the user computes $g_k^{sr_n}$ and so

$$\begin{aligned} \psi &= e\big(g^s, \sigma\big) \cdot g_k^{r_n s} \\ &= e\big(g^s, g_{k-1}^{\alpha - r_n}\big) \cdot g_k^{r_n s} \\ &= g_k^{\alpha s} = t^s = \gamma_0 \end{aligned}$$

if $m = 1$. □

6.3 Security Proof

Assume there is an adversary Adv* that breaks the security of the ABE scheme. We construct an adversary Adv that breaks the (k, ℓ)-Multi-linear Diffie-Hellman Exponent Assumption. The adversary Adv is given a challenge

$$\big(g^{c_1}, \ldots, g^{c_1^\ell}, \ldots, g^{c_1^{\ell+2}}, \ldots, g^{c_1^{2\ell}}, g^{c_2}, \ldots, g^{c_k}, \beta\big)$$

where β is either $g_k^{c_1^{\ell+1} \prod_{2 \le i \le k} c_i}$ or a random element of G_k. The adversary invokes Adv* and gets x^* as the challenge index. Let X be the set of indices i such that $x_i = 1$. The adversary will ensure the following induction: for every inactive wire w at depth j, $r_w = c_1^{\ell+1} \prod_{2 \le i \le j} c_i$ (plus known randomness). Hence, for all input wires w, $r_w = c_1^{\ell+1}$ (plus known randomness).

We now define simulated experiments which Adv will be using to break the assumption.

- $\mathsf{Setup}^*(1^\ell)$: For each input bit $i \notin X$, choose a random element b_i in $\mathbb{Z}_q$ and implicitly set $q_i = c_1^{\ell+1-i} + b_i$. For each $i \in X$, choose a random $q_i \in \mathbb{Z}_q$. Let $g = g_1$ be the generator of the first group. For all i, compute $h_i = g^{q_i}$. Randomly choose γ and let $t = g_k^\alpha = g_k^{c_1^{\ell+1} \prod_{2 \le i \le k-1} c_i + \gamma}$ which can be computed from the challenge component by repeated pairing. Set the master public key

  $$\mathsf{mpk} := (h_1, \ldots, h_\ell, t)$$

  and the master secret key as $\mathsf{msk} := \perp$.

- $\mathsf{Keygen}^*(C, \mathsf{msk})$: The key-generation algorithm takes a circuit C with ℓ input bits and a master secret key msk and outputs a secret key sk_C defined as follows.

  1. For all $i \in X$, choose randomly $r_i \in \mathbb{Z}_q$. For all $i \notin X$, randomly choose $f_i \in \mathbb{Z}_q$ and implicitly set $r_i = c_1^{\ell+1} + f_i$ (that is, we embed the challenge into the attributes $\notin X$).
  2. For all $i \in [\ell]$, choose $p_i \in \mathbb{Z}_q$ at random and implicitly set $z_i = -c_1^i + p_i$.

3. Compute the matrix M:

$$M := \begin{bmatrix} g^{-z_1} & g^{-z_2} & g^{-z_3} & \dots & g^{-z_\ell} \\ (h_1)^{z_1} g^{r_1} & (h_1)^{z_2} & (h_1)^{z_3} & \dots & (h_1)^{z_\ell} \\ (h_2)^{z_1} & (h_2)^{z_2} g^{r_2} & (h_2)^{z_3} & \dots & (h_2)^{z_\ell} \\ (h_3)^{z_1} & (h_3)^{z_2} & (h_3)^{z_3} g^{r_3} & \dots & (h_3)^{z_\ell} \\ \vdots & & \ddots & & \vdots \\ (h_\ell)^{z_1} & (h_\ell)^{z_2} & (h_\ell)^{z_3} & \dots & (h_\ell)^{z_\ell} g^{r_\ell} \end{bmatrix}$$

4. We now argue that the adversary can compute every entry in the matrix M.
   (a) Entries of the first row can be computed by $g^{-z_i} = g^{c_1^i - p_i} = g^{c_1^i} \cdot g^{-p_i}$, where p_i is known.
   (b) Note that for all $i = j$ (i.e. the diagonal entries). If $i \notin X$, then

$$(h_i)^{z_i} \cdot g^{r_i} = g^{(c_1^{\ell+1-i} + b_i)(-c_1^i + p_i)} \cdot g^{c_1^{\ell+1} + f_i} = g^{c_1^{\ell+1-i} p_i - b_i c_1^i + b_i p_i + f_i}$$

