How to Compress (Reusable) Garbled Circuits

Craig Gentry* Sergey Gorbunov† Shai Halevi‡ Vinod Vaikuntanathan§
Dhinakaran Vinayagamurthy¶

December 9, 2013

Abstract

A fundamental question about (reusable) circuit garbling schemes is: how small can the garbled circuit be? Our main result is a reusable garbling scheme which produces garbled circuits that are the same size 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. In contrast, all previous constructions of even single-use garbled circuits incurred a *multiplicative* $\mathsf{poly}(\lambda)$ blowup.

Our techniques result in constructions of attribute-based and (single key secure) functional encryption schemes where the secret key of a depth d circuit C consists of C itself, *plus* $\mathsf{poly}(\lambda, d)$ additional bits. All of these constructions are based on the subexponential hardness of the learning with errors problem.

We also study the dual question of how short the garbled inputs can be, relative to the original input. We demonstrate a (different) reusable circuit garbling scheme, based on multilinear maps, where the size of the garbled input is the same as that of the original input, *plus* a $\mathsf{poly}(\lambda, d)$ factor. Similar to the above, this also results in attribute-based and (single key secure) functional encryption schemes where the size of the ciphertext encrypting an input x is the same as that of x, plus $\mathsf{poly}(\lambda, d)$ additional bits.

*IBM Research. Email: cbgentry@us.ibm.com.

†University of Toronto. Email: sgorbunov@cs.toronto.edu. This work was partially done while visiting IBM T. J. Watson Research Center. Supported by Alexander Graham Bell Canada Graduate Scholarship (CGSD3).

‡IBM Research. Email: shaih@alum.mit.edu.

§MIT and University of Toronto. Email: vinodv@csail.mit.edu. 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.

¶University of Toronto. Email: dhinakaran5@cs.toronto.edu.

1 Introduction

In this paper, we construct reusable garbled circuits, attribute-based encryption and (single key) functional encryption schemes with optimally compact keys and ciphertexts. We begin by describing these cryptographic objects and motivating the questions we study.

Reusable Garbled Circuits. A circuit garbling scheme, one of the most useful primitives in modern cryptography, is a construct originally suggested by Yao in the 1980s in the context of secure two-party computation [Yao86]. This construction relies on the existence of a one-way function to encode an arbitrary circuit C into a garbled circuit $\hat{C}$ and then encode any input x into a garbled input $\hat{x}$; a party given $\hat{C}$ and $\hat{x}$ can obtain $C(x)$. The most important properties of garbled circuits are circuit and input privacy: an adversary, given the garbled circuit and the encoded input, should learn nothing about the circuit C or the input x beyond the result $C(x)$.

A highly desirable feature of garbled circuits is *reusability*. Namely, once the expensive process of garbling a circuit is complete, one would like to use it in conjunction with polynomially many garbled inputs. Unfortunately, Yao's original construction and its variants offered only one-time security. Specifically, providing an encoding of more than one input (for the same garbled circuit) compromises the secrecy of the circuit. Quite recently, Goldwasser, Kalai, Popa, Vaikuntanathan and Zeldovich [GKP+13b] constructed the first fully reusable circuit garbling scheme, based on the learning with errors assumption.

Over the years, garbled circuits and variants have found many applications: two party [Yao86] and multi-party secure protocols [GMW87], one-time programs [GKR08], key-dependent message security [BHHI10], verifiable computation [GGP10], homomorphic computations [GHV10] and many others. Reusable garbling schemes were used to construct token-based program obfuscation schemes and k-time programs [GKP+13b].

A fundamental question regarding (reusable) garbling schemes is:

How small can the garbled circuit be?

Most known constructions of garbled circuits (both single-use and reusable) proceed by garbling each gate to produce a garbled truth table, resulting in a *multiplicative* size blowup of $\mathrm{poly}(\lambda)$. We are aware of three exceptions. The first is the "free XOR" optimization (for single-use garbling schemes) introduced by Kolesnikov and Schneider [KS08] and studied in a sequence of works [CKKZ12, App13]. Here, one produces garbled tables only for the AND gates in the circuit C, still resulting in a multiplicative $\mathsf{poly}(\lambda)$ overhead but proportional to the number of AND gates (as opposed to the total number of gates). The security of the transformation was eventually shown to hold under the learning parity with noise (LPN) assumption [App13]. Secondly, Lu and Ostrovsky [LO13] recently showed how to garble RAM programs (for single use); here, the size of the garbled RAM 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.

Our first (and main) result is a reusable circuit garbling scheme for arbitrary depth-d circuits with an additive overhead, namely $|\hat{G}| = |G| + \mathsf{poly}(\lambda, d)$. Security is based on the subexponential hardness of the learning with errors (LWE) problem [Reg09]. Modulo the dependence on the depth and the specific polynomial dependence on λ, this gives us the shortest possible garbled circuit. 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 1.1 (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 $|\hat{x}| = |x| \cdot \mathsf{poly}(\lambda, d)$.*

This brings us to our next question, namely, how much can we compress the garbled input. Namely, is it possible to ensure that $|\hat{x}| = |x| + \mathsf{poly}(\lambda)$? Indeed, in a beautiful recent work, Applebaum, Ishai, Kushilevitz and Waters [AIKW13] showed a construction of *single-use* garbled circuits with exactly this property. (We remark that while their garbled inputs are short, their garbled circuits still incur a multiplicative overhead.) Can we achieve this in the reusable setting?

While we are not able to compress the garbled inputs for the reusable garbled circuits construction from Theorem 1.1, we come up with a different construction, based on multilinear maps, where $|\hat{x}| = |x| + \mathsf{poly}(\lambda, d)$ where d is maximum depth of circuits that can be computed in conjunction with $\hat{x}$. Our construction relies on a generalization of broadcast encryption [FN93, BGW05, BW13] and the attribute-based encryption scheme of [GGH+13c].

Theorem 1.2 (Informal)**.** *Assuming subexponential LWE and that d-level multilinear maps exist, 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 x into an encoded input $\hat{x}$ such that $|\hat{x}| = |x| + \mathsf{poly}(\lambda, d)$.*

We refer the read to Appendix 5 for the construction, and focus on the short garbled circuits question for the rest of the introduction. A natural question, which we leave open, is that of getting the best of both schemes. Namely, a single scheme which produces both short garbled circuits *and* short garbled inputs.

Our reusable garbling scheme arises out of a new construction of attribute-based encryption scheme with short secret keys, which we describe next.

Attribute-based and Functional Encryption. In an attribute-based encryption scheme [SW05, GPSW06], a message M is encrypted together with a (public) attribute vector $\mathbf{x} \in \{0,1\}^\ell$. The ciphertext can later be decrypted to recover the message M if one has in his possession a secret key SK_P for a predicate P such that $P(\mathbf{x}) = \mathtt{true}$. In a functional encryption scheme [SW05, KSW08, BSW11], given the secret key SK_F for a function F, one can compute $F(\mathbf{x})$ from an encryption of an input $\mathbf{x}$. Moreover, all other information about $\mathbf{x}$ remains hidden. It is easy to see that attribute-based encryption is a special case of functional encryption.

The two properties that are highly desirable for attribute-based and functional encryption schemes are *collusion-resistance* and *succinctness*. Collusion-resistance (for functional encryption) stipulates that a set of users with secret keys $SK_{F_1}, \ldots, SK_{F_n}$ collectively learn nothing about $\mathbf{x}$ other than the function outputs $F_1(\mathbf{x}), \ldots, F_n(\mathbf{x})$. Succinctness requires that the size of the encryption of an input $\mathbf{x}$ should depend polynomially on the size of $\mathbf{x}$, but should be independent of the size of circuits computed on it.

In the past few years, there has been significant progress on constructing attribute-based encryption schemes, in terms of efficiency, security guarantees, and diversifying security assumptions [GPSW06, Wat09, LW10, LOS+10, CHKP12, ABB10a, OT10, Boy13]. The culmination of this line of research resulted in the works of Gorbunov, Vaikuntanathan and Wee [GVW13] who constructed attribute-based encryption for all circuits under the learning with errors (LWE) assumption, and the work of Garg, Gentry, Halevi, Sahai and Waters [GGH+13c] who achieved the goal using multilinear maps [GGH13a].[1] Subsequently, Goldwasser et al. [GKP+13b] gave a general construction of succinct single-key functional encryption schemes and reusable garbled circuits, starting from any attribute-based encryption and (leveled)

[1] Both constructions support circuits of a priori bounded, but polynomial depth.

fully homomorphic encryption schemes. Finally, Garg et al. [GGH+13b] constructed a many-key (also called collusion-resistant) functional encryption scheme, albeit based on a non-standard assumption related to multilinear maps.

In all these constructions, the secret key for a circuit C has $|C| \cdot \mathsf{poly}(\lambda)$ bits, and the encryption of an input $\mathbf{x}$ has $|\mathbf{x}| \cdot \mathsf{poly}(\lambda)$ bits. The only exception we are aware of is the work of Attrapadung, Libert and Panafieu [ALdP11] who construct an attribute-based encryption scheme with short ciphertexts (but not short secret keys) for the special case of Boolean formulas.

We construct:

1. An attribute-based encryption and a functional encryption scheme where the secret key for a circuit C consists of C itself, *plus* $\mathsf{poly}(\lambda, d)$ additional bits. The constructions are based on the subexponential hardness of LWE.

   The ABE schemes are in fact the starting points for all our constructions, including those of the reusable circuit garbling schemes. In a sense, we think of the secret key for an ABE scheme as itself a weak garbling scheme – it provides neither circuit privacy nor input privacy, but what it does provide is some form of authenticity. Once we have an ABE scheme, we apply the compiler of Goldwasser et al. [GKP+13b] to obtain the functional encryption and reusable garbled circuits.

2. An attribute-based encryption and a functional encryption scheme where the size of the encryption of an attribute vector $\mathbf{x}$ is $|\mathbf{x}| + \mathsf{poly}(\lambda, d)$ bits. The ABE is based on multilinear maps, whereas the functional encryption scheme is secure based on both multilinear maps and subexponential LWE.

1.1 Overview of Our Techniques

Our starting point is the observation that *key-homomorphic encryption* can be used as a tool to achieve compression. Informally speaking, a key-homomorphic encryption is one where, given $\mathsf{Enc}_{pk_1}(m_1)$ and $\mathsf{Enc}_{pk_2}(m_2)$, one can compute $\mathsf{Enc}_{pk_1+pk_2}(m_1 + m_2)$. Furthermore, given the secret keys of pk_1 and pk_2, one can compute a secret key for $pk_1 + pk_2$.[2]

This observation has already turned up in a few works. It has been used implicitly in the work of Agrawal, Freeman and Vaikuntanathan [AFV11] who constructed a functional encryption scheme *with short secret keys* for the relatively simple, inner-product checking functions. As we will describe below, their scheme can be built from an additively key homomorphic encryption scheme (with additional properties). More recently, this has also been explicitly exploited in the work of Applebaum, Ishai, Kushilevitz and Waters [AIKW13] who used additively key-homomorphic encryption to build a (single-use) circuit garbling scheme with a compact input encoding. (As we mentioned before, although their input encodings are compact, their garbled circuits grow by a multiplicative $\mathsf{poly}(\lambda)$ factor.)

Let us mention, jumping ahead, that the constructions in this paper arise out of a new method that allows us to not only add ciphertexts encrypted under different keys, but also multiply them to obtain $\mathsf{Enc}_{pk_1 \cdot pk_2}(m_1 \cdot m_2)$. Our main technical contribution is a construction of such an object based on the hardness of LWE. For the moment, though, let us focus on additively key-homomorphic encryption.

