Attribute-Based Encryption for a Subclass of Circuits with Bounded Depth from Lattices

Xiang Xie[1], Rui Xue[2]

[1] Institute of Software, Chinese Academy of Sciences
[2] The State Key Laboratory of Information Security
Institute of Information Engineering, Chinese Academy of Sciences
`xiexiang@is.iscas.ac.cn, xuerui@iie.ac.cn`

Abstract. In this work, we present two Key-Policy Attribute-Based Encryption (ABE) schemes for some subclass of circuits based on the Learning with Error (LWE) assumption. Our constructions are selectively secure in the standard model. More specifically, our first construction supports a subclass of circuits with polynomially bounded depth. We call this subclass the OR-restricted circuits which means that for any input x, if $f(x) = 0$ then for all the OR gates in f, at least one of its incoming wires will evaluate to 0. The second one is a Key-Policy ABE scheme for shallow circuits whose depth is bounded by $O(\log \log \lambda)$, where λ is the security parameter.

Keywords: Functional Encryption, Attribute-Based Encryption, Lattices

1 Introduction

In traditional public key encryption the encrypted message can only be recovered by the person who has the corresponding secret key. However, More general ways of expressing who should be able to decrypt the message may be needed. Functional Encryption (FE) [9,32] is a new paradigm for public key encryption that enables fine-grained control of access to encrypted data. Generally speaking, in a functional encryption the key generation center allows the user to only reveal some specific function of encrypted data according to her secret key. One special and powerful case of FE is the notion of Attribute-Based Encryption (ABE) which is proposed by Sahai and Waters [36]. Two different variants of ABE are considered: Key-Policy ABE (KP-ABE) and Ciphertext-Policy ABE (CP-ABE) [23]. In a KP-ABE scheme, a ciphertext is associated with a set of attributes which can be denoted as an assignment x of boolean variables. A secret key is produced by an authority and is associated with a boolean function or circuit f. A user can decrypt the ciphertext if and only if $f(x) = 1$. Alternatively, in a CP-ABE scheme, a ciphertext is associated with a boolean function or circuit f, while a secret key is associated with an assignment x.

A plenty of constructions have been proposed since the introduction of ABE. These including efficient constructions e.g. [23,38], lattice-based constructions [11], new techniques to achieve adaptive security [26,30,27,28] and applications in outsourcing computation [33]. However, the class of allowable circuits in the above constructions are all restricted to Boolean formula whose fanout of every gate is one and is a subclass of NC1.

Very recently, Sahai and Waters [17] achieve ABE scheme for general circuits. Garg, Gentry, Sahai, and Waters [18] show a general primitive called witness encryption and turn it to ABE schemes for circuits along with witness indistinguishable proofs. Both work uses the multilinear map which is instantiated from the recent exciting work presented by [16]. A concurrent work with better results proposed by Gorbunov, Vaikuntanathan, and Wee [22] presents ABE schemes for all circuits (not restricted ones as ours) based on the Learning with Errors (LWE) assumption. The schemes in [17] and [22] have succinct ciphertext in that the ciphertext size depends on the depth of the circuits. Goldwasser et al. [21] show how to use succinct ABE for circuits and fully homomorphic encryption schemes (e.g. [19,14,13,12]) to get succinct function encryptions.

Our results. In this paper we show a way to construct ABE for some restricted circuits, which we call OR-restricted circuits. A circuit f is said to be OR-restricted, if for any input x such that $f(x) = 0$ then for all the OR gates in f, at least one of its incoming wires will evaluate to 0. Our scheme supports any OR-restricted circuits with polynomially bounded depth and arbitrary fanout (the fanout of the entire circuit is 1). The security of the construction is based on the LWE assumption (with subexponential module). Our method is a lattice-based analogy to the one in [17]. The ciphertexts of our KP-ABE scheme are associated with a k-tuple x of boolean variables and keys are associated with boolean circuits f of a max depth ℓ, where k and ℓ are polynomially bounded and previously determined at the stage of setup. The circuits in our construction are monotonic which means that it only contains AND and OR gates. As discussed in [17], any general circuit can be transformed into a monotonic one with negation only appearing at the input wires for the same function with the same depth and twice as the size. The ciphertext size only depends on the max depth of the circuit ℓ (not the size). We also present a possible solution to overcome the restricted circuits and give an ABE for circuits whose depth is very shallow,say, $O(\log\log\lambda)$, where λ is the security parameter.

We remark that our first result is *incomparable* with the ones for Boolean formulas. While our scheme supports restricted circuits with polynomially bounded depth, their constructions support Boolean formulas without restriction but only with depth $O(\log\lambda)$.

Difficulty and Our Techniques. The major difficulty (as discussed in [17]) to achieve circuit ABE is to prevent backtracking attacks. More specifically, consider a boolean circuit with fanout greater than 1. Suppose an OR gate C with two input wires which are the output wires of gates A and B. When decrypting, if C evaluates to 1, the user (or adversary) can compute some intermediate value E_C for C from the one E_A for A (if A evaluates to 1) or E_B for B (if B evaluates to 1). Now suppose that A evaluates to 1, B evaluates to 0. If the user or adversary can backtrack the value E_B from E_C (possibly,together with the secret key) then we call it a backtracking attack. The side effect of backtracking attack is that although B evaluates to 0, the user or adversary can still get the value E_B (which represents B evaluates to 1), and follow an undesired path to the output gate.

If the fanout of the circuits is one (e.g. Boolean formulas), then the backtracking attack does not affect the security. Because the output wire of gate B only feeds to C and has nowhere to go. Even the attacker backtracks the value E_B, it will eventually only compute E_C. However, suppose we want to consider circuit with fanout of two or more, and the output wire of B also feeds into an AND gate D. In this case, the backtracking attack would allow an adversary to act like B evaluates to 1 in the circuit even it does not. This misrepresentation can then be applied to other different paths due to the large fanout and harms the security of the scheme. This is also the major reason that the construction in [39] only works for Deterministic Finite Automata (DFA).