   If $i \in X$, then q_i, z_i, r_i are all known.
   (c) Now, consider non-diagonal entries $i \neq j$. If $i \notin X$ and $j \in X$, then

$$(h_i)^{z_j} = \left(g^{c_1^{\ell+1-i} + b_i}\right)^{-c_1^j + p_j} = g^{-c_1^{\ell+1-i+j}} \cdot g^{-b_i c_1^j} \cdot g^{p_j c_1^{\ell+1-i}} \cdot g^{b_i p_j}$$

   which can be computed given the challenge and the knowledge of b_i, p_j. Also, if $i \in X$ and $j \notin X$, similarly

$$(h_i)^{z_j} = \left(g^{q_i}\right)^{-c_1^j + p_j} = g^{-c_1^j q_i} \cdot g^{q_i p_j}$$

   can be computed given the challenge and the knowledge of q_i, p_j.
5. Consider a gate $\Gamma = (u, v, w)$ where wires u, v are at depth $j - 1$ and w is at depth j.

   (a) If Γ is an OR gate and $C(x^*)_w = 1$, then values r_w, a_w, b_w are randomly chosen from $\mathbb{Z}_q$. Otherwise, we implicitly set $a_w = c_j + d_w, b_w = c_j + k_w$, where $d_w, k_w \in \mathbb{Z}_q$ are randomly chosen and c_j is the value a part of the challenge. Also, implicitly set $r_w = c_1^{\ell+1} \prod_{2 \leq i \leq j} c_i + e_w$, where $e_w \mathbb{Z}_q$ is randomly chosen. Compute

$$K_\Gamma = \left(K_\Gamma^1 = g^{a_w}, K_\Gamma^2 = g^{b_w}, K_\Gamma^3 = g_j^{r_w - a_w r_u}, K_\Gamma^4 = g_j^{r_w - b_w r_v}\right)$$

   Note that in the case $C(x^*)_w = 0$,

$$\begin{aligned} r_w - a_w r_u &= c_1^{\ell+1} \prod_{2 \leq i \leq j} c_i + e_w - \left(c_j + d_w\right)\left(c_1^{\ell+1} \prod_{2 \leq i \leq j-1} c_i + n_u\right) \\ &= -c_j n_u - d_w \left(c_1^{\ell+1} \prod_{2 \leq i \leq j-1} c_i\right) - d_w n_u + e_w \end{aligned}$$

Hence, component K_Γ^3 can be computed by pairing j elements from the challenge: $g^{c_1}, g^{\ell}, g^{c_2}, \ldots, g^{c_{j-1}}$. Similarly, for term K_Γ^4.

(b) Else if Γ is an AND gate and $C(x^*)_w = 1$, then values r_w, a_w, b_w are randomly chosen from $\mathbb{Z}_q$. And the adversary computes

$$K_\Gamma = \big(K_\Gamma^1 = g^{a_w}, K_\Gamma^2 = g^{b_w}, K_\Gamma^3 = g_j^{r_w - a_w r_u - b_w r_v}\big)$$

Otherwise, if $C(x^*)_u = 0$, then implicitly set $r_w = c_1^{\ell+1} \prod_{2 \le i \le j} c_i + e_w$, $a_w = c_j + d_w$ where e_w, d_w are randomly chosen. Also, choose b_w at random. Again, the adversary can compute

$$K_\Gamma = \big(K_\Gamma^1 = g^{a_w}, K_\Gamma^2 = g^{b_w}, K_\Gamma^3 = g_j^{r_w - a_w r_u - b_w r_v}\big)$$

Note that,

$$\begin{aligned} r_w - a_w r_u - b_w r_v &= c_1^{\ell+1} \prod_{2 \le i \le j} c_i + e_w - \big(c_j + d_w\big)\big(c_1^{\ell+1} \prod_{2 \le i \le j-1} c_i + n_u\big) - b_w r_v \\ &= e_w - c_j n_u - d_w\big(c_1^{\ell+1} \prod_{2 \le i \le j-1} c_i\big) - d_w n_u - b_w r_v \end{aligned}$$

Hence, K_Γ^3 can be computed by the adversary by applying j pairings to the challenge components $g^{c_1}, g^{\ell}, g^{c_2}, \ldots, g^{c_{j-1}}$ and using the other *known* randomness components.

The adversary performs the symmetric operations if $C(x^*)_v = 0$.