The foundations of our constructions arise from the techniques used in the identity-based encryption (IBE) scheme of Agrawal, Boneh and Boyen [ABB10b]. We start by describing the inner product functional

[2] An astute reader would have noticed that the two ciphertexts that we add have to be encrypted using the same – or at least correlated – randomness. A moment of thought reveals that this is necessary to achieve key homomorphism and still maintain semantic security.

encryption scheme of Agrawal et al. [AFV11] which is based on the techniques in [ABB10b]. We start by describing the inner product functional encryption scheme of Agrawal et al. [AFV11] which itself is based on the identity-based encryption (IBE) scheme of Agrawal, Boneh and Boyen [ABB10b]. A ciphertext in the scheme encrypting a vector $\mathbf{x} = (x_1, \ldots, x_\ell) \in \{0,1\}^\ell$ consists of ℓ *encodings*[3] of the form

$$\left(\left(\mathbf{A}_1 + x_1 \cdot \mathbf{B}\right)^T \mathbf{s} + \mathbf{e}_1, \ldots, \left(\mathbf{A}_\ell + x_\ell \cdot \mathbf{B}\right)^T \mathbf{s} + \mathbf{e}_\ell \right)$$

where the random matrices $\mathbf{A}_i \leftarrow \mathbb{Z}_q^{n \times m}$ and the matrix $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$ are part of the public key, the LWE secret $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ is chosen randomly during encryption, the $\mathbf{e}_i$ are "small" LWE error vectors, and all computations are performed modulo a large enough prime q.

How does one check whether $\langle \mathbf{x}, \mathbf{y} \rangle = 0$, given an appropriate secret key $SK_{\mathbf{y}}$? Looking at this as a set of ciphertexts under public keys $\mathbf{A}_1 + x_1\mathbf{B}$, $\mathbf{A}_2 + x_2\mathbf{B}$, $\ldots$, $\mathbf{A}_\ell + x_\ell\mathbf{B}$, we simply take a linear combination of the encodings with coefficients y_i. The result is

$$\sum_{i=1}^{\ell} y_i \cdot \left[\left(\mathbf{A}_i + x_i \cdot \mathbf{B}\right)^T \mathbf{s} + \mathbf{e}_i \right] \approx \left(\sum_{i=1}^{\ell} y_i \mathbf{A}_i + \langle \mathbf{x}, \mathbf{y} \rangle \cdot \mathbf{B} \right)^T \mathbf{s}$$

If $\langle \mathbf{x}, \mathbf{y} \rangle = 0$, then by key homomorphism, we just computed an encoding under the "public key" $\sum_{i=1}^{\ell} y_i \mathbf{A}_i$. Given a secret key for this public key, we can decrypt the resulting ciphertext and recover $\mathbf{s}$ which, in particular, allows us to check if $\langle \mathbf{x}, \mathbf{y} \rangle = 0$ in the first place! A particular feature of interest to us is that the size of the secret key is independent of ℓ, the length of $\mathbf{y}$.

We note that one could use this mechanism to not just check, but also recover a hidden message m in the case that $\langle \mathbf{x}, \mathbf{y} \rangle = 0$ (simply by using a hardcore bit of $\mathbf{s}$ to mask the message).

So far, we just used the additive key-homomorphic features of this encoding scheme. Which raises the natural question: if only we could get both additive and multiplicative key homomorphism, we could use this to design a functional encryption scheme with short secret keys for arbitrary circuits (and not just inner products). We are not quite able to achieve this dream, but we do obtain a restricted type of multiplicative property and consequently an attribute-based encryption scheme with short secret keys. Then, using the reduction of Goldwasser et al. [GKP+13b], we obtain single-key functional encryption with short secret keys as well as compact reusable garbled circuits.

One of the main technical contributions of this work is a novel method of multiplying these encodings. Assume that we want to (key-homomorphically) multiply the encodings $\mathbf{c}_i = \left(\mathbf{A}_i + x_i \cdot \mathbf{B}\right)^T \mathbf{s} + \mathbf{e}_i$ and $\mathbf{c}_j = \left(\mathbf{A}_j + x_j \cdot \mathbf{B}\right)^T \mathbf{s} + \mathbf{e}_j$. This works as follows. Let us first define the transformation $\mathbf{B}^{-1} : \mathbb{Z}_q^{n \times m} \to \mathbb{Z}^{m \times m}$ in such a way that for a matrix $\mathbf{M} \in \mathbb{Z}_q^{n \times m}$, $\mathbf{N} := \mathbf{B}^{-1}(\mathbf{M})$ is a matrix with small entries such that $\mathbf{BN} = \mathbf{M}$.

With this in mind, we compute

$$\mathbf{B}^{-1}(\mathbf{A}_j)^T \cdot \mathbf{c}_i = \mathbf{B}^{-1}(\mathbf{A}_j)^T \cdot \left[\left(\mathbf{A}_i + x_i \cdot \mathbf{B}\right)^T \mathbf{s} + \mathbf{e}_i \right] \approx \left(\mathbf{A}_i \cdot \mathbf{B}^{-1}(\mathbf{A}_j) + x_i \mathbf{A}_j\right)^T \mathbf{s}$$

Secondly, we compute

$$x_i \cdot \mathbf{c}_j = x_i \cdot \left[\left(\mathbf{A}_j + x_j \cdot \mathbf{B}\right)^T \mathbf{s} + \mathbf{e}_j \right] \approx \left(x_i \cdot \mathbf{A}_j + x_i x_j \cdot \mathbf{B}\right)^T \mathbf{s}$$

[3] What follows is a simplified exposition of [AFV11] that ignores some technical details. In particular, the AFV ciphertext consists of $\ell + 1$ encodings, and not ℓ.

Subtracting the two expressions above gives us

$$\big(- \mathbf{A}_i \cdot \mathbf{B}^{-1}(\mathbf{A}_j) + x_i x_j \cdot \mathbf{B} \big)^T \mathbf{s} + \mathsf{noise}$$

which is an encoding of the product $x_i x_j$ under a "public-key" which is a "product" of the two matrices $\mathbf{A}_i$ and $\mathbf{A}_j$. Here, the product is defined to be $-\mathbf{A}_i \cdot \mathbf{B}^{-1}(\mathbf{A}_j)$. The only catch is that performing this homomorphic operation requires us to know x_i, which is precisely the reason for the limitation to (public-index) attribute-based encryption schemes.

Building on this key homomorphic property, we start with matrices $\mathbf{A}_1, \ldots, \mathbf{A}_\ell$ at the input level and recursively define a matrix $\mathbf{A}_C$ for any circuit C. Key-homomorphic computation will then take the input encodings and produce an encoding $(\mathbf{A}_C + C(\mathbf{x}) \cdot \mathbf{B})^T \mathbf{s} + \mathsf{noise}$ which can be decrypted if (1) $C(\mathbf{x}) = 0$, and (2) one knows the trapdoor for $\mathbf{A}_C$. This, with some additional tricks that we use in the security proof, give us the ABE scheme.

Once we construct our attribute-based encryption scheme, we use it to construct functional encryption and reusable garbled circuits. The scheme also has a number of other features. It can be turned into an attribute-based fully homomorphic encryption scheme (following the lines of [GSW13]) and it can support outsourcing of the decryption algorithm (following along the lines of [GHW11, GVW13]). For more details, we refer the reader to Appendix 6.

2 Attribute-Based Encryption

An attribute-based encryption scheme [GPSW06] for a class of predicate circuits $\mathcal{C}$ (namely, circuits with a single bit output) consists of four algorithms $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{Keygen}, \mathsf{Dec})$:

$\mathsf{Params}(1^\lambda, d_{\max}) \to \mathsf{pp}$; The parameter generation algorithm takes the security parameter 1^λ, and the maximum circuit depth $d_{\max}$ and outputs a public parameter pp which is implicitly given to all the other algorithms of the scheme.

$\mathsf{Setup}(1^\ell) \to (\mathsf{mpk}, \mathsf{msk})$: The setup algorithm gets as input the length ℓ of the input index, and outputs the master public key mpk, and the master key msk.

$\mathsf{Enc}(\mathsf{mpk}, x, \mu) \to \mathsf{ct}_x$: The encryption algorithm gets as input mpk, an index $x \in \{0,1\}^\ell$ and a message $\mu \in \mathcal{M}$. It outputs a ciphertext $\mathsf{ct}_{\mathbf{x}}$.

$\mathsf{Keygen}(\mathsf{msk}, C) \to \mathsf{sk}_C$: The key generation algorithm gets as input msk and a predicate specified by $C \in \mathcal{C}$. It outputs a secret key sk_C.

$\mathsf{Dec}(\mathsf{ct}_{\mathbf{x}}, \mathsf{sk}_C) \to \mu$: The decryption algorithm gets as input $\mathsf{ct}_{\mathbf{x}}$ and sk_C, and outputs either $\perp$ or a message $\mu \in \mathcal{M}$.

Definition 2.1 (Correctness)**.** *We require that for all $(\mathbf{x}, C)$ such that $C(\mathbf{x}) = 1$ and for all $\mu \in \mathcal{M}$, we have $\Pr[\mathsf{ct}_{\mathbf{x}} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathbf{x}, \mu); \mathsf{Dec}(\mathsf{ct}_{\mathbf{x}}, \mathsf{sk}_C) = \mu)] = 1$ where the probability is taken over $\mathsf{pp} \leftarrow \mathsf{Params}(1^\lambda, d_{\max}), (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\ell)$ and the coins of all the algorithms in the expression above.*

Definition 2.2 (Security). *For a stateful adversary $\mathcal{A}$, we define the advantage function* $\mathsf{Adv}^{\mathrm{ABE}}_{\mathcal{A}}(\lambda)$ *to be*

$$\Pr\left[b = b' : \begin{array}{l} \mathbf{x}^* \leftarrow \mathcal{A}(1^\lambda, 1^\ell); \\ \mathsf{pp} \leftarrow \mathsf{Params}(1^\lambda, d_{\max}); \\ (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\ell, \mathbf{x}^*); \\ (\mu_0, \mu_1) \leftarrow \mathcal{A}^{\mathsf{Keygen}(\mathsf{msk},\cdot)}(\mathsf{mpk}), |\mu_0| = |\mu_1|; \\ b \xleftarrow{\$} \{0,1\}; \\ \mathsf{ct}_{\mathbf{x}} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathbf{x}, \mu_b); \\ b' \leftarrow \mathcal{A}^{\mathsf{Keygen}(\mathsf{msk},\cdot)}(\mathsf{ct}_{\mathbf{x}}) \end{array}\right] - \frac{1}{2}$$

with the restriction that all queries y that $\mathcal{A}$ makes to $\mathsf{Keygen}(\mathsf{msk}, \cdot)$ *satisfies $C(\mathbf{x}) = 0$ (that is,* sk_C *does not decrypt* $\mathsf{ct}_{\mathbf{x}}$*). An attribute-based encryption scheme is* selectively secure *if for all PPT adversaries $\mathcal{A}$, the advantage* $\mathsf{Adv}^{\mathrm{ABE}}_{\mathcal{A}}(\lambda)$ *is a negligible function in λ. We call an attribute-based encryption scheme* fully secure *if the adversary $\mathcal{A}$ is allowed to choose the challenge index $\mathbf{x}$ after seeing secret keys, namely, along with choosing (μ_0, μ_1).*

3 Preliminaries

Notation For any integer $q \geq 2$, we let $\mathbb{Z}_q$ denote the ring of integers modulo q and we represent $\mathbb{Z}_q$ as integers in $(-q/2, q/2]$. We let $\mathbb{Z}_q^{n \times m}$ denote the set of $n \times m$ matrices with entries in $\mathbb{Z}_q$. We use bold capital letters (e.g. $\mathbf{A}$) to denote matrices, bold lowercase letters (e.g. $\mathbf{x}$) to denote vectors. The notation $\mathbf{A}^\intercal$ denotes the transpose of the matrix $\mathbf{A}$.