The technique in [17] to avoid backtracking is called "move and shift" by using multilinear map. Informally speaking, the secret key of a monotonic circuit f with depth ℓ works on a sequence of group $(\mathbb{G}_1, ..., \mathbb{G}_\ell)$. Each wire w in the j-th layer is associated with a random value r_w and a ciphertext is associated with a random value s. When decrypting, the algorithm works by computing $g_j^{sr_w}$ for each wire w in the circuit that evaluates to 1 on input x. The values are computed from the input wire, and get $g_j^{sr_w}$ from $g_{j-1}^{sr_{A(w)}}$ and $g_{j-1}^{sr_{A(w)}}$ by multilinear map according to the gate, where $A(w), B(w)$ are the incoming wires of w. The reason of this method to resist backtracking is that it's infeasible to compute $g_{j-1}^{sr_{A(w)}}$ or $g_{j-1}^{sr_{B(w)}}$ from $g_j^{sr_w}$ even with secret keys according to the assumptions of multilinear map.

In our construction, we use the Regev's [35] encryption to "move and shift". The decryption works by computing $\mathbf{h}_w = \mathbf{C}_w^t\mathbf{s} + \mathit{\Delta}$ for each wire w in the circuit that evaluates to 1 on the input, where $\mathbf{C}_w \in \mathbb{Z}_q^{n\times m}$ is a random matrix chosen by the authority when issuing the secret key, $\mathbf{s} \in \mathbb{Z}_q^n$ is a random vector in the ciphertext, and Δ is a vector with small norm.[3] If w evaluates to 0, the decryptor should not be able to obtain this value. The decryption works from the bottom up. We now focus on to OR gates to illustrate how we prevent backtracking attacks. Suppose wire w is the output of an OR gate with incoming wires $A(w)$, $B(w)$, and on a given input x the wire $A(w)$ evaluates to 1 and $B(w)$ evaluates to 0. Therefore, the decryptor obtains $\mathbf{h}_{A(w)} = \mathbf{C}^t_{A(w)}\mathbf{s} + \Delta$, but not $\mathbf{h}_{(B(w))}\mathbf{C}^t_{B(w)}\mathbf{s} + \Delta$. The private key associated with w is $(\mathbf{L}_{A(w)}, \mathbf{L}_{B(w)}) \in (\mathbb{Z}^{m\times m})^2$ such that the norm of these two matrices is small and

$$\mathbf{C}_{A(w)}\mathbf{L}_{A(w)} = \mathbf{C}_w \mod q \quad , \quad \mathbf{C}_{B(w)}\mathbf{L}_{B(w)} = \mathbf{C}_w \mod q.$$

To move decryption forward the algorithm computes

$$\mathbf{L}^t_{A(w)}\mathbf{h}_{A(w)} = \mathbf{L}^t_{A(w)}(\mathbf{C}^t_{A(w)}\mathbf{s} + \Delta) = \mathbf{C}^t_w\mathbf{s} + \Delta \mod q = \mathbf{h}_w$$

If the attacker wants to backtrack, recall that the attacker's goal would be to compute $\mathbf{h}_{B(w)} = \mathbf{C}^t_{B(w)}\mathbf{s} + \Delta$ even though $B(w)$ evaluates to false. By the Regev's encryption, it's infeasible to compute some small $\mathbf{L}' \in \mathbb{Z}^{m\times m}$, such that $\mathbf{C}_{B(w)} = \mathbf{C}_w\mathbf{L}'$ even given the secret key component $\mathbf{L}_{B(w)}$, without a trapdoor of $\mathbf{C}_w$. The security of our scheme lies in this intuition, we will give the formal proof in Section 3 which captures this intuition.

The AND gates have a similar mechanism, but require both $\mathbf{h}_{A(w)}$ and $\mathbf{h}_{B(w)}$ for decryption. The secret key component associated with an AND gate w is a small matrix $\mathbf{L}_w \in \mathbb{Z}^{2m\times m}$ such that $[\mathbf{C}_{A(w)}\|\mathbf{C}_{B(w)}]\mathbf{L}_w = \mathbf{C}_w \mod q$. If this process is applied iteratively to the output gate w^* then one can obtain $\mathbf{h}_{w^*}$. The final key component is used to obtain the message according to Regev's scheme.

1.1 Other Related Work

Other functionality in a similar vain to ABE includes Inner-Product Encryption (IPE) [25,31], Spatial Encryption [24] and regular language functionality [39]. Indeed, Waters [39] argues that backtracking attacks are the major obstacles to achieve Nondeterministic Finite Automata.

Identity-Based Encryption is a simpler functionality of ABE. Since the first construction [7] in the standard model, many different constructions from bilinear map are proposed, e.g. [10,8,37]. Besides, there are also many lattice-based constructions of IBE, HIBE [1,2,15], Fuzzy IBE [3] and IPE [4]. Very recently, Boyen [11] proposes a lattice-based ABE scheme by using the "basis-splicing" framework.

2 Preliminaries

For an integer m, we denote $[m]$ as an integer set $\{1, ..., m\}$. We use bold capital letters to denote matrices, and bold lowercase letters to denote vectors. The notation $\mathbf{A}^t$ denotes the transpose of the matrix $\mathbf{A}$. When we say a matrix defined over $\mathbb{Z}_q$ has full rank, we mean that it has full rank module q. 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 $\mathbf{A}_2$. If $\mathbf{x}_1$ is a vector of length m and $\mathbf{x}_2$ is of length m', then we let $[\mathbf{x}_1\|\mathbf{x}_2]$ denote the length $m + m'$ vector formed by concatenating $\mathbf{x}_1$ and $\mathbf{x}_2$. When doing matrix-vector multiplication, we always view vectors as column vectors.