6. Set $\sigma = g_{k-1}^{\alpha - r_n}$. Note that since $C(x^*) = 0$ the component r_n embeds parts challenge into it. Hence, σ can be computed by the adversary due to cancellation in the exponent:

$$\alpha - r_n = c_1^{\ell+1} \prod_{2 \le i \le k-1} c_j + \gamma - c_1^{\ell+1} \prod_{2 \le i \le k-1} c_j + e_n = \gamma + e_n$$

7. Define and output the secret key as

$$\mathsf{sk}_C := \big(C, \{K_\Gamma\}_{g \in C}, \sigma\big)$$

- $\mathsf{Enc}^*(\mathsf{mpk}, x^*, m)$: The encryption algorithm takes the master public key mpk, an index x^* and a message m, and outputs a ciphertext ct_{x^*} defined as follows. Let X be the set of indices i such that $x_i^* = i$. Implicitly let $s = c_k$. Let $\gamma_0 = \gamma = \beta \cdot g_k^{\gamma c_k}$. Output ciphertext as

$$\mathsf{ct}_x := \left(x,\ \gamma_0,\ g^{c_k},\ \gamma_1 = \big(\prod_{i \in X} h_i\big)^{c_k}\right)$$

where b is a randomly chosen bit. Note that $\big(\prod_{i \in X} h_i\big)^s$ can be computed given the challenge component g^{c_k} and known randomness q_i for $i \in X$. If $\beta = g_k^{c_1^{\ell+1} \prod_{2 \le i \le k} c_i}$, then,

$$\begin{aligned} \beta \cdot g_k^{\gamma c_k} &= \big(g_k^{c_1^{\ell+1} \prod_{2 \le i \le k-1}} \cdot g_k^{\gamma}\big)^{c_k} \\ &= \big(g_k^{c_1^{\ell+1} \prod_{2 \le i \le k-1} + \gamma}\big)^{c_k} \\ &= t^{c_k} = t^s \end{aligned}$$

which corresponds to an encryption of 1. Otherwise, if β is a randomly chosen in G_k, this corresponds to an encryption of 0.

The adversary Adv uses the above simulated algorithms to answer the queries to Adv*. If Adv* returns $m = 1$, then Adv outputs that $\beta = g_k^{c_1^{\ell+1} \prod_{2 \le i \le k} c_i}$. Otherwise, it outputs that β is randomly chosen in the target.

7 Applications and Extensions

7.1 Single-Key Functional Encryption and Reusable Garbled Circuits

Goldwasser, Kalai, Popa, Vaikuntanathan and Zeldovich showed how to obtain a Single-Key Functional Encryption (SKFE) and Reusable Garbled Circuits from: (1) Attribute-based Encryption, (2) Fully-Homomorphic Encryption and (3) "one-time" Garbled Circuits [GKP$^+$13b]. In this section we show what we gain in efficiency in the secret key and ciphertext sizes for these two construction by using our ABE schemes.

Theorem 7.1 ([GKP$^+$13b])**.** *There is a (fully/selectively secure) single-key functional encryption scheme $\mathcal{FE}$ for any class of circuits $\mathcal{C}$ that take ℓ bits of input and produce a one-bit output, assuming the existence of (1) $\mathcal{C}$-homomorphic encryption scheme, (2) a (fully/selectively) secure ABE scheme for a related class of predicates and (3) Yao's Garbling Scheme, where:*

1. *The size of the secret key is $2 \cdot \alpha \cdot \mathsf{abe.keysize}$, where $\mathsf{abe.keysize}$ is the size of the ABE key for circuit performing homomorphic evaluation of C and outputting a bit of the resulting ciphertext.*
2. *The size of the ciphertext is $2 \cdot \alpha \cdot \mathsf{abe.ctsize}(\ell \cdot \alpha + \gamma) + \mathrm{poly}(\lambda, \alpha, \beta)$*

where (α, β, γ) denote the sizes of the FHE (ciphertext, secret key, public key), respectively. $\mathsf{abe.keysize}$, $\mathsf{abe.ctsize}(k)$ are the size of ABE secret key, ciphertext on k-bit attribute vector and λ is the security parameter.

Since FHE (and Yao's Garbled Circuits) can also be instantiated assuming the sub-exponential hardness of LWE ([BV11], [BGV12]), we obtain the following corollaries.