If $\mathbf{A}_1$ is an $n \times m$ matrix and $\mathbf{A}_2$ is an $n \times m'$ matrix, then $[\mathbf{A}_1 \| \mathbf{A}_2]$ denotes the $n \times (m + m')$ matrix formed by concatenating $\mathbf{A}_1$ and $\mathbf{A}_2$. A similar notation applies to vectors. When doing matrix-vector multiplication we always view vectors as column vectors.

We say a function $f(n)$ is *negligible* if it is $O(n^{-c})$ for all $c > 0$, and we use $\mathrm{negl}(n)$ to denote a negligible function of n. We say $f(n)$ is *polynomial* if it is $O(n^c)$ for some $c > 0$, and we use $\mathrm{poly}(n)$ to denote a polynomial function of n. We say an event occurs with *overwhelming probability* if its probability is $1 - \mathrm{negl}(n)$.

3.1 Lattice Preliminaries

In this section we collect the results from the literature that we will need for our construction and the proof of security. See Section-3 for the standard notation description.

3.1.1 Learning With Errors (LWE) Assumption

The LWE problem was introduced by Regev [Reg09], who showed that solving it *on the average* is as hard as (quantumly) solving several standard lattice problems *in the worst case*.

Definition 3.1 (LWE). *For an integer $q = q(n) \geq 2$ and an error distribution $\chi = \chi(n)$ over $\mathbb{Z}_q$, the learning with errors problem* $\mathsf{dLWE}_{n,m,q,\chi}$ *is to distinguish between the following pairs of distributions:*

$$\{\mathbf{A}, \mathbf{A}^T\mathbf{s} + \mathbf{x}\} \quad \textit{and} \quad \{\mathbf{A}, \mathbf{u}\}$$

where $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}, \mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n, \mathbf{x} \xleftarrow{\$} \chi^m, \mathbf{u} \xleftarrow{\$} \mathbb{Z}_q^m$.

Connection to lattices. Let $B = B(n) \in \mathbb{N}$. A family of distributions $\chi = \{\chi_n\}_{n\in\mathbb{N}}$ is called B-bounded if

$$\Pr[\chi \in \{-B, \ldots, B-1, B\}] = 1.$$

There are known quantum [Reg09] and classical [Pei09] reductions between $\mathsf{dLWE}_{n,m,q,\chi}$ and approximating short vector problems in lattices in the worst case, where χ is a B-bounded (truncated) discretized Gaussian for some appropriate B. The state-of-the-art algorithms for these lattice problems run in time nearly exponential in the dimension n [AKS01, MV10]; more generally, we can get a 2^k-approximation in time $2^{\tilde{O}(n/k)}$. Combined with the connection to LWE, this means that the $\mathsf{dLWE}_{n,m,q,\chi}$ assumption is quite plausible for a $\mathrm{poly}(n)$-bounded distribution χ and q as large as 2^{n^ϵ} (for any constant $0 < \epsilon < 1$). Throughout this paper, the parameter $m = \mathrm{poly}(n)$, in which case we will shorten the notation slightly to $\mathsf{dLWE}_{n,q,\chi}$.

3.1.2 Trapdoors for Lattices

Gaussian distributions. Let $D_{\mathbb{Z}^m,s,\mathbf{c}}$ be the truncated discrete Gaussian distribution over $\mathbb{Z}^m$ with parameter s and center $\mathbf{c}$, that is, we replace the output by $\mathbf{0}$ whenever the $||\cdot||_\infty$ norm exceeds $\sqrt{m}\cdot s$. For notational convenience, we would assume $\mathbf{c} = \mathbf{0}$ in most cases and hence abbreviate $D_{\mathbb{Z}^m,s,\mathbf{0}}$ to $D_{\mathbb{Z}^m,s}$. Note that $D_{\mathbb{Z}^m,s}$ is $\sqrt{m}\cdot s$-bounded.

The following lemma gives a bound on the length of vectors sampled from a discrete Gaussian. The result follows from [MR07, Lemma 4.4], using [GPV08, Lemma 5.3] to bound the smoothing parameter.

Lemma 3.1. *Let Λ be an n-dimensional lattice, let $\mathbf{T}$ be a basis for Λ, and suppose $s \geq \|\widetilde{\mathbf{T}}\| \cdot \omega(\sqrt{\log n})$. Then for any $\mathbf{c} \in \mathbb{R}^n$ we have*

$$\Pr\left[\, \|\mathbf{x} - \mathbf{c}\| > s\sqrt{n} : \mathbf{x} \xleftarrow{\mathrm{R}} D_{\Lambda,\sigma,\mathbf{c}} \right] \;\leq\; \mathrm{negl}(n)$$

Thus, the probability for a random sample from $D_{\mathbb{Z}^m,s}$ to be $\mathbf{0}$ is negligible.

Lemma 3.2 (Lattice Trapdoors [Ajt99, GPV08, MP12])**.** *There is an efficient randomized algorithm $\mathsf{TrapSamp}(1^n, 1^m, q)$ that, given any integers $n \geq 1$, $q \geq 2$, and sufficiently large $m = \Omega(n \log q)$, outputs a parity check matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and a 'trapdoor' matrix $\mathbf{T} \in \mathbb{Z}^{m\times m}$ such that the distribution of $\mathbf{A}$ is $\mathrm{negl}(n)$-close to uniform and $\|\widetilde{\mathbf{T}}\| \leq O(n \log q)$. Moreover, there is an efficient algorithm $\mathsf{SampleD}$ that with overwhelming probability over all random choices, does the following: For any $\mathbf{u} \in \mathbb{Z}_q^n$, and large enough $s = \Omega(\sqrt{n \log q})$, the randomized algorithm $\mathsf{SampleD}(\mathbf{A}, \mathbf{T}, \mathbf{u}, s)$ outputs a vector $\mathbf{r} \in \mathbb{Z}^m$ with norm $||\mathbf{r}||_\infty \leq ||\mathbf{r}||_2 \leq s\sqrt{n}$ (with probability 1). Furthermore, the following distributions of the tuple $(\mathbf{A}, \mathbf{T}, \mathbf{U}, \mathbf{R})$ are within $\mathrm{negl}(n)$ statistical distance of each other for any polynomial $k \in \mathbb{N}$:*

- $(\mathbf{A}, \mathbf{T}) \leftarrow \mathsf{TrapSamp}(1^n, 1^m, q)$; $\mathbf{U} \leftarrow \mathbb{Z}_q^{n\times k}$; $\mathbf{R} \leftarrow \mathsf{SampleD}(\mathbf{A}, \mathbf{T}, \mathbf{U}, s)$.
- $(\mathbf{A}, \mathbf{T}) \leftarrow \mathsf{TrapSamp}(1^n, 1^m, q)$; $\mathbf{R} \leftarrow (D_{\mathbb{Z}^m,s})^k$; $\mathbf{U} := \mathbf{AR} \pmod q$.

3.1.3 Sampling algorithms

We will use the following algorithms to sample short vectors from specific lattices. Looking ahead, the algorithm $\mathsf{SampleLeft}$ [ABB10a, CHKP12] will be used to sample keys in the real system, while the algorithm $\mathsf{SampleRight}$ [ABB10a] will be used to sample keys in the simulation.

Algorithm $\mathsf{SampleLeft}(\mathbf{A}, \mathbf{B}, \mathbf{T_A}, \mathbf{u}, \alpha)$**:**

Inputs: a full rank matrix $\mathbf{A}$ in $\mathbb{Z}_q^{n\times m}$, a "short" basis $\mathbf{T_A}$ of $\Lambda_q^{\perp}(\mathbf{A})$, a matrix $\mathbf{B}$ in $\mathbb{Z}_q^{n\times m_1}$, a vector $\mathbf{u} \in \mathbb{Z}_q^n$, and a Gaussian parameter α. (1)

Output: Let $\mathbf{F} := (\mathbf{A} \parallel \mathbf{B})$. The algorithm outputs a vector $\mathbf{e} \in \mathbb{Z}^{m+m_1}$ in the coset $\Lambda_{\mathbf{F}+\mathbf{u}}$.

Theorem 3.3 ([ABB10a, Theorem 17], [CHKP12, Lemma 3.2])**.** *Let $q > 2$, $m > n$ and $\alpha > \|\widetilde{\mathbf{T_A}}\| \cdot \omega(\sqrt{\log(m+m_1)})$. Then* $\mathsf{SampleLeft}(\mathbf{A}, \mathbf{B}, \mathbf{T_A}, \mathbf{u}, \alpha)$ *taking inputs as in (1) outputs a vector $\mathbf{e} \in \mathbb{Z}^{m+m_1}$ distributed statistically close to $D_{\Lambda_{\mathbf{F}+\mathbf{u}},\alpha}$, where $\mathbf{F} := (\mathbf{A} \parallel \mathbf{B})$.*

Algorithm $\mathsf{SampleRight}(\mathbf{A}, \mathbf{B}, \mathbf{R}, \mathbf{T_B}, \mathbf{u}, \alpha)$**:**

Inputs: matrices $\mathbf{A}$ in $\mathbb{Z}_q^{n\times k}$ and $\mathbf{R}$ in $\mathbb{Z}^{k\times m}$, a full rank matrix $\mathbf{B}$ in $\mathbb{Z}_q^{n\times m}$, a "short" basis $\mathbf{T_B}$ of $\Lambda_q^{\perp}(\mathbf{B})$, a vector $\mathbf{u} \in \mathbb{Z}_q^n$, and a Gaussian parameter α. (2)
Output: Let $\mathbf{F} := (\mathbf{A} \parallel \mathbf{AR} + \mathbf{B})$. The algorithm outputs a vector $\mathbf{e} \in \mathbb{Z}^{m+k}$ in the coset $\Lambda_{\mathbf{F}+\mathbf{u}}$.

Often the matrix $\mathbf{R}$ given to the algorithm as input will be a random matrix in $\{1,-1\}^{m\times m}$. Let S^m be the m-sphere $\{\mathbf{x} \in \mathbb{R}^{m+1} : \|\mathbf{x}\| = 1\}$. We define $s_R := \|\mathbf{R}\| := \sup_{\mathbf{x}\in S^{m-1}} \|\mathbf{R}\cdot\mathbf{x}\|$.

Theorem 3.4 ([ABB10a, Theorem 19])**.** *Let $q > 2, m > n$ and $\alpha > \|\widetilde{\mathbf{T_B}}\| \cdot s_R \cdot \omega(\sqrt{\log m})$. Then* $\mathsf{SampleRight}(\mathbf{A}, \mathbf{B}, \mathbf{R}, \mathbf{T_B}, \mathbf{u}, \alpha)$ *taking inputs as in (2) outputs a vector $\mathbf{e} \in \mathbb{Z}^{m+k}$ distributed statistically close to $D_{\Lambda_{\mathbf{F}+\mathbf{u}},\alpha}$, where $\mathbf{F} := (\mathbf{A} \parallel \mathbf{AR} + \mathbf{B})$.*

3.1.4 Additional algorithms

We will also use the following two algorithms, throughout our paper. Let $y = \lfloor \log q \rfloor + 1$.

- $\mathsf{Powersof2}(a)$: The algorithm takes an element $a \in \mathbb{Z}_q$ as input and outputs $\mathbf{a}' = [2^0 a, 2^1 a, \ldots, 2^{y-1} a]$. Given a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times n}$ as input, $\mathsf{Powersof2}(\mathbf{A})$ returns a matrix in $\mathbb{Z}_q^{n\times ny}$, with Powersof2 algorithm applied to each element of $\mathbf{A}$.
- $\mathsf{BD}(a)$: The algorithm takes in an element $a \in \mathbb{Z}_q$ and outputs a vector $\mathbf{a}^T = [a_0, \ldots, a_{y-1}]$, where a_i is the ith bit of the binary decomposition of a ordered from LSB to MSB. Given a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ as input, $\mathsf{BD}(\mathbf{A})$ returns a $ny \times m$ matrix with BD applied to each element of $\mathbf{A}$.