[3] we use Δ denote small vectors in the following of the introduction.

A function $\text{negl}(\lambda)$ is *negligible*, if it vanishes faster than the inverse of any polynomial in λ. The *statistical distance* between two distributions X, Y over some finite or countable set S is defined as $\Delta(X,Y) = \frac{1}{2}\sum_{s \in S} \big| \Pr[X = s] - \Pr[Y = s] \big|$. X and Y are statistically indistinguishable if $\Delta(X,Y)$ is negligible.

2.1 Lattices

A full-rank m-dimensional integer lattice $\Lambda \subseteq \mathbb{Z}^m$ is a discrete additive subgroup whose linear span is $\mathbb{R}^m$. Every integer lattice is generated as the $\mathbb{Z}$-linear combination of some basis of linearly independent vectors $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_m\} \subset \mathbb{Z}^m$, i.e.,$\Lambda = \{\sum_{i=1}^{m} z_i \mathbf{b}_i : z_i \in \mathbb{Z}\}$. For a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, define the "$q$-ary" integer lattices $\Lambda^{\perp}(\mathbf{A}) = \{\mathbf{z} \in \mathbb{Z}^m : \mathbf{A}\mathbf{z} = 0 \mod q\}$.

Let Λ be a discrete subset of $\mathbb{Z}^m$. For any vector $\mathbf{c} \in \mathbb{R}^m$ and any positive parameter $\sigma \in \mathbb{R}_{>0}$, let $\rho_{\sigma,\mathbf{c}}(\mathbf{x}) = \exp(-\pi \|\mathbf{x} - \mathbf{c}\|^2/\sigma^2)$ be the Gaussian function on $\mathbb{R}^m$ with center $\mathbf{c}$ and parameter σ. Denote $\rho_{\sigma,\mathbf{c}}(\Lambda) = \sum_{\mathbf{x} \in \Lambda} \rho_{\sigma,\mathbf{c}}(\mathbf{x})$ be the discrete integral of $\rho_{\sigma,\mathbf{c}}$ over Λ, and $\mathcal{D}_{\Lambda,\sigma,\mathbf{c}}$ be the discrete Gaussian distribution over Λ with center $\mathbf{c}$ and parameter σ. Specifically, for all $\mathbf{y} \in \Lambda$, we have $\mathcal{D}_{\Lambda,\sigma,\mathbf{c}}(\mathbf{y}) = \frac{\rho_{\sigma,\mathbf{c}}(\mathbf{y})}{\rho_{\sigma,\mathbf{c}}(\Lambda)}$. For notional convenience, $\rho_{\sigma,\mathbf{0}}$ and $\mathcal{D}_{\Lambda,\sigma,\mathbf{0}}$ are abbreviated as ρ_σ and $\mathcal{D}_{\Lambda,\sigma}$, respectively.

2.2 The Learning with Errors Problem

We recall the learning with errors (LWE) problem, a classic hard problem on lattices defined by Regev [35].

Definition 1. *Let $n \geq 1$ and $q \geq 2$ be integers, let $\alpha \in (0,1)$. For $\mathbf{s} \in \mathbb{Z}_q^n$, let $A_{\mathbf{s},\alpha}$ be the distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$ obtained by choosing a vector $\mathbf{a} \in \mathbb{Z}_q^n$ uniformly at random, $e \leftarrow \mathcal{D}_{\mathbb{Z},\alpha q}$, and output $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e)$.*

The LWE problem is : for uniformly random $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, given a poly(n) number of samples that are either from $A_{\mathbf{s},\alpha}$ or uniformly random in $\mathbb{Z}_q^n \times \mathbb{Z}_q$, output 0 if the former holds and 1 if the latter holds.

It is known that when $\alpha q \geq 2\sqrt{n}$ and $q = \text{poly}(n)$, this decision problem is at least as hard as approximating several problems on n-dimensional lattices in the *worst-case* to within $\widetilde{O}(n/\alpha)$ factors with a quantum computer [35] or on a classical computer for a subset of these problems [34]. For subexponential q, i.e. $q = 2^{n^{\delta}}$ for any $0 < \delta < 1$, given the current state of the art algorithms on this problem, the running time is $2^{O(n^{1-\delta})}$. Therefore, the subexponential LWE says that the LWE is infeasible even in subexponential q.

In the following, we list some useful facts that make our constructions work.

Lemma 1 ([29] Lemma 2.11). *Let $\mathbf{x} \leftarrow \mathcal{D}_{\mathbb{Z}^m,r}$ with $r > 0$, then with overwhelming probability, $\|x\| \leq r\sqrt{m}$.*

Lemma 2 ([5,29]). *Let q, n, m be positive integers with $q \geq 2$ and $m \geq 6n \log q$. There is a probabilistic polynomial-time algorithm* $\texttt{TrapGen}(q,n,m)$ *that outputs a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ which is statistically close to uniform in $\mathbb{Z}_q^{n \times m}$ and a trapdoor $\mathbf{T}$ with $\|\mathbf{T}\| \leq O(n \log q)$. We note that $\mathbf{T}$ is either a short basis of $\Lambda^{\perp}(\mathbf{A})$ or a short matrix satisfying some relation with $\mathbf{A}$. Generally, we call $\mathbf{T}$ a trapdoor of $\mathbf{A}$*

Lemma 3 ([1,20]). *Let $q > 2, m > n$ and $m', n' \geq 0$. Let $\mathbf{T}$ be a trapdoor of $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and $\sigma \geq \|\mathbf{T}\| \cdot \omega(\sqrt{\log m})$. There exists an efficient randomized algorithm* $\texttt{Sample}$ *that, for any $\mathbf{B} \in$*