Corollary 7.2. *Combining our short secret key ABE construction (Theorem-4.4) and Theorem-7.1, we obtain a single-key functional encryption scheme for a circuit class $\mathcal{C}$ with depth at most $d_{\max}$, where the secret key size is some $\mathrm{poly}(d_{\max}, \lambda)$ and λ is the security parameter.*

To obtain a short ciphertext for functional encryption scheme, we need another observation. There exists a fully-homomorphic encryption scheme where ciphertext encrypting k bits of input is of size $k + \mathrm{poly}(\lambda)$, where λ is the security parameter. We refer the reader to the full version for further details.

Corollary 7.3. *Combining the above observation, our short ciphertext ABE construction (Theorem-6.1) and Theorem-7.1, we obtain a single-key functional encryption scheme for any circuit class $\mathcal{C}$ with depth at most $d_{\max}$ and ℓ bit inputs, where the size of the ciphertext is $\ell + \mathrm{poly}(d_{\max}, \lambda)$ and λ is the security parameter.*

Next, we apply our results to get the optimal construction of reusable garbled circuits.

Theorem 7.4 ([GKP+13b]). *There exists a reusable garbling scheme for any class of circuits $\mathcal{C}$ that take ℓ bits of input, assuming the existence (1) symmetric-encryption algorithm, (2) a single-key functional encryption for $\mathcal{C}$, where:*

1. *The size of the secret key is $|C| + \mathsf{fe.keysize} + \mathrm{poly}(\lambda)$, where $\mathsf{fe.keysize}$ is the size of the FE key for circuit performing symmetric-key decryption and evaluation of C.*
2. *The size of the ciphertext is $\mathsf{fe.ctsize}(\lambda + \ell)$*

where $\mathsf{fe.ctsize}(\lambda + \ell)$ is the size of FE ciphertext on $\lambda + \ell$-bit input.

Corollary 7.5. *From Corollary-7.2 and Theorem-7.4, we obtain a reusable garbled circuits scheme for any class of polynomial-size circuits with depth at most $d_{\max}$, where the secret key size is $|C| + \mathrm{poly}(d_{\max}, \lambda)$.*

Corollary 7.6. *From Corollary-7.3 and Theorem-7.4, we obtain a reusable garbled circuits scheme for any class of polynomial-size circuits with depth at most $d_{\max}$ and ℓ bit inputs, where the ciphertext size is $\ell + \mathrm{poly}(d_{\max}, \lambda)$.*

8 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 $y = f(x_1, \ldots, x_\ell)$ 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.

Acknowledgments. We thank Chris Peikert for his helpful comments and for suggesting Remark 4.3.

D. Boneh is supported by NSF, the DARPA PROCEED program, an AFO SR MURI award, a grant from ONR, an IARPA project provided via DoI/NBC, and Google faculty award. 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.

S. Gorbunov is supported by Alexander Graham Bell Canada Graduate Scholarship (CGSD3).

G. Segev is supported by the European Union's Seventh Framework Programme (FP7) via a Marie Curie Career Integration Grant, by the Israel Science Foundation (Grant No. 483/13), and by the Israeli Centers of Research Excellence (I-CORE) Program (Center No. 4/11).

V. Vaikuntanathan is supported by an NSERC Discovery Grant, DARPA Grant number FA8750-11-2-0225, a Connaught New Researcher Award, an Alfred P. Sloan Research Fellowship, and a Steven and Renee Finn Career Development Chair from MIT.

References

[ABB10] S. Agrawal, D. Boneh, and X. Boyen. Efficient lattice (H)IBE in the standard model. In *EUROCRYPT*, 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 *PKC*, 2012.

[AFV11] S. Agrawal, D. M. Freeman, and V. Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In *ASIACRYPT*, 2011.

[AIKW13] B. Applebaum, Y. Ishai, E. Kushilevitz, and B. Waters. Encoding functions with constant online rate or how to compress garbled circuits keys. In *CRYPTO*, 2013.

[Ajt99] M. Ajtai. Generating hard instances of the short basis problem. In *ICALP*, 1999.

[ALdP11] N. Attrapadung, B. Libert, and E. de Panafieu. Expressive key-policy attribute-based encryption with constant-size ciphertexts. In *Public Key Cryptography*, volume 6571, pages 90–108, 2011.

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

[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 *CRYPTO* '01.

[BGG+14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit abe, and compact garbled circuits. In *Proc. of Eurocrypt'14*, 2014.

[BGV12] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *ITCS*, 2012.