We note that for any two matrices $\mathbf{A} \in \mathbb{Z}_q^{n\times n}, \mathbf{B} \in \mathbb{Z}_q^{n\times m}$: $\mathsf{Powersof2}(\mathbf{A}) \cdot \mathsf{BD}(\mathbf{B}) = \mathbf{AB}$. We will frequently use matrix $\mathbf{P} = \mathsf{Powersof2}(\mathbf{I}_n)$, where $\mathbf{I}_n$ is an $n \times n$ identity matrix.

3.2 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 3.2 ((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.

4 ABE with Short Secret Key from LWE

In this section we show how to construct an Attribute-Based Encryption based on the hardness of the standard Learning-With-Errors (LWE) assumption. As opposed to [GVW13] (where there are two *randomly chosen* matrices assigned for each wire and the transformation key for gate consists of 4 "recoding" matrices) in our construction the matrices for each wire are assigned as *functions* of the input matrices fixed at the setup and there are **no** explicit "recoding" matrices. We now define algorithms $(\mathsf{Params}, \mathsf{Setup}, \mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec})$ for a family of circuits $\mathcal{C}$ of bounded depth $d_{\max}$.

- $\mathsf{Params}(1^\lambda, d_{\max})$: Let $n = n(\lambda, d_{\max})$, $q = q(n, d_{\max})$ and $m = m(n, d_{\max})$. Set the error distribution $\chi = \chi(n)$ and the error bound $B = B(n)$. We also additionally choose two Gaussian parameters: a "small" Gaussian parameter $s = s(n)$ (which the reader should think of as polynomially bounded), and a "large" Gaussian parameter $\alpha = \alpha(n, d_{\max})$ (which the reader should think of as growing exponentially in $d_{\max}$.) Output the global public parameters $\mathsf{pp} = (n, \chi, B, q, m, s, \alpha)$. This is implicitly given to all of the algorithms defined below[4].

- $\mathsf{Setup}(1^\ell)$:

  1. Run two instances of the trapdoor generation algorithm $\mathsf{TrapGen}(1^n, 1^m, q)$ to obtain $(\mathbf{A}, \mathbf{T_A})$ and $(\mathbf{B}, \mathbf{T_B})$.
  2. Choose ℓ matrices $\{\mathbf{A}_i\}_{i \in [\ell]}$ at random from $\mathbb{Z}_q^{n \times m}$.
  3. Choose a vector $\mathbf{u}$ at random from $\mathbb{Z}_q^n$.
  4. Sample a matrix $\mathbf{R_B} \in \mathbb{Z}_q^{m \times n(\lfloor \log q \rfloor + 1)}$ by running $\mathsf{SampleD}(\mathbf{B}, \mathbf{T_B}, \mathbf{P} = \mathsf{Powersof2}(\mathbf{I}_n), s)$ such that $\mathbf{B} \cdot \mathbf{R_B} = \mathbf{P}$.

[4] See Sections 4.1 and B for concrete instantiation of the parameters.

5. Define and output the master public key as

$$\mathsf{mpk} := \left(\mathbf{A}, \{\mathbf{A}_i\}_{i\in[\ell]}, \mathbf{B}, \mathbf{R}_\mathbf{B}, \mathbf{u}\right)$$

and the master secret key as $\mathsf{msk} := (\mathbf{T}_\mathbf{A})$.

- $\mathsf{Enc}(\mathsf{mpk}, \mathbf{x} = (x_1, \ldots, x_\ell), \mu)$: For encrypting a message $\mu \in \{0,1\}$, do the following:

1. Choose a vector $\mathbf{s} \in \mathbb{Z}_q^n$ at random.
2. Choose a noise term $\mathbf{e} \leftarrow \chi^m$ and compute $\psi_0 = \mathbf{A}^T\mathbf{s} + \mathbf{e}$.
3. For all input wires $i \in [\ell]$:
   (a) Choose a random matrix $\mathbf{R}_i \leftarrow \{-1,1\}^{m\times m}$ and let $\mathbf{e}_i = \mathbf{R}_i^T\mathbf{e}$.
   (b) Compute $\psi_i = (\mathbf{A}_i + x_i\mathbf{B})^T\mathbf{s} + \mathbf{e}_i$.
4. Encrypt the message μ as $\tau = \mathbf{u}^T\mathbf{s} + e + \lfloor q/2 \rfloor \mu$, where $e \leftarrow \chi$.
5. Output the ciphertext as $\mathsf{ct}_\mathbf{x} = \left(\mathbf{x}, \psi_0, \{\psi_i\}_{i\in[\ell]}, \tau\right)$.

- $\mathsf{Keygen}(\mathsf{msk}, C)$:

1. Inductively, from input to output, consider a gate $g = (u, v; w)$. Without loss of generality assume g is a binary NAND gate. The matrices for the input wires (u, v) are fixed as $\mathbf{A}_u, \mathbf{A}_v$ by induction and the matrix for the output wire w is assigned the value

$$\mathbf{A}_w = \mathbf{A}_v \cdot \mathbf{R}_\mathbf{B} \cdot \mathsf{BD}(\mathbf{A}_u) - \mathbf{B}$$

2. Finally, let $\mathbf{A}_{\text{out}}$ be the matrix defined at the output wire by this process. Define $\mathbf{F} = [\mathbf{A}||(\mathbf{A}_{\text{out}} + \mathbf{B})] \in \mathbb{Z}_q^{n\times 2m}$. Compute $\mathbf{r}_{\text{out}} \in \mathbb{Z}_q^{2m}$ by running $\mathsf{SampleLeft}(\mathbf{A}, (\mathbf{A}_{\text{out}} + \mathbf{B}), \mathbf{T}_\mathbf{A}, \mathbf{u}, \alpha)$, satisfying $\mathbf{F} \cdot \mathbf{r}_{\text{out}} = \mathbf{u}$.
3. Output the secret key for the circuit C, $\mathsf{sk}_C := (C, \mathbf{r}_{\text{out}})$.

- $\mathsf{Dec}(\mathsf{sk}_C, \mathsf{ct}_\mathbf{x})$: If $C(\mathbf{x}) = 0$, output $\perp$. Otherwise, proceed the evaluation from input to output as follows:

1. Consider a gate $g = (u, v; w)$ carrying input values x_u, x_v and hence output value $x_w = x_u \ \mathsf{NAND}\ x_v = 1 - x_u x_v$. By induction, the user holds $\psi_u = (\mathbf{A}_u + x_u\mathbf{B})^T\mathbf{s} + \mathbf{e}_u$ and $\psi_v = (\mathbf{A}_v + x_v\mathbf{B})^T\mathbf{s} + \mathbf{e}_v$ (for some error vectors $\mathbf{e}_u$ and $\mathbf{e}_v$).
   Compute ψ_w as follows:

$$\psi_w = \left(\mathbf{R}_\mathbf{B} \cdot \mathsf{BD}(\mathbf{A}_u)\right)^T\psi_v - x_v\psi_u$$

   As we show below (in Lemma 4.1), this new ciphertext has the form $\left(\mathbf{A}_w + x_w\mathbf{B}\right)^T\mathbf{s} + \mathbf{e}_w$ (for a small enough noise term $\mathbf{e}_w$).
2. Finally, at the output gate the user computes

$$\psi_{\text{out}} = (\mathbf{A}_{\text{out}} + C(\mathbf{x}) \cdot \mathbf{B})^T\mathbf{s} + \mathbf{e}_{\text{out}} = (\mathbf{A}_{\text{out}} + \mathbf{B})^T\mathbf{s} + \mathbf{e}_{\text{out}}$$

   (since $C(\mathbf{x}) = 1$).
3. Compute $\beta = \mathbf{r}_{\text{out}}^T \cdot [\psi||\psi_{\text{out}}]$ As we show below (in Lemma 4.1), $\beta \approx \mathbf{u}^T\mathbf{s} \mod q$.
   Output $\mu = 0$ if $|\tau - \beta| < q/4$ and $\mu = 1$ otherwise.

4.1 Correctness and Compactness

Lemma 4.1 (Correctness)**.** *Let $\mathcal{C}$ be a family of circuits with their depth bounded by $d_{\max}$ and let $\mathcal{ABE} = (\mathsf{Params}, \mathsf{Setup}, \mathsf{Enc}, \mathsf{Keygen}, \mathsf{Dec})$ be our attribute-based encryption. Assuming that, for a LWE dimension $n = n(\lambda, d_{\max})$, the parameters for $\mathcal{ABE}$ are instantiated as follows*[5]*:*

$$\chi = D_{\mathbb{Z},\sqrt{n}} \qquad B = O(n)$$
$$q = \tilde{O}(nd_{\max})^{O(d_{\max})} \qquad s = O(\sqrt{n \log q})$$
$$m = O(n \log q) \qquad \alpha = O(n \log q)^{O(d_{\max})}$$

then the scheme $\mathcal{ABE}$ is correct, according to Definition 2.1.

Proof. Let us consider a circuit $C \in \mathcal{C}$ of depth atmost $d_{\max}$, such that $C(\mathbf{x}) = 1$. Informally, we have to prove that the encoding obtained at the output wire is of the "correct" form and that the noise component e of β is "small" enough.

Claim 4.1.1. *For each encoding ψ_u for wire u at level j, the user holds $(\mathbf{A}_u + x_u\mathbf{B})^T\mathbf{s} + \mathbf{e}_u$, where $||\mathbf{e}_u||_\infty \leq B \cdot m^{2j+1}(2s\sqrt{m})^j$.*

Proof. First, note that when $\mathbf{e} \leftarrow \chi^m$, $||\mathbf{e}||_\infty \leq B$ by the definition of χ and B. Hence, for the noise term $\mathbf{e}_i = \mathbf{R}_i^T\mathbf{e}$, $||\mathbf{e}_i||_\infty \leq mB$ since $\mathbf{R}_i \in \{-1, 1\}^{m\times m}$. Thus, the base case for the input encodings holds.

- Now, consider a gate $g = (u, v, w)$ and any two input encodings $\psi_u = (\mathbf{A}_u + x_u\mathbf{B})^T\mathbf{s} + \mathbf{e}_u, \psi_v = (\mathbf{A}_v + x_v\mathbf{B})^T\mathbf{s} + \mathbf{e}_v$ at depths j_0, j_1 respectively, where $||\mathbf{e}_u||_\infty \leq B \cdot m^{2j_0+1}(2s\sqrt{m})^{j_0}$ and $||\mathbf{e}_1||_\infty \leq B \cdot m^{2j_1+1}(2s\sqrt{m})^{j_1}$. Then, the recoded encoding ψ_w is computed as follows:

$$\begin{aligned}
\psi_w &= (\mathbf{R_B} \cdot \mathsf{BD}(\mathbf{A}_u))^T \psi_v - x_v\psi_u \\
&= \left(\mathbf{A}_v \cdot \mathbf{R_B} \cdot \mathsf{BD}(\mathbf{A}_u) + x_v\mathbf{B} \cdot \mathbf{R_B} \cdot \mathsf{BD}(\mathbf{A}_u)\right)^T\mathbf{s} + (\mathbf{R_B} \cdot \mathsf{BD}(\mathbf{A}_u))^T \mathbf{e}_v \\
&\quad -x_v \left(\left(\mathbf{A}_u + x_u\mathbf{B}\right)^T\mathbf{s} + \mathbf{e}_u\right) \\
&= \left(\mathbf{A}_v \cdot \mathbf{R_B} \cdot \mathsf{BD}(\mathbf{A}_u) + x_v\mathbf{A}_u\right)^T\mathbf{s} + (\mathbf{R_B} \cdot \mathsf{BD}(\mathbf{A}_u))^T \mathbf{e}_v - x_v \left(\left(\mathbf{A}_u + x_u\mathbf{B}\right)^T\mathbf{s} + \mathbf{e}_u\right) \\
&= \left(\mathbf{A}_w + (1 - x_ux_v) \cdot \mathbf{B}\right)^T\mathbf{s} + \mathbf{e}_w
\end{aligned}$$

  where the last equation is because $\mathbf{A}_w = \mathbf{A}_v \cdot \mathbf{R_B} \cdot \mathsf{BD}(\mathbf{A}_u) - \mathbf{B}$. Also, we define $\mathbf{e}_w := (\mathbf{R_B} \cdot \mathsf{BD}(\mathbf{A}_u))^T \mathbf{e}_v - x_v\mathbf{e}_u$.