$\mathbb{Z}_q^{n\times m'}$ *and* $\mathbf{C} \in \mathbb{Z}_q^{n\times n'}$*, takes as inputs* $\mathbf{A}, \mathbf{B}, \mathbf{T}, \mathbf{C}, \sigma$*, and outputs* $\mathbf{S} \leftarrow \mathtt{Sample}(\mathbf{A}\|\mathbf{B}, \mathbf{T}, \mathbf{C}, \sigma)$*, where* $\mathbf{S} \in \mathbb{Z}^{(m+m')\times n'}$ *and the distribution of each* $\mathbf{s}_i$ *is statistically close to* $\mathbf{e}_i + \mathcal{D}_{\Lambda^{\perp}(\mathbf{U}),\sigma,-\mathbf{e}_i}$ *with* $\|\mathbf{s}_i\| \leq O(\sigma\cdot\sqrt{m+m'})$*. Where* $\mathbf{U} = [\mathbf{A}\|\mathbf{B}]$*,* $\mathbf{S} = (\mathbf{s}_1, ..., \mathbf{s}_{n'})$*,* $\mathbf{C} = (\mathbf{c}_1, ..., \mathbf{c}_{n'})$ *and* $\mathbf{E} = (\mathbf{e}_1, ..., \mathbf{e}_{n'})$ *be arbitrary solution to* $\mathbf{U}\mathbf{e}_i = \mathbf{c}_i \mod q$*. We note that the order of* $\mathbf{A}, \mathbf{B}$ *does not matter, i.e. the algorithm still works for* $\mathbf{U} = [\mathbf{B}\|\mathbf{A}]$*.*

Lemma 4 ([20]). *Let* n, q *be positive integers and* q *be prime, let* $m \geq n\log q$*. Then for uniformly random* $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$*, and any* $\sigma \geq \omega(\sqrt{\log m})$*, the distribution of* $\mathbf{u} = \mathbf{A}\mathbf{e} \mod q$ *is statistically close to uniform in* $\mathbb{Z}_q$*, where* $\mathbf{e} \leftarrow \mathcal{D}_{\mathbb{Z}^m,\sigma}$*.*

In particular, fix $\mathbf{u} \in \mathbb{Z}_q^n$*, and let* $\mathbf{t} \in \mathbb{Z}^m$ *be arbitrary solution to* $\mathbf{A}\mathbf{t} = \mathbf{u} \mod q$*, then the conditional distribution of* $\mathbf{e} \leftarrow \mathcal{D}_{\mathbb{Z}^m,\sigma}$ *given* $\mathbf{A}\mathbf{e} = \mathbf{u} \mod q$ *is exactly* $\mathbf{t} + \mathcal{D}_{\Lambda^{\perp}(\mathbf{A}),\sigma,-\mathbf{t}}$*.*

2.3 Circuits

As discussed in [17], given a Boolean circuit C, one can use De Morgan's rule to construct a monotonic circuit $\tilde{C}$ with negation only appearing in the input and computes the same function of C. The depth of $\tilde{C}$ is the same as C and the size is no more than twice the size of C (for more details, we refer to [17]). Therefore, we only consider the monotonic circuits, i.e., only has AND and OR gates. Note that inputs to the circuit correspond to boolean variables x_i and $x_i = 1$ captures having the i-th attribute.

We define the notation for circuits by adapting the model and notation in [6]. We first restrict our circuits to be boolean circuits, then our circuits will have a single output gate. Finally we only consider monotonic circuits where gates are either AND or OR gates with two inputs. We remark that the output wires of the intermediate gates will feed to more than one gate.

We denote a circuit to be a five tuple $f = (k, r, A, B, \mathtt{GateType})$. Here $k \geq 2$ is the number of inputs and q is the number of the gates. Denote $\mathsf{Inputs}=\{1, ..., k\}$, $\mathsf{Wires}=\{1, ..., k+r\}$, and $\mathsf{Gates}=\{k+1, ..., k+r\}$. The wire $k+r$ is the designated output wire. Then $A : \mathsf{Gates} \rightarrow \mathsf{Wires}$ is a function where $A(w)$ identifies w's first incoming wire. $B : \mathsf{Gates} \rightarrow \mathsf{Wires}$ is a function where $B(w)$ identifies w's second incoming wire. Finally, $\mathtt{GateType} : \mathsf{Gates} \rightarrow \{\text{AND, OR}\}$ is a function that identifies a gate is either an AND or an OR gate. We require that $w > B(w) > A(w)$. In addition, we let $f(x)$ be the evaluation of the circuit f on input $x \in \{0,1\}^k$, and let $f_w(x)$ be the value of wire w of the circuit on input x.

A circuit f is said to be OR-restricted on x, if for every OR gate w in f it will have $f_{A(w)}(x) = 0$ or $f_{B(w)(x)} = 0$. In other words, f will not evaluate both to 1 on the two incoming wires of an OR gate with input x. Denote $\mathcal{C}_\ell$ be the set of all the circuits whose depth are bounded by ℓ. We further defined a subset of $\mathcal{C}_\ell$ called OR-restricted circuits family

$$\text{OR-}\mathcal{C}_\ell = \{f \in \mathcal{C}_\ell : \forall x, f(x) = 0 \Rightarrow f \text{ is OR-restricted on } x\}.$$

Our first construction supports OR-$\mathcal{C}_\ell$ circuits for previously settled polynomial ℓ before setup. The other construction supports $\mathcal{C}_\ell$ circuits with $\ell = O(\log\log\lambda)$, where λ is the security parameter.