[BGW05] D. Boneh, C. Gentry, and B. Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *CRYPTO*, 2005.

[BHHI10] B. Barak, I. Haitner, D. Hofheinz, and Y. Ishai. Bounded key-dependent message security. In *EUROCRYPT*, 2010.

[BMR90] D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols (extended abstract). In *STOC*, 1990.

[BNS13] Dan Boneh, Valeria Nikolaenko, and Gil Segev. Attribute-based encryption for arithmetic circuits. Cryptology ePrint Archive, Report 2013/669, 2013. http://eprint.iacr.org/.

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

[BS02] D. Boneh and A. Silverberg. Applications of multilinear forms to cryptography. *Contemporary Mathematics*, 324:71–90, 2002.

[BSW11] D. Boneh, A. Sahai, and B. Waters. Functional encryption: Definitions and challenges. In *TCC*, 2011.

[BV11] Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. In *FOCS*, 2011.

[BW07] D. Boneh and B. Waters. Conjunctive, subset, and range queries on encrypted data. In *TCC*, 2007.

[BW13] D. Boneh and B. Waters. Constrained pseudorandom functions and their applications. In *ASIACRYPT*, 2013.

[CHKP10] D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert. Bonsai trees, or how to delegate a lattice basis. In *EUROCRYPT*, 2010.

[CLT13] J. Coron, T. Lepoint, and M. Tibouchi. Practical multilinear maps over the integers. In *CRYPTO*, 2013.

[Coc01] C. Cocks. An identity based encryption scheme based on quadratic residues. In *IMA Int. Conf.*, 2001.

[FN93] A. Fiat and M. Naor. Broadcast encryption. In *CRYPTO*, 1993.

[GGH13a] S. Garg, C. Gentry, and S. Halevi. Candidate multilinear maps from ideal lattices. In *EUROCRYPT*, 2013.

[GGH+13b] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, 2013.

[GGH+13c] S. Garg, C. Gentry, S. Halevi, A. Sahai, and B. Waters. Attribute-based encryption for circuits from multilinear maps. In *CRYPTO*, 2013.

[GGH+13d] Craig Gentry, Sergey Gorbunov, Shai Halevi, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. How to compress (reusable) garbled circuits. Cryptology ePrint Archive, Report 2013/687, 2013. http://eprint.iacr.org/.

[GGP10] R. Gennaro, C. Gentry, and B. Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *CRYPTO*, 2010.

[GGSW13] S. Garg, C. Gentry, A. Sahai, and B. Waters. Witness encryption and its applications. In *STOC*, 2013.

[GHV10] C. Gentry, S. Halevi, and V. Vaikuntanathan. A simple BGN-type cryptosystem from LWE. In *EUROCRYPT*, 2010.

[GKP+13a] S. Goldwasser, Y. T. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich. How to run turing machines on encrypted data. In *CRYPTO*, 2013.

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

[GKR08] S. Goldwasser, Y. T. Kalai, and G. N. Rothblum. Delegating computation: interactive proofs for muggles. In *STOC*, 2008.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC*, 1987.

[GPSW06] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM CCS*, 2006.

[GPV08] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, 2008.

[GVW13] S. Gorbunov, V. Vaikuntanathan, and H. Wee. Attribute-based encryption for circuits. In *STOC*, 2013.

[HW13] S. Hohenberger and B. Waters. Attribute-based encryption with fast decryption. In *PKC*, 2013.

[KS08] V. Kolesnikov and T. Schneider. Improved garbled circuit: Free xor gates and applications. In *ICALP*, 2008.

[KSW08] J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT*, 2008.

[LO13] S. Lu and R. Ostrovsky. How to garble ram programs. In *EUROCRYPT*, 2013.

[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.

[LPRTJ05] 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*, 2012.

[MP12] D. Micciancio and C. Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT*, 2012.

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

[Pei09] C. Peikert. Public-key cryptosystems from the worst-case shortest vector problem. In *STOC*, 2009.

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

[PTMW06] M. Pirretti, P. Traynor, P. McDaniel, and B. Waters. Secure attribute-based systems. In *ACM CCS*, 2006.

[Reg05] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, 2005.

[Sha84] A. Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO*, 1984.

[Sho08] Victor 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*, 2005.

[Wat12] B. Waters. Functional encryption for regular languages. In *CRYPTO*, 2012.

[Yao86] A. C. Yao. How to generate and exchange secrets (extended abstract). In *FOCS*, 1986.