- Thus,

$$\begin{aligned}
||\mathbf{e}_w||_\infty &\leq m \cdot \|\mathbf{R_B} \cdot \mathsf{BD}(\mathbf{A}_u)\|_{\max} \cdot ||\mathbf{e}_u||_\infty + ||\mathbf{e}_v||_\infty \\
&\leq m \cdot \left((ms\sqrt{m}) \cdot (B \cdot m^{2j_0+1}(2s\sqrt{m})^{j_0}) + B \cdot m^{2j_1+1}(2s\sqrt{m})^{j_1}\right) \\
&\leq B \cdot (m^2 \cdot m^{2j_0+1}2^{j_0}(s\sqrt{m})^{j_0+1} + m \cdot m^{2j_1+1}(2s\sqrt{m})^{j_1}) \\
&\leq B \cdot m^{2\left(\max(j_0,j_1)+1\right)+1} \cdot (2s\sqrt{m})^{\max(j_0,j_1)+1}
\end{aligned}$$

  as claimed (the second inequality is because $\|\mathbf{R_B} \cdot \mathsf{BD}(\mathbf{A}_u)\|_{\max} \leq nys\sqrt{m} \leq ms\sqrt{m}$).

[5] We refer the reader to Section-B for derivation of these parameters.

□

Hence, by Claim-4.1.1, if $C(\mathbf{x}) = 1$, the user obtains $\psi_{\text{out}} = (\mathbf{A}_{\text{out}} + C(\mathbf{x}) \cdot \mathbf{B})^T\mathbf{s} + \mathbf{e}_{\text{out}} = (\mathbf{A}_{\text{out}} + \mathbf{B})^T\mathbf{s} + \mathbf{e}_{\text{out}}$, where $\|\mathbf{e}_{\text{out}}\|_\infty \leq B \cdot m^{2d_{\max}+1}(2s\sqrt{m})^{d_{\max}}$. After final re-encoding, the user obtains $\beta := \mathbf{r}_{\text{out}}^T[\psi^*||\psi_{\text{out}}] = \mathbf{u}^T\mathbf{s} + e_f$, where $e_f = [\mathbf{e}^T||\mathbf{e}_{\text{out}}^T]\mathbf{r}_{\text{out}}$. Here, $\mathbf{r}_{\text{out}}$ is the output of SampleLeft algorithm (Algorithm 1. Hence, it has an infinite norm $\|\mathbf{r}\text{out}\|_\infty \leq \alpha\sqrt{m} \leq O(n\log q)^{O(d_{\max})}$. Thus, $\|e_f\|_\infty \leq m \cdot (\|\mathbf{e}\|_\infty + \|\mathbf{e}_{\text{out}}\|_\infty) \cdot \|\mathbf{r}_{\text{out}}\|_\infty \leq O(B \cdot O(n\log q)^{O(d_{\max})})$
Also, ciphertext component τ is computed using noise term $||e||_\infty \leq B$. Hence,

$$||e_f - e||_\infty \leq O(B \cdot O(n\log q)^{O(d_{\max})}) < q/4,$$

which holds given the above setting of the parameters.Thus, ψ and ψ' are "close", message $\mu \in \{0,1\}$ is decoded correctly as required. □

Corollary 4.2. *For any depth $d_{\max}$ family of circuits $\mathcal{C}$, the secret key size in our Attribute-Based Encryption is $O(n\log q) = \text{poly}(d_{\max}, \lambda)$.*

4.2 Security Proof

Theorem 4.3 (Selective security)**.** *For all ℓ and polynomial $d_{\max} = d_{\max}(\ell)$, there exists a selectively-secure attribute-based encryption for any family of polynomial-size circuits with ℓ inputs and depth at most $d_{\max}$ with constant secret key size (independent on the number of gates in the circuits), assuming hardness of $\mathsf{dLWE}_{n,q,\chi}$ for sufficiently large $n = \text{poly}(\lambda, d_{\max}), q = n^{O(d_{\max})}$ and $\text{poly}(n)$ bounded error distribution χ.*

Proof. We follow the security strategy of [ABB10a, AFV11] to prove the security of our construction. In particular, we define a series of hybrid games, where the first and last games correspond to the real experiments encrypting messages μ_0, μ_1, respectively. We show that these games are indistinguishable. Recall that in the selective security game, the challenge $\mathbf{x}^*$ is declared before the Setup algorithm and all circuit queries C must be such that $C(\mathbf{x}^*) = 0$. Now, consider the following simulated ABE algorithms:

- $\mathsf{Setup}^*\big(1^\lambda, \mathbf{x}^* = (x_1^*, \ldots, x_\ell^*)\big)$: Takes as input the challenge $\mathbf{x}^*$ and does the following:
  1. Choose a random matrix $\mathbf{A}^* \in \mathbb{Z}_p^{n\times m}$ and a random vector $\mathbf{u} \in \mathbb{Z}_p^n$.
  2. Run the trapdoor generation algorithm $\mathsf{TrapGen}(1^n, 1^m, q)$ to obtain $(\mathbf{B}, \mathbf{T_B})$.
  3. Sample an $m \times n\,(\lfloor \log q \rfloor + 1)$ matrix $\mathbf{R_B} \leftarrow \mathsf{SampleD}(\mathbf{B}, \mathbf{T_B}, \mathbf{P} = \mathsf{Powersof2}(\mathbf{I}_n), s)$ such that $\mathbf{B} \cdot \mathbf{R_B} = \mathbf{P}$.
  4. For all $i \in [\ell]$, sample a short matrix $\mathbf{R}_i \in \{-1, 1\}^{m\times m}$ at random and let $\mathbf{A}_i = \mathbf{A}^*\mathbf{R}_i - x_i^*\mathbf{B}$.
  5. Define and output the master public key as

$$\mathsf{mpk} := \big(\mathbf{A}^*, \{\mathbf{A}_i\}_{i\in[\ell]}, \mathbf{B}, \mathbf{R_B}, \mathbf{u}\big)$$

- $\mathsf{Enc}^*\big(\mathsf{mpk}, x^* = (x_1^*, \ldots, x_\ell^*), \mu\big)\big)$: For encrypting a message $\mu \in \{0,1\}$, do the following:
  1. Choose a vector $\mathbf{s} \in \mathbb{Z}_q^n$ at random.
  2. Choose a noise term $\mathbf{e} \leftarrow \chi^m$ and compute $\psi^* = (\mathbf{A}^*)^T\mathbf{s} + \mathbf{e}$.

3. For all input wires $i \in [\ell]$:
   (a) Compute $\psi_i = (\mathbf{A}_i + x^*_i \mathbf{B})^T \mathbf{s} + \mathbf{R}_i^T \mathbf{e}$, where $\mathbf{R}_i$ is the matrix chosen at the setup.
4. Encrypt the message as $\tau = \mathbf{u}^T \mathbf{s} + e + \lfloor q/2 \rfloor \mu$, where $e \leftarrow \chi$.
5. Output the ciphertext as $\mathsf{ct}_{\mathbf{x}^*} = \left(\mathbf{x}^*, \psi^*, \{\psi_i\}_{i \in [\ell]}, \tau\right)$.

- $\mathsf{Keygen}^*(\mathsf{msk}, C)$:

1. The adversary will ensure the following induction: for every wire u, $\mathbf{A}_u = \mathbf{A}^* \mathbf{R}_u - C(\mathbf{x}^*)_u \mathbf{B}$, where $\mathbf{R}$ is a short matrix, and $C(\mathbf{x}^*)_u$ is the value carried on the wire u by evaluating C on input $\mathbf{x}^*$. Now, inductively, from input to output, consider a gate $g = (u, v; w)$. The matrices for the input wires (u, v) are fixed as $\mathbf{A}_u, \mathbf{A}_v$ by induction and assign the matrix $\mathbf{A}_w$ for the output wire w to be $\mathbf{A}_w = \mathbf{A}_v \cdot \mathbf{R}_{\mathbf{B}} \cdot \mathsf{BD}(\mathbf{A}_u) - \mathbf{B}$.

   Note that, when $\mathbf{A}_w$ is assigned this way,

$$\begin{aligned}
\mathbf{A}_w &= (\mathbf{A}^* \mathbf{R}_v - C(\mathbf{x}^*)_v \mathbf{B}) \cdot \mathbf{R}_{\mathbf{B}} \cdot \mathsf{BD}(\mathbf{A}_u) - \mathbf{B} \\
&= \mathbf{A}^* \mathbf{R}_v \cdot \mathbf{R}_{\mathbf{B}} \cdot \mathsf{BD}(\mathbf{A}_u) - C(\mathbf{x}^*)_v \mathbf{A}_u - \mathbf{B} \\
&= \mathbf{A}^* \mathbf{R}_v \cdot \mathbf{R}_{\mathbf{B}} \cdot \mathsf{BD}(\mathbf{A}_u) - C(\mathbf{x}^*)_v (\mathbf{A}^* \mathbf{R}_u - C(\mathbf{x}^*)_u \mathbf{B}) - \mathbf{B} \\
&= \mathbf{A}^* (\mathbf{R}_v \cdot \mathbf{R}_{\mathbf{B}} \cdot \mathsf{BD}(\mathbf{A}_u) - C(\mathbf{x}^*)_v \mathbf{R}_u) + (C(\mathbf{x}^*)_u C(\mathbf{x}^*)_v - 1)\mathbf{B} \\
&= \mathbf{A}^* \mathbf{R}_w - (1 - C(\mathbf{x}^*)_u C(\mathbf{x}^*)_v)\mathbf{B} \\
&= \mathbf{A}^* \mathbf{R}_w - C(\mathbf{x}^*)_w \mathbf{B}
\end{aligned}$$

2. Therefore, by the above inductive property $\mathbf{A}_{\text{out}} = \mathbf{A}^* \mathbf{R} - C(\mathbf{x}^*)\mathbf{B}$ for some *known* matrix $\mathbf{R} \in \mathbb{Z}^{m \times m}$. Since $C(\mathbf{x}^*) = 0$, we have $\mathbf{A}_{\text{out}} = \mathbf{A}^* \mathbf{R}$. Let $\mathbf{F} = [\mathbf{A}^* || (\mathbf{A}_{\text{out}} + \mathbf{B})] = [\mathbf{A}^* || \mathbf{A}^* \mathbf{R} + \mathbf{B}]$. Compute $\mathbf{r}_{\text{out}} \in \mathbb{Z}^{2m}$ by running $\mathsf{SampleRight}(\mathbf{A}^*, \mathbf{B}, \mathbf{R}, \mathbf{T}_{\mathbf{B}}, \mathbf{u}, \alpha)$, satisfying $\mathbf{F} \cdot \mathbf{r}_{\text{out}} = \mathbf{u}$.
3. The secret key for the circuit C is $\mathsf{sk}_C := (C, \mathbf{r}_{\text{out}})$.

Game Sequence We now define a series of games and then prove that all games **Game i** and **Game i+1** are either statistically or computationally indistinguishable.