2.4 Definition of ABE for Circuits

We now describe the formal definition of Key-Policy Attribute-Based Encryption for circuits in $\mathcal{C}_\ell$.

Setup$(1^\lambda, k, \mathcal{C}_\ell)$ This algorithm takes as input the security parameter λ, then length of the input k and a bound ℓ on the circuit depth. It outputs the public parameters PP and a master secret key MSK.

Enc$(PP, x \in \{0,1\}^k, M \in \{0,1\})$ This algorithm takes as input the public parameters PP, a bit string $x \in \{0,1\}^k$, and a message M. It outputs a ciphertext CT.

KeyGen$(MSK, f = (k, r, A, B, \texttt{GateType}) \in \mathcal{C}_\ell)$ This algorithm takes as input the master secret key MSK and a description of a circuit $f \in \mathcal{C}_\ell$. It outputs a private key SK_f.

Dec(SK_f, CT) This algorithm takes as input a secret key SK_f and a ciphertext CT. It outputs M or a special symbol $\perp$.

Correctness For all $(PP, MSK) \leftarrow$ **Setup**$(1^\lambda, k, \mathcal{C}_\ell)$, for all message M, string $x \in \{0,1\}^k$ and $f \in \mathcal{C}_\ell$. For all $CT \leftarrow$ **Enc**(PP, x, M), and $SK_f \leftarrow$ **KeyGen**(MSK, f), if $f(x) = 1$ then **Dec**$(SK_f, CT) = M$; otherwise, **Dec**$(SK_f, CT) = \perp$.

Security Model of Attribute-Based Encryption for Circuits We now describe the selective security model of ABE for circuits. An attacker first announces an input string x^* he wants to challenge, after that the attacker will be able to query for multiple keys but not ones that evaluate to true on input x^*. In the end, the adversary needs to distinguish two ciphertexts whose messages are chosen by himself. We review the selective security game (e.g. [23]) as follows.

Init The adversary announces a challenge string x^*.

Setup The challenger runs the setup algorithm and gives the public parameters PP to the adversary and keeps MSK secret.

Phase 1 The adversary makes any polynomial number of private key queries for circuit description $f \in \mathcal{C}_\ell$ of its choice. The challenger returns **KeyGen**(MSK, f). For any circuit f requested, it should satisfy that $f(x^*) = 0$.

Challenge The adversary submits two equal length message M_0 and M_1. Then the challenger flips a random bit $b \in \{0,1\}$, and computes $CT^* \leftarrow$ **Enc**(PP, x^*, M_b). The challenge ciphertext CT^* is given to the adversary.

Phase 2 The same as Phase 1.

Guess The adversary outputs a guess b' of b.

The advantage of an adversary $\mathcal{A}$ in this game is defined as $\Pr[b' = b] - 1/2$.

Definition 2. *An Attribute-Based Encryption scheme for circuits in $\mathcal{C}_\ell$ is selectively secure if all polynomial time adversaries have negligible advantage.*

3 Construction

In this section, we describe our constructions. The first one is a Key-Policy ABE (KP-ABE) scheme for all circuits in OR-$\mathcal{C}_\ell$. We also give another construction for circuits in $\mathcal{C}_\ell$. However, the length is very shallow, say, $\ell = O(\log \log \lambda)$.

3.1 ABE for OR-restricted Circuits

We now give the construction. Our main construction is a Key-Policy ABE scheme. The scheme is a "public index" variety of functional encryption, where only the message M is hidden while x can be efficiently discovered from the ciphertext. The ciphertext size only depends on the circuit depth ℓ, not the size of the circuit.

Setup$(1^\lambda, k, \mathcal{C}_\ell)$ The setup algorithm takes as input a security parameter λ, the maximum depth ℓ of circuits, and the number of boolean inputs k. It first generates integers $q = q(\lambda, \ell), n = n(\lambda, \ell), m = m(\lambda, \ell)$ and error rate $\alpha = \alpha(\lambda, \ell)$ and sampling parameter $\sigma = \sigma(\lambda, \ell)$. Then it generates k uniformly random matrices from $\mathbb{Z}_q^{n\times m}$ together with trapdoors $(\mathbf{A}_i, \mathbf{T}_i) \leftarrow$ `TrapGen`(q, n, m) for $i \in [k]$, chooses an uniformly random vector $\mathbf{u} \leftarrow \mathbb{Z}_q^n$. Output

$$PP = (\{\mathbf{A}_i\}_{i\in[k]}, \mathbf{u}, n, m, q, \alpha, \sigma) \quad , \quad MSK = (\{\mathbf{T}_i\}_{i\in[k]}).$$

Enc$(PP, x \in \{0,1\}^k, M \in \{0,1\})$ This algorithm takes as input the public parameters PP, an input $x \in \{0,1\}^k$ and a message $M \in \{0,1\}$. Define S to be the set of i such that $x_i = 1$. It chooses a uniformly random vector $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, and samples Gaussian error vectors $\mathbf{x}_i \leftarrow \mathcal{D}_{\mathbb{Z}^m,\alpha q}$ for $i \in S$, a Gaussian error scale $x \leftarrow \mathcal{D}_{\mathbb{Z},\alpha q}$. It computes, for $i \in S$

$$\mathbf{c}_i = \mathbf{A}_i^t \mathbf{s} + \mathbf{x}_i \mod q \quad ; \quad c = \mathbf{u}^t \mathbf{s} + x + M \cdot \lfloor q/2 \rfloor \mod q.$$

Output ciphertext $CT = (x, \{\mathbf{c}_i\}_{i\in S}, c)$.