- **Game 0**: The challenger runs the real ABE algorithms and encrypts message μ_0 for the challenge index $\mathbf{x}^*$.
- **Game 1**: The challenger runs the simulated ABE algorithms $\mathsf{Setup}^*, \mathsf{Keygen}^*, \mathsf{Enc}^*$ and encrypts message μ_0 for the challenge index $\mathbf{x}^*$.
- **Game 2**: The challenger runs the simulated ABE algorithms $\mathsf{Setup}^*, \mathsf{Keygen}^*$, but chooses a uniformly random element of the ciphertext space for challenge index $\mathbf{x}^*$.
- **Game 3**: The challenger runs the simulated ABE algorithms $\mathsf{Setup}^*, \mathsf{Keygen}^*, \mathsf{Enc}^*$ and encrypts message μ_1 for the challenge index $\mathbf{x}^*$.
- **Game 4**: The challenger runs the real ABE algorithms and encrypts message μ_1 for the challenge index $\mathbf{x}^*$.

Lemma 4.4. *The view of an adversary in* **Game 0** *is statistically indistinguishable from* **Game 1**. *Similarly, the view of an adversary in* **Game 3** *is statistically indistinguishable from* **Game 4**.

Proof. We prove for the case of **Game 0** and **Game 1**, as the other case is identical. First, note the differences between the games:

- In **Game 0**, matrix $\mathbf{A}^*$ is sampled using $\mathsf{TrapGen}$ algorithm and matrices $\mathbf{A}_i \in \mathbb{Z}_p^{m \times m}$ are randomly chosen. In **Game 1**, matrix $\mathbf{A}^* \in \mathbb{Z}_p^{n \times m}$ is chosen uniformly at random and matrices $\mathbf{A}_i = \mathbf{A}^* \mathbf{R}_i - x_i^* \mathbf{B}$ for uniformly random $\mathbf{R}_i \in \{-1, 1\}^{m \times m}$.
- In **Game 0**, each ciphertext component is computed as:

$$\psi_i = (\mathbf{A}_i + x_i^* \mathbf{B})^T \mathbf{s} + \mathbf{e}_i = (\mathbf{A}_i + x_i^* \mathbf{B})^T \mathbf{s} + \mathbf{R}_i^T \mathbf{e}$$

  On the other hand, in **Game 1** the ciphertext is computed as:

$$\psi_i = (\mathbf{A}_i + x_i^* \mathbf{B})^T \mathbf{s} + \mathbf{R}_i^T \mathbf{e} = (\mathbf{A}^* \mathbf{R}_i)^T \mathbf{s} + \mathbf{R}_i^T \mathbf{e} = \mathbf{R}_i^T \big((\mathbf{A}^*)^T \mathbf{s} + \mathbf{e} \big)$$

- Finally, in **Game 0** the vector $\mathbf{r}_{\text{out}}$ is sampled using $\mathsf{SampleLeft}$, whereas in **Game 1** it is sampled using $\mathsf{SampleRight}$ algorithm.

For sufficiently large α (See Section-B), the distributions produced in two games are statistically indistinguishable which follows readily from [AFV11, Lemma 4.3], Theorem-3.3 and Theorem-3.4.

□

Lemma 4.5. *If the decisional* LWE *assumption holds, then the view of an adversary in* **Game 1** *is computationally indistinguishable from* **Game 2**. *Similarly, if the decisional* LWE *assumption holds, then the view of an adversary in* **Game 2** *is computationally indistinguishable from* **Game 3**.

Proof. Assume there exist an adversary Adv that distinguishes between **Game 1** and **Game 2**. We show how to break LWE problem given a challenge $\{(\mathbf{a}_i, y_i)\}_{i \in [m+1]}$ where $y_i = \mathbf{a}_i^T \cdot \mathbf{s} + e_i$ (for a fixed, random $\mathbf{s} \in \mathbb{Z}_q^n$ and a noise term sampled from the error distribution $e_i \leftarrow \chi$) or y_i are random samples in $\mathbb{Z}_q$. Let $\mathbf{A}^* = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_m] \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{u} = \mathbf{a}_{m+1}$. Let $\psi^* = [y_1, y_2, \ldots, y_m]$ and $\tau = y_{m+1} + \mu \lfloor q/2 \rfloor$.

Now, run the simulated Setup^* algorithm where $\mathbf{A}^*, \mathbf{u}$ are as defined above. Run the simulated Keygen^* algorithm. Finally, to simulate the challenge ciphertext set ψ^*, τ as defined above and compute

$$\psi_i = \mathbf{R}_i^T \psi^* = \mathbf{R}_i^T \big((\mathbf{A}^*)^T \mathbf{s} + \mathbf{e} \big)$$

Note that if y_i's are LWE samples, then this corresponds exactly to the **Game 1**. Otherwise, the ciphertext corresponds to an independent random sample by the left-over hash lemma.

□

To conclude, note that **Game 0** always corresponds to an encryption of the challenge message μ_0 in the real experiment and **Game 4** corresponds to an encryption of the challenge message μ_1 (also in the real experiment). Hence, by the standard hybrid argument, no adversary can distinguish between encryptions of μ_0 and μ_1 establishing the selective security of our ABE scheme.

□

5 ABE with Short Ciphertext from Mmaps

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]).

Applebaum, Ishai, Kushilevitz and Waters showed how to compress "one-time" garbled keys using an additively key-homomorphic encryption [AIKW13]. In particular, the "compressed" key is a summation of keys corresponding to the set of *active* inputs to the circuit. This approach works only for "one-time" evaluation of garbled circuits. Next, we view the construction of ABE through the lens of (weakly) "reusable" garbled circuits proposed in [GVW13], where for each gate the user performs double-key proxy re-encryption (referred to as TOR). Similarly, we generalize [AIKW13] in a novel way to support reusability: instead of decrypting, we perform re-encryption where randomness source s is specified by the ABE encryptor. Hence, our mechanism can be viewed as achieving key-homomorphic proxy re-encryption. Given $(\prod_{i\in X} h_i)^s$ in the ciphertext viewed as an encryption of 0 using randomness s under the product of keys h_i, the user obtains $g_2^{sr_i}$ for all active wires i, which can be viewed as an encryption of 0 using randomness s and key $g_2^{r_i}$. From that moment, the construction is very similar to [GGH+13c]. To prove the security, we generalize the bilinear Diffie-Hellman Exponent Assumption (used to prove the security of the first broadcast encryption with constant size ciphertext [BGW05]) to the multi-linear setting.

Similar to [GGH+13c], our construction can be converted to multi-linear graded-encodings, recently instantiated by Garg, Gentry, and Halevi [GGH13a] and Jean-Sébastien Coron, Tancrède Lepoint and Mehdi Tibouchi [CLT13].

Theorem 5.1 (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.*

5.1 Our Construction

- $\mathsf{Params}(1^\lambda, d_{\max})$: The parameters generation algorithms takes the security parameter and the maximum circuit depth. It generate 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 output the public parameters $\mathsf{pp} = \big(\{G_i, g_i\}_{i\in[k]}, \{e_{ij}\}_{i,j\in[k]}\big)$, 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 $\mathsf{msk} := \alpha$.

- $\mathsf{Keygen}(C, \mathsf{msk})$: 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. Choose randomly $\big((r_1, z_1), \ldots, (r_\ell, z_\ell)\big) \in \mathbb{Z}_q$ 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 $\mathbb{Z}_q$ randomly for all internal wires of C.

2. Compute the matrix M:

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

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

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

Else if Γ is an AND gate, compute

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

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

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

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

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

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

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

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

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 $C(x)_u = 1$ 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 $C(x)_v = 1$ 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 $C(x) = 1$, 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 $m = 1$ if $\psi = \gamma_0$, otherwise output 0.

5.2 Correctness

We show the correctness in two steps: first, we show that for all active input wires $i \in X$, the user correctly computes $g_2^{r_i s}$.

$$e\bigg(g^s, g^{r_i}\big(\prod_{j \in X} h_j\big)^{z_i}\bigg) = \cdot e\bigg(\big(\prod_{j \in X} h_j\big)^s, g^{-z_i}\bigg) =$$
$$e\big(g, g\big)^{r_i s} \cdot \big(g, \prod_{j \in X} h_j\big)^{s z_i} \cdot \big(\prod_{j \in X} h_j, g\big)^{-s z_i} = g_2^{r_i s}$$

Claim 5.1.1. *For all active wires w at level j (that is, $C(x)_w = 1$) 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

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

Hence, if $C(x) = 1$, the user computes $g_k^{s r_n}$ 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 $m = 1$. □

5.3 Security Proof

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

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

where β is 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 wire w at depth j, $r_w = c_1^{\ell+1} \prod_{2 \le i \le j} c_i$ (plus some known randomness). Hence, for all input wires w, $r_w = c_1^{\ell+1}$ (plus some 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:

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

  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

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

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

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

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

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

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

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

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 $r_w = c_1^{\ell+1} \prod_{2 \leq i \leq j} c_i + e_w$, $a_w = c_j + d_w$ 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,

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

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

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

- $\mathsf{Enc}^*(\mathsf{mpk}, x^*, m)$: 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

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

where b is a randomly chosen bit. Note that $\left(\prod_{i \in X} h_i\right)^s$ 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,

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

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.

6 Applications

6.1 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 (See Section-A for definitions) 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 6.1 ([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 6.2. *Combining our short secret key ABE construction (Theorem-4.3) and Theorem-6.1, 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 secret key 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. Starting from any FHE scheme and a symmetric-key encryption where the size of the ciphertext is the size of the input plus $\mathrm{poly}(\lambda)$, this can be done as follows:

1. The new FHE encryption algorithm on input m of size k samples a fresh symmetric-key K of size $\mathrm{poly}(\lambda)$ and encrypts m using the symmetric-key encryption algorithm to get a ciphertext ct_0 of size $k + \mathrm{poly}(\lambda)$.
2. Next, it encrypts K using the standard FHE algorithm to get ct_1. Hence, the total size of the ciphertext is $k + \mathrm{poly}(\lambda)$.
3. The homomorphic evaluation algorithm can first run the symmetric decryption on $\mathsf{ct}_0, \mathsf{ct}_1$ to get a homomorphic encryption of m, and then continue with the homomorphic circuit evaluation.

Corollary 6.3. *Combining the above observation, our short ciphertext ABE construction (Theorem-5.1) and Theorem-6.1, 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 6.4 ([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 6.5. *From Corollary-6.2 and Theorem-6.4, 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 6.6. *From Corollary-6.3 and Theorem-6.4, 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 $\ell + \mathrm{poly}(d_{\max}, \lambda)$.*

6.2 Attribute-Based Fully Homomorphic Encryption (ABFHE)

An attribute-based fully homomorphic encryption (ABFHE) scheme has all of the functionality of ABE, but also allows messages encrypted under the same public index to be processed homomorphically by anyone (without any public or private keys), such that the final ciphertexts can still be decrypted by any party that was entitled to decrypt the original ciphertexts. Recently, Gentry, Sahai and Waters [GSW13] constructed the first ABFHE scheme, building on the ABE scheme of Gorbunov, Vaikuntanathan and Wee [GVW13]. By apply Gentry et al.'s techniques to our ABE scheme, we obtain an *ABFHE scheme with constant size secret keys*, with security based on LWE.

Recall that Gentry et al. actually provided a general compiler to transform *any* LWE-based ABE scheme with suitable properties into an LWE-based ABFHE scheme, the properties being:

1. **Property 1 (Ciphertext and decryption key vectors)**: Decryption keys $\vec{s}_C$ and ciphertexts $\vec{c}_x$ are vectors over $\mathbb{Z}_q$. The first coefficient of each $\vec{s}_C$ is 1.

2. **Property 2 (Small Dot Product)**: If $\vec{c}_x$ encrypts 0, then $\langle \vec{c}_x, \vec{s}_C \rangle$ is "small".

3. **Property 3 (Security)**: Encryptions of 0 are indistinguishable from uniform vectors over $\mathbb{Z}_q$ (under LWE).

As in [GVW13], decryption keys in our LWE-based ABE scheme do not natively have the form needed by the compiler. But, as with [GVW13], one can re-interpret decryption as a two-step process. In the first step, rather than decrypting "incrementally" gate-by-gate, we "holistically" view decryption as applying an overall linear transformation (in particular, a dot product) to the ciphertext vector: we thereby obtain a "sub-key" $\vec{s}_{C,x}$, a vector that depends on sk_C and x, that represents this overall linear transformation. In the second step, we simply determine whether the dot product $\langle \vec{c}_x, \vec{s}_{C,x} \rangle$ is small or not. Viewed in this way, our ABE scheme has all of the properties required by Gentry et al.'s compiler, and we obtain ABFHE with short secret keys.

6.3 Outsourcing Decryption

Since ABE decryption is often computationally expensive, the question arises: can we outsource it [GHW11]? Gorbunov et al. [GVW13] showed that decryption in their scheme can be outsourced easily: roughly, the user associated to circuit C gives the server its entire secret key sk_C, except for the secret key material associated to the final gate (used in the final step of decryption). In our scheme, we can do exactly the same thing, but the payoff is greater: since all of the recoding matrices in our secret keys are *public* except the last one, they do not need to be *transmitted* to or *stored* by the server; the server only needs the circuit C associated to each user.

References

[ABB10a] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In *EUROCRYPT*, pages 553–572, 2010.

[ABB10b] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (h)ibe in the standard model. In *EUROCRYPT*, pages 553–572, 2010.

[AFV11] Shweta Agrawal, David Mandell Freeman, and Vinod Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In *ASIACRYPT*, pages 21–40, 2011.

[AIKW13] Benny Applebaum, Yuval Ishai, Eyal Kushilevitz, and Brent Waters. Encoding functions with constant online rate or how to compress garbled circuits keys. In *CRYPTO (2)*, pages 166–184, 2013.

[Ajt99] Miklós Ajtai. Generating hard instances of the short basis problem. In *ICALP*, pages 1–9, 1999.

[AKS01] Miklós Ajtai, Ravi Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *STOC*, pages 601–610, 2001.

[ALdP11] Nuttapong Attrapadung, Benoît Libert, and Elie de Panafieu. Expressive key-policy attribute-based encryption with constant-size ciphertexts. In *Public Key Cryptography*, pages 90–108, 2011.

[App13] Benny Applebaum. Garbling xor gates "for free" in the standard model. In *TCC*, pages 162–181, 2013.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *ITCS*, pages 309–325, 2012.

[BGW05] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In Victor Shoup, editor, *CRYPTO*, volume 3621 of *Lecture Notes in Computer Science*, pages 258–275. Springer, 2005.

[BHHI10] Boaz Barak, Iftach Haitner, Dennis Hofheinz, and Yuval Ishai. Bounded key-dependent message security. In *EUROCRYPT*, pages 423–444, 2010.

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

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *TCC*, pages 253–273, 2011.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*, pages 97–106, 2011.

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. Cryptology ePrint Archive, Report 2013/352, 2013. http://eprint.iacr.org/.

[CHKP12] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. *J. Cryptology*, 25(4):601–639, 2012.

[CKKZ12] Seung Geol Choi, Jonathan Katz, Ranjit Kumaresan, and Hong-Sheng Zhou. On the security of the "free-xor" technique. In *TCC*, pages 39–53, 2012.

[CLT13] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In *CRYPTO (1)*, pages 476–493, 2013.

[FN93] Amos Fiat and Moni Naor. Broadcast encryption. In Douglas R. Stinson, editor, *CRYPTO*, volume 773 of *Lecture Notes in Computer Science*, pages 480–491. Springer, 1993.

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *EUROCRYPT*, pages 1–17, 2013.

[GGH+13b] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. *IACR Cryptology ePrint Archive*, 2013:451, 2013. To appear in FOCS 2013.

[GGH+13c] Sanjam Garg, Craig Gentry, Shai Halevi, Amit Sahai, and Brent Waters. Attribute-based encryption for circuits from multilinear maps. In *CRYPTO (2)*, pages 479–499, 2013.

[GGP10] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *CRYPTO*, pages 465–482, 2010.

[GHV10] Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. A simple BGN-type cryptosystem from LWE. In *EUROCRYPT*, pages 506–522, 2010.

[GHW11] Matthew Green, Susan Hohenberger, and Brent Waters. Outsourcing the decryption of abe ciphertexts. In *Proceedings of the 20th USENIX conference on Security*, SEC'11, pages 34–34, Berkeley, CA, USA, 2011. USENIX Association.

[GKP+13a] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. How to run turing machines on encrypted data. In *CRYPTO (2)*, pages 536–553, 2013.

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

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. In *STOC*, pages 113–122, 2008.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC*, pages 218–229, 1987.

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

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206, 2008.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *CRYPTO (1)*, volume 8042 of *Lecture Notes in Computer Science*, pages 75–92. Springer, 2013.

[GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In *CRYPTO*, pages 162–179, 2012.

[GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In *Proceedings of the 45th annual ACM symposium on Symposium on theory of computing*, STOC '13, pages 545–554, New York, NY, USA, 2013. ACM.

[KS08] Vladimir Kolesnikov and Thomas Schneider. Improved garbled circuit: Free xor gates and applications. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP (2)*, volume 5126 of *Lecture Notes in Computer Science*, pages 486–498. Springer, 2008.

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

[LO13] Steve Lu and Rafail Ostrovsky. How to garble ram programs. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT*, volume 7881 of *Lecture Notes in Computer Science*, pages 719–734. Springer, 2013.

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

[LW10] Allison B. Lewko and Brent Waters. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In *TCC*, pages 455–479, 2010.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT*, pages 700–718, 2012.

[MR07] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM J. Comput.*, 37(1):267–302, 2007.

[MV10] Daniele Micciancio and Panagiotis Voulgaris. A deterministic single exponential time algorithm for most lattice problems based on voronoi cell computations. In *STOC*, pages 351–358, 2010.

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

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem. In *STOC*, pages 333–342, 2009.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6), 2009.

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

[Wat09] Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In *CRYPTO*, pages 619–636, 2009.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *FOCS*, pages 162–167, 1986.

A Additional Preliminaries

A.1 Functional Encryption (FE)

We recall the functional encryption definition from the literature [KSW08, BSW11, GVW12] with some notational changes.

Definition A.1. *A functional encryption scheme $\mathcal{FE}$ for a class of boolean circuits with an n-bit input $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ is a tuple of four p.p.t. algorithms (*$\mathsf{FE.Setup}$*,* $\mathsf{FE.Keygen}$*,* $\mathsf{FE.Enc}$*,* $\mathsf{FE.Dec}$*) such that:*

- $\mathsf{FE.Setup}(1^\lambda)$ *takes as input the security parameter 1^λ and outputs a master public key* mpk *and a master secret key* msk*.*
- $\mathsf{FE.Keygen}(\mathsf{msk}, C)$ *takes as input the master secret key* msk *and a circuit $C \in \mathcal{C}$ and outputs a key sk_C.*
- $\mathsf{FE.Enc}(\mathsf{mpk}, x)$ *takes as input the master public key* mpk *and an input $x \in \{0,1\}^*$ and outputs a ciphertext c.*
- $\mathsf{FE.Dec}(\mathsf{sk}_C, c)$ *takes as input a key sk_C and a ciphertext c and outputs a value y.*

Correctness. *For any polynomial $n(\cdot)$, for every sufficiently large security parameter λ, for $n = n(\lambda)$, for all $C \in \mathcal{C}_n$, and all $x \in \{0,1\}^n$,*

$$\Pr[(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda); \mathsf{sk}_C \leftarrow \mathsf{FE.Keygen}(\mathsf{msk}, C); c \leftarrow \mathsf{FE.Enc}(\mathsf{mpk}, x) : \mathsf{FE.Dec}(\mathsf{sk}_C, c) = C(x)] = 1 - \mathrm{negl}(\lambda).$$

A.1.1 Security of Functional Encryption

Intuitively, the security of functional encryption requires that an adversary should not learn anything about the input x other than the computation result $C(x)$, for some circuit C for which a key was issued (the adversary can learn the circuit C). As mentioned, two notions of security have been used in previous work: full and selective security, with the same meaning as for ABE. We present both definitions because we achieve them with different parameters of the gapSVP assumption. Our definitions are simulation-based: the security definition states that whatever information an adversary is able to learn from the ciphertext and the function keys can be simulated given only the function keys and the output of the function on the inputs.

Definition A.2. *(*$\mathsf{FULL\text{-}SIM}$*-Security FE Security) Let $\mathcal{FE}$ be a functional encryption scheme for the family of circuits $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$. For every p.p.t. adversary $A = (A_1, A_2)$ and p.p.t. simulator S, consider the following two experiments:*

$\underline{\exp^{\mathsf{real}}_{\mathcal{FE},A}(1^\lambda)}$**:**	$\underline{\exp^{\mathsf{ideal}}_{\mathcal{FE},A,S}(1^\lambda)}$**:**
	1: $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$
	2: $(C, \mathsf{state}_A) \leftarrow A_1(\mathsf{mpk})$
	3: $\mathsf{sk}_C \leftarrow \mathsf{FE.Keygen}(\mathsf{msk}, C)$
	4: $(x, \mathsf{state}'_A) \leftarrow A_2(\mathsf{state}_A, \mathsf{sk}_C)$
5: $c \leftarrow \mathsf{FE.Enc}(\mathsf{mpk}, x)$	5: $\tilde{c} \leftarrow S(\mathsf{mpk}, \mathsf{sk}_C, C, C(x), 1^{\lvert x\rvert})$
6: Output (state'_A, c)	6: Output $(\mathsf{state}'_A, \tilde{c})$

The scheme is said to be (single-key) FULL-SIM*−secure if there exists a p.p.t. simulator S such that for all pairs of p.p.t. adversaries (A_1, A_2), the outcomes of the two experiments are computationally indistinguishable:*

$$\left\{ \exp^{\mathsf{real}}_{\mathcal{FE},A}(1^\lambda) \right\}_{\lambda\in\mathbb{N}} \overset{c}{\approx} \left\{ \exp^{\mathsf{ideal}}_{\mathcal{FE},A,S}(1^\lambda) \right\}_{\lambda\in\mathbb{N}}.$$

We now define selective security, which is a weakening of full security, by requiring the adversary to provide the challenge input x before seeing the public key or any other information besides the security parameter. We simply specify the difference from full security.

Definition A.3 (SEL-SIM-Security)**.** *The same as Def. A.2, but modify the game so that the first step consists of A specifying the challenge input x given only the security parameter.*

A.2 Reusable Garbled Circuits

We use the term reusable garbled circuit to refer to the most interesting variant of garbled circuits: the ones that can run on an arbitrary number of encoded inputs without compromising the privacy of the circuit or of the input. We recall the definition of a reusable garbled circuit presented in [GKP+13b].