KeyGen$(MSK, f = (k, r, A, B, \texttt{GateType}) \in \mathcal{C}_\ell)$ The algorithm takes as input the master secret key MSK and a description f of a circuit. Recall that the circuit has $k+r$ wires with k input wires, r gates and the wire $k+r$ is designated as the output wire. The algorithm chooses random matrices together with trapdoors $(\mathbf{C}_i, \mathbf{Y}_i) \leftarrow$ `TrapGen`(q, n, m) for $i \in [k+r]$, where we think of random matrix $\mathbf{C}_w$ as being associated with wire w. The algorithm first generates a "header" component

$$K_H = \mathbf{e}_{k+r} \in \mathbb{Z}^m,$$

where $\mathbf{e}_{k+r} \leftarrow$ `Sample`$(\mathbf{C}_{k+r}, \mathbf{Y}_{k+r}, \mathbf{u}, \sigma)$. By Lemma 3, we know that $\|\mathbf{e}_{k+r}\|$ is small and $\mathbf{C}_{k+r}\mathbf{e}_{k+r} = \mathbf{u} \mod q$.
Next, the algorithm generates key components for every wire $w < k + r$. The structure of the key components depends upon if w is an Input wire, an OR gate, or an AND gate. We describe how it generates for each case.

- Input Wire: If $w \in [k]$, then it corresponds to the w-th input. The key generation algorithm samples $\mathbf{L}_w \leftarrow$ `Sample`$(\mathbf{A}_w, \mathbf{T}_w, \mathbf{C}_w, \sigma)$ such that $\|\mathbf{L}_w\|$ is small and $\mathbf{A}_w\mathbf{L}_w = \mathbf{C}_w \mod q$. We note this is possible since the algorithm knows the master secret key $\mathbf{T}_w$. Output

$$K_w = \mathbf{L}_w \in \mathbb{Z}^{m\times m}.$$

- OR Gate: Suppose that wire w such that `GateType`(w) =OR. The key generation algorithm generates $\mathbf{L}_{A(w)} \leftarrow$ `Sample`$(\mathbf{C}_{A(w)}, \mathbf{Y}_{A(w)}, \mathbf{C}_w, \sigma)$ and $\mathbf{L}_{B(w)} \leftarrow$ `Sample`$(\mathbf{C}_{B(w)}, \mathbf{Y}_{B(w)}, \mathbf{C}_w, \sigma)$. Again, from Lemma 3 we have $\|\mathbf{L}_{A(w)}\|, \|\mathbf{L}_{B(w)}\|$ are small, and $\mathbf{C}_{A(w)}\mathbf{L}_{A(w)} = \mathbf{C}_w \mod q$, $\mathbf{C}_{B(w)}\mathbf{L}_{B(w)} = \mathbf{C}_w \mod q$. Output

$$K_w = (\mathbf{L}_{A(w)}, \mathbf{L}_{B(w)}) \in \mathbb{Z}^{m\times m} \times \mathbb{Z}^{m\times m}.$$

– AND Gate: Suppose that wire w such that $\texttt{GateType}(w)$ =AND. The key generation algorithm generates $\mathbf{L}_w \leftarrow \texttt{Sample}(\mathbf{C}_{A(w)} \| \mathbf{C}_{B(w)}, \mathbf{Y}_{A(w)}, \mathbf{C}_w, \sigma)$. We note that the algorithm can also use $\mathbf{Y}_{B(w)}$ to sample $\mathbf{L}_w$. By Lemma 3, $\|\mathbf{L}_w\|$ is small and $[\mathbf{C}_{A(w)} \| \mathbf{C}_{B(w)}]\mathbf{L}_w = \mathbf{C}_w \mod q$. Output

$$K_w = \mathbf{L}_w \in \mathbb{Z}^{2m \times m}.$$

Finally, the algorithm outputs the secret key $SK_f = (f, K_H, \{K_w\}_{1 \leq w < k+r})$.

Dec(SK_f, CT) Suppose that we are evaluating decryption for a secret key associated with a circuit $f = (k, r, A, B, \texttt{GateType})$ and a ciphertext $CT = (x, \{\mathbf{c}_i\}_{i \in S}, c)$. We will be able to decrypt if $f(x) = 1$. We will evaluate the circuit from the bottom up. Consider wire w; if $f_w(x) = 1$ then, the algorithm will compute $\mathbf{h}_w = \mathbf{C}_w^t \mathbf{s} + \hat{\mathbf{x}}_w$ where $\hat{\mathbf{x}}_w$ is some small error vector. If $f_w(x) = 0$ no one can get the value $\mathbf{h}_w$. The algorithm proceeds iteratively starting with computing $\mathbf{h}_1$ and proceeds in order to finally computes $\mathbf{h}_{k+r}$. We breaking the cases according to whether the wire is an Input, AND or OR gate.

– Input Wire: If $w \in [k]$, then it corresponds to the w-th input. Suppose that $f_w(x) = 1$. The algorithm computes

$$\mathbf{h}_w = \mathbf{L}_w^t \mathbf{c}_w = \mathbf{L}_w^t (\mathbf{A}_w^t \mathbf{s} + \mathbf{x}_w) = \mathbf{C}_w^t \mathbf{s} + \hat{\mathbf{x}}_w \mod q,$$

where $\hat{\mathbf{x}}_w = \mathbf{L}_w^t \mathbf{x}_w$. We note that if $f_w(x) = 1$, i.e., $x_w = 1$, we always get $\mathbf{c}_w$ from the ciphertext, otherwise if $x_w = 0$, no one can compute $\mathbf{h}_w$ because now $\mathbf{c}_w$ is not included in the ciphertext.

– OR Gate: Consider a wire w such that $\texttt{GateType}(w)$ =OR. Suppose $f_w(x) = 1$. Then, If $f_{A(w)}(x) = 1$, the algorithm computes:

$$\mathbf{h}_w = \mathbf{L}_{A(w)}^t \mathbf{h}_{A(w)} = \mathbf{L}_{A(w)}^t (\mathbf{C}_{A(w)}^t \mathbf{s} + \hat{\mathbf{x}}_{A(w)}) = \mathbf{C}_w^t \mathbf{s} + \hat{\mathbf{x}}_w,$$

where $\hat{\mathbf{x}}_w = \mathbf{L}_{A(w)}^t \hat{\mathbf{x}}_{A(w)}$. Note that now we have $\mathbf{h}_{A(w)}$ from the iteration. Alternatively, if $f_{B(w)}(x) = 1$, it computes:

$$\mathbf{h}_w = \mathbf{L}_{B(w)}^t \mathbf{h}_{B(w)} = \mathbf{L}_{B(w)}^t (\mathbf{C}_{B(w)}^t \mathbf{s} + \hat{\mathbf{x}}_{B(w)}) = \mathbf{C}_w^t \mathbf{s} + \hat{\mathbf{x}}_w,$$

where $\hat{\mathbf{x}}_w = \mathbf{L}_{B(w)}^t \hat{\mathbf{x}}_{B(w)}$.

– AND Gate: Consider a wire w such that $\texttt{GateType}(w)$ =AND. Suppose $f_w(x) = 1$, then $f_{A(w)}(x) = f_{B(x)}(x) = 1$. It computes:

$$\mathbf{h}_w = \mathbf{L}_w^t [\mathbf{h}_{A(w)} \| \mathbf{h}_{B(w)}] = \mathbf{L}_w^t ([\mathbf{C}_{A(w)} \| \mathbf{C}_{B(w)}]^t \mathbf{s} + [\hat{\mathbf{x}}_{A(w)} \| \hat{\mathbf{x}}_{B(w)}]) = \mathbf{C}_w^t \mathbf{s} + \hat{\mathbf{x}}_w,$$

where $\hat{\mathbf{x}}_w = \mathbf{L}_w^t [\hat{\mathbf{x}}_{A(w)} \| \hat{\mathbf{x}}_{B(w)}]$.

If $f(x) = f_{k+r}(x) = 1$, then the algorithm will compute $\mathbf{h}_{k+r} = \mathbf{C}_{k+r}^t \mathbf{s} + \hat{\mathbf{x}}_{k+r}$. It finally computes $v = c - \mathbf{e}_{k+r}^t \mathbf{h}_{k+r} \mod q$, if $|v| \leq q/4$, then returns 0; otherwise, returns 1.

3.2 Correctness and Parameters

To analyze the correctness and the parameters. We first estimate the norm of the error terms in the shifting step. Before that, we view the circuits as layered ones. Denote B_s for $0 \le s \le \ell$ to be the norm of the error vectors in $\mathbf{h}_w$ when w is in the s-th layer. Since for any $\mathbf{c}_i = \mathbf{A}_i^t \mathbf{s} + \mathbf{x}_i$, we have $\|\mathbf{x}_i\| \le \alpha q \sqrt{m}$, then $B_0 \le \alpha q \sqrt{m}$. Denote R be the upper bound of the norm of the matrix generated from the `sample` algorithm. Lemma 3 implies $R \le O(\sigma \cdot m)$. The shifting step in the decryption algorithm shows that $B_s \le 2\sqrt{m} \cdot R \cdot B_{s-1} \le O(\sigma \cdot m^{3/2}) \cdot B_{s-1}$. Therefore, the norm of the error vector B_ℓ in $\mathbf{h}_{k+r}$ satisfies $B_\ell \le 2^{O(\log(\sigma \cdot m)\cdot \ell)} \cdot B_0$. Furthermore,

$$v = c - \mathbf{e}_{k+r}^t \mathbf{h}_{k+r} = \mathbf{u}^t \mathbf{s} + x + M \cdot \lfloor q/2 \rfloor - \mathbf{e}_{k+r}^t (\mathbf{C}_{k+r}^t \mathbf{s} + \hat{\mathbf{x}}_{k+r}) = x - \mathbf{e}_{k+r}^t \hat{\mathbf{x}}_{k+r} + M \cdot \lfloor q/2 \rfloor \mod q.$$

A simple analysis shows that if $|x - \mathbf{e}_{k+r}^t \hat{\mathbf{x}}_{k+r}| \le q/4$, the decryption algorithm will return the right message. Therefore, we only need to set the parameters to satisfy the inequality. Since $|x| \le \alpha q \sqrt{m}$, $\|\mathbf{e}_{k+r}\| \le R$ and $\|\hat{\mathbf{x}}_{k+r}\| \le B_\ell$. Hence $|x - \mathbf{e}_{k+r}^t \hat{\mathbf{x}}_{k+r}| \le \alpha q \sqrt{m} + \sqrt{m} \cdot R \cdot B_\ell \le \alpha q \cdot 2^{O(\log(\sigma \cdot m)\cdot \ell)}$. A feasible selection of parameters is that given ℓ, set $\lambda = \ell^{1/\delta}$ for some constant $0 < \delta < 1/2$. Let $q = 2^{\ell^2} = 2^{\lambda^{2\delta}}$ to be prime, $n = \lambda$, $m = 2n \log q = 2\lambda^{1+2\delta}$, $\sigma = n \log q \cdot \sqrt{2m} = 2\lambda^{1.5+3\delta}$, $\alpha = n/q = \lambda / 2^{\lambda^{2\delta}}$. It's easy to check that for sufficient small δ, it will hold that $|x - \mathbf{e}_{k+r}^t \hat{\mathbf{x}}_{k+r}| \le q/4$.

Note that, the module q here is set to be subexponential, then the security of our scheme will be based on the subexponential LWE assumption.

3.3 Security

We now prove selective security of our scheme in the standard model. The security is based on the subexponential LWE assumption.

Theorem 1. *Our scheme is selectively secure for circuits in OR-$\mathcal{C}_\ell$, if the subexponential LWE assumption holds.*

Proof. We show that if there exists a polynomial time adversary $\mathcal{A}$ on our ABE scheme for circuit in OR-$\mathcal{C}_\ell$ with non-negligible advantage, then we can construct a polynomial time algorithm $\mathcal{B}$ on the subexponential LWE assumption with non-negligible advantage. We describe how $\mathcal{B}$ interacts with $\mathcal{A}$.

Init $\mathcal{A}$ first declares challenge input $x^* \in \{0,1\}^k$.

Setup $\mathcal{B}$ sets the public parameters according to x^*. If $x_i^* = 1$, then $\mathcal{B}$ receives $(\mathbf{A}_i, \mathbf{b}_i)$ from the LWE oracle; if $x_i^* = 0$, it invokes `TrapGen`(q,n,m) to generate matrix $\mathbf{A}_i$ together with a trapdoor $\mathbf{T}_i$. It receives another pair $(\mathbf{u}, w)$ from the LWE oracle and generates parameters $\alpha = \alpha(\lambda, \ell)$ and $\sigma = \sigma(\lambda, \ell)$. $\mathcal{B}$ sends $(\{\mathbf{A}_i\}_{i\in[k]}, \mathbf{u}, n, m, q, \alpha, \sigma\ )$to $\mathcal{A}$. According to `TrapGen` algorithm, we know that the public parameters are statistically indistinguishable from the ones in the real game.

Challenge $\mathcal{A}$ submits two messages M_0, M_1. Let $S^* \subseteq [n]$ be the set of input indices where $x^* = 1$. $\mathcal{B}$ flips a random bit $b \in \{0,1\}$ and creates challenge ciphertext as:

$$CT^* = (x^*, \{\mathbf{b}_i\}_{i \in S^*}, w + M_b \cdot \lfloor q/2 \rfloor).$$

If $\mathbf{b}_i$ and w are LWE samples, then the ciphertext is the encryption of M_b; otherwise, the ciphertext is uniformly random and independent of M_b.

Key Generation Phase Both key generation phases are in the same manner by the reduction algorithm. To answer queries, we can think of the proof as having some invariant properties on each wire of the circuit. For any $f \in$ OR-$\mathcal{C}_\ell$, consider a wire w, if $f_w(x^*) = 1$ then the intermediate matrix $\mathbf{C}_w$ is totally random, and $\mathcal{B}$ does not know the corresponding trapdoor; if $f_w(x^*) = 0$ then $\mathbf{C}_w$ is generated from the `TrapGen` algorithm and $\mathcal{B}$ knows the trapdoor of $\mathbf{C}_w$, i.e., $\mathbf{Y}_w$. This enables $\mathcal{B}$ to sample $K_H = \mathbf{e}_{k+r}$, since $f_{k+r}(x^*) = 0$. We describe how to generate key for each wire w.

- Input Wire: Suppose $w \in [k]$, if $f_w(x^*) = 1$, then sample $\mathbf{L}_w \leftarrow (\mathcal{D}_{\mathbb{Z}^m,\sigma})^m$, and set $\mathbf{C}_w = \mathbf{A}_w\mathbf{L}_w \mod q$. Applying Lemma 4, $\mathbf{C}_w$ is statistically close to uniform random in $\mathbb{Z}_q^{n\times m}$. By Lemma 3 and 4, the conditional distribution of $\mathbf{L}_w \leftarrow (\mathcal{D}_{\mathbb{Z}^m,\sigma})^m$ given $\mathbf{A}_w\mathbf{L}_w = \mathbf{C}_w \mod q$ is statistically close to the one in the real game. Note that $\mathcal{B}$ does not know the trapdoor of $\mathbf{C}_w$. $\mathcal{B}$ sets $K_w = \mathbf{L}_w$.
  If $f_w(x^*) = 0$, $\mathcal{B}$ first generates $(\mathbf{C}_w, \mathbf{Y}_w) \leftarrow \texttt{TrapGen}(q, n, m)$, and then invoke the `Sample` algorithm to generate $\mathbf{L}_w \leftarrow \texttt{Sample}(\mathbf{A}_w, \mathbf{T}_w, \mathbf{C}_w, \sigma)$. Note that in the **Setup** phase, $\mathcal{B}$ knows the trapdoor $\mathbf{T}_w$ if $x_w^* = 0$. In this way, the generated key component is exactly the same as the one in the real game and $\mathcal{B}$ has the trapdoor of $\mathbf{C}_w$. $\mathcal{B}$ sets $K_w = \mathbf{L}_w$.

- OR Gate: Suppose wire w such that $\texttt{GateType}(w)$ =OR. If $f_w(x^*) = 1$, since $f \in$ OR-$\mathcal{C}_\ell$, by the definition of OR-$\mathcal{C}_\ell$ circuits and $f(x^*) = 0$ we know that f is OR-restricted in x^*. Therefore, either $f_{A(w)}(x^*) = 0$ or $f_{B(w)}(x^*) = 0$. Without loss of generality, assuming $f_{A(w)}(x^*) = 0$, which means $\mathcal{B}$ knows $\mathbf{Y}_{A(w)}$ the trapdoors of $\mathbf{C}_{A(w)}$. Now $\mathcal{B}$ samples $\mathbf{L}_{B(w)} \leftarrow (\mathcal{D}_{\mathbb{Z}^m,\sigma})^m$, and sets $\mathbf{C}_w = \mathbf{C}_{B(w)}\mathbf{L}_{B(w)} \mod q$. From Lemma 4, $\mathbf{C}_w$ is statistically close to uniformly random in $\mathbb{Z}_q^{n\times m}$. Again by Lemma 3 and 4, the conditional