Definition A.4. *A reusable garbling scheme for a family of circuits $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ with $\mathcal{C}_n$ a set of boolean circuits taking as input n bits, is a tuple of p.p.t. algorithms* $\mathsf{RGb} = (\mathsf{RGb.Garble}, \mathsf{RGb.Enc}, \mathsf{RGb.Eval})$ *such that*

- $\mathsf{RGb.Garble}(1^\lambda, C)$ *takes as input the security parameter λ and a circuit $C \in \mathcal{C}_n$ for some n, and outputs the garbled circuit Γ and a secret key* sk*.*
- $\mathsf{RGb.Enc}(\mathsf{sk}, x)$ *takes as input $x \in \{0,1\}^*$ and outputs an encoding c.*
- $\mathsf{RGb.Eval}(\Gamma, c)$ *takes as input a garbled circuit Γ, an encoding c and outputs a value y which should be $C(x)$.*

Correctness. *For any polynomial $n(\cdot)$, for all sufficiently large security parameters λ, for $n = n(\lambda)$, for all circuits $C \in \mathcal{C}_n$ and all $x \in \{0,1\}^n$,*

$$\Pr[(\Gamma, \mathsf{sk}) \leftarrow \mathsf{RGb.Garble}(1^\lambda, C); c \leftarrow \mathsf{RGb.Enc}(\mathsf{sk}, x); y \leftarrow \mathsf{RGb.Eval}(\Gamma, c) : C(x) = y] = 1 - \mathrm{negl}(\lambda).$$

Efficiency. *There exists a universal polynomial $p = p(\lambda, n)$ (p is the same for all classes of circuits $\mathcal{C}$) such that for all input sizes n, security parameters λ, for all boolean circuits C of with n bits of input, for all $x \in \{0,1\}^n$,*

$$\Pr[(\Gamma, \mathsf{sk}) \leftarrow \mathsf{RGb.Garble}(1^\lambda, C) : |\mathsf{sk}| \le p(\lambda, n) \text{ and } \mathit{runtime}(\mathsf{RGb.Enc}(\mathsf{sk}, x)) \le p(\lambda, n)] = 1.$$

Note that since $\mathsf{RGb.Enc}$ is a p.p.t. algorithm, it suffices to ensure that $|\mathsf{sk}| \le p(\lambda, n)$ and obtain that $\mathsf{RGb.Enc}$'s runtime is also at most a polynomial. We prefer to keep the runtime of $\mathsf{RGb.Enc}$ in the definition as well for clarity.

A.2.1 Security of Reusable Garbled Circuits

Here, we present the security definition of the reusable garbled circuits, as presented in [GKP+13b].

Definition A.5. *(Input and circuit privacy with reusability)*

Let RGb *be a garbling scheme for a family of circuits* $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$*. For a pair of p.p.t. algorithms* $A = (A_1, A_2)$ *and a p.p.t. simulator* $S = (S_1, S_2)$*, consider the following two experiments:*

$\underline{\exp^{\mathsf{real}}_{\mathsf{RGb},A}(1^\lambda)\textbf{:}}$

1: $(C, \mathsf{state}_A) \leftarrow A_1(1^\lambda)$
2: $(\mathsf{gsk}, \Gamma) \leftarrow \mathsf{RGb.Garble}(1^\lambda, C)$
3: $\alpha \leftarrow A_2^{\mathsf{RGb.Enc}(\mathsf{gsk},\cdot)}(C, \Gamma, \mathsf{state}_A)$
4: Output α

$\underline{\exp^{\mathsf{ideal}}_{\mathsf{RGb},A,S}(1^\lambda)\textbf{:}}$

1: $(C, \mathsf{state}_A) \leftarrow A_1(1^\lambda)$
2: $(\tilde{\Gamma}, \mathsf{state}_S) \leftarrow S_1(1^\lambda, 1^{|C|})$
3: $\alpha \leftarrow A_2^{\mathsf{O}(\cdot,C)[[\mathsf{state}_S]]}(C, \tilde{\Gamma}, \mathsf{state}_A)$
4: Output α

In the above, $\mathsf{O}(\cdot, C)[[\mathsf{state}_S]]$ *is an oracle that on input* x *from* A_2*, runs* S_2 *with inputs* $C(x)$*,* $1^{|x|}$*, and the latest state of* S*; it returns the output of* S_2 *(storing the new simulator state for the next invocation).*

We say that the garbling scheme RGb *is input- and circuit-private with reusability if there exists a p.p.t. simulator* S *such that for all pairs of p.p.t. adversaries* $A = (A_1, A_2)$*, the following two distributions are computationally indistinguishable:*

$$\left\{\exp^{\mathsf{real}}_{\mathsf{RGb},A}(1^\lambda)\right\}_{\lambda\in\mathbb{N}} \overset{c}{\approx} \left\{\exp^{\mathsf{ideal}}_{\mathsf{RGb},A,S}(1^\lambda)\right\}_{\lambda\in\mathbb{N}}.$$

We can see that this security definition enables reusability of the garbled circuit: A_2 is allowed to make as many queries for input encodings as it wants.

B Parameters Derivation for Short Secret Key ABE

This section provides a detailed description on the selection of parameters for our scheme, so that both correctness (see Section 4.1) and security (see Section 4.2) of our scheme are satisfied.

For a family of circuits $\mathcal{C}$ of bounded depth $d_{\max}$, with the LWE dimension n, the parameters can be chosen as follows:

- The error distribution χ is typically a truncated discrete Gaussian distribution $D_{\mathbb{Z},\sqrt{n}}$ with parameter $\sqrt{n}$. And, the error bound $B = O(\sqrt{n}\sqrt{n}) = O(n)$. For this B, the probability for a random sample from $D_{\mathbb{Z},\sqrt{n}}$ to be $\mathbf{0}$ would be $\mathrm{negl}(n)$, according to Theorem 3.1.

From now, we will consider the LWE modulus parameter $q = q(n, d_{\max})$, without instantiating it, to calculate the other parameters m, s, α. Later, we will instantiate q with a value which would make m, s, α satisfy the correctness and security properties.

- The parameter $m = O(n \log q)$.
- The "small" Gaussian parameter s is chosen to be $O(\sqrt{n \log q})$.

- Now, let us calculate the value of the "large" Gaussian parameter $\alpha = \alpha(n, d_{\max})$. We should choose α such that the output of the SampleLeft and the SampleRight algorithms are statistically indistinguishable from each other, when provided with the same set of inputs $\mathbf{F}$ and $\mathbf{u}$.

  The SampleRight algorithm (Algorithm 1) requires

  $$\alpha > \|\widetilde{\mathbf{T_B}}\| \cdot \|\mathbf{R}\| \cdot \omega(\sqrt{\log m}) \tag{3}$$

  Hence, we proceed as follows:

  1. First, we calculate the value of $\|\mathbf{R}\|$, where $\mathbf{R}$ is the matrix obtained corresponding to $\mathbf{A}_{\text{out}}$ in the simulated $\mathcal{ABE}$ algorithm KeyGen^*. In KeyGen^*, at each step of the induction, we get the matrix

     $$\mathbf{R}_w = \mathbf{R}_v \cdot \mathbf{R}_{\mathbf{B}} \cdot \mathsf{BD}(\mathbf{A}_u) - C(\mathbf{x}^*)_v \mathbf{R}_u$$

     corresponding to the matrix $\mathbf{A}_w$ of the outgoing wire w of the gate $g(u, v; w)$. We prove the following claim which would help us deduce the value of $\|\mathbf{R}\|_{\max}$.

     Claim B.0.1. *Suppose that for the gate $g(u, v; w)$, with the incoming wires u, v at depths j_0, j_1 respectively, $\|\mathbf{R}_u\|_{\max} \leq m^{2j_0}(2s\sqrt{m})^{j_0}$ and $\|\mathbf{R}_v\|_{\max} \leq m^{2j_1}(2s\sqrt{m})^{q_1}$. Then, for the outgoing wire w, the maximum norm of the matrix $\mathbf{R}_w$ would be $\|\mathbf{R}_w\|_{\max} \leq m^{2(\max(j_0,j_1)+1)}(2s\sqrt{m})^{\max(j_0,j_1)+1}$*

     Proof. This proof proceeds similar to the calculation of $\|\mathbf{e}_w\|_\infty$ in Claim-4.1.1. In particular,

     $$\begin{aligned}
     \|\mathbf{R}_w\|_{\max} &\leq m \cdot \|\mathbf{R}_{\mathbf{B}} \cdot \mathsf{BD}(\mathbf{A}_u)\|_{\max} \cdot \|\mathbf{R}_u\|_{\max} + \|\mathbf{R}_v\|_{\max} \\
     &\leq m\big((ms\sqrt{m}) \cdot m^{2j_0}(2s\sqrt{m})^{j_0} + m^{2j_1}(2s\sqrt{m})^{j_1}\big) \\
     &\leq \big(m^2 \cdot m^{2j_0} 2^{j_0} (s\sqrt{m})^{j_0+1} + m \cdot m^{2j_1}(2s\sqrt{m})^{j_1}\big) \\
     &\leq m^{2(\max(j_0,j_1)+1)}(2s\sqrt{m})^{\max(j_0,j_1)+1}
     \end{aligned}$$

     Thus, the maximum norm of the matrix $\mathbf{R}_w$ corresponding to $\mathbf{A}_w$ would be $\|\mathbf{R}_w\|_{\max} \leq m^{2(\max(j_0,j_1)+1)}(2s\sqrt{m})^{\max(j_0,j_1)+1}$. □

     Thus, each element of the matrix $\mathbf{R}$ corresponding to $\mathbf{A}_{\text{out}}$, has an absolute value of atmost $\|\mathbf{R}\|_{\max} \leq m^{2d_{\max}}(2s\sqrt{m})^{d_{\max}}$.
  2. We then get s_R as follows:

     $$s_R := \|\mathbf{R}\| := \sup_{\mathbf{x} \in S^{m-1}} \|\mathbf{R} \cdot \mathbf{x}\| \leq m \cdot \|\mathbf{R}\|_{\max} \leq m^{2d_{\max}+1}(2s\sqrt{m})^{d_{\max}}$$

  3. For a matrix $\mathbf{T_B}$ got from TrapSamp (Algorithm 3.2), $\|\widetilde{\mathbf{T_B}}\| = O(\sqrt{n \log q})$.
  4. Finally, we substitute these values in Equation 3 to get the value of α required for the SampleRight algorithm.

     $$\alpha \geq O(\sqrt{n \log q}) \cdot m^{2d_{\max}+1}(2s\sqrt{m})^{d_{\max}} \cdot \omega(\sqrt{\log m}) \tag{4}$$

  The value of the parameter α required for the SampleLeft algorithm (Algorithm 2) is

  $$\alpha \geq O(\sqrt{n \log q}) \cdot \omega(\sqrt{\log 2m}) \tag{5}$$

Thus, to satisfy both Equation 4 and Equation 5, we set the parameter

$$\alpha \geq O(\sqrt{n \log q}) \cdot m^{2d_{\max}+1}(2s\sqrt{m})^{d_{\max}} \cdot \omega(\sqrt{\log m})$$

Thus, the outputs of the SampleLeft and the SampleRight algorithms will be statistically indistinguishable from each other, when provided with the same set of inputs $\mathbf{F}$ and $\mathbf{u}$. Since we had assigned the "small" Gaussian parameter $s = O(\sqrt{n \log q})$, the parameter $\alpha \geq O(m^{3d_{\max}+2})$. Hiding the constants, we assign $\alpha = O(n \log q)^{O(d_{\max})}$.

When our scheme is instantiated with these parameters, the correctness (see Section 4.1) of the scheme is satisfied when

$$O(B \cdot O(n \log q)^{O(d_{\max})}) < q/4$$

Clearly, this condition is satisfied when $q = \tilde{O}(n \log d_{\max})^{O(d_{\max})}$. Also, this value of $q = \text{poly}(n)$, enables both the quantum reduction [Reg09] and the classical reduction [Pei09] from $\mathsf{dLWE}_{n,q,\chi}$ to approximating lattice problems in the worst case, when n, χ chosen as described above. To conclude this section, for a given max depth $d_{\max}$ and an LWE dimension $n = n(d_{\max})$, we set the parameters for our scheme to satisfy both the correctness and security, as follows:

$$\begin{aligned}
\chi &= D_{\mathbb{Z},\sqrt{n}} \\
B &= O(n) \\
q &= \tilde{O}(nd_{\max})^{O(d_{\max})} \\
m &= O(n \log q) \\
s &= O(n \log q) \\
\alpha &= O(n \log q)^{O(d_{\max})}
\end{aligned}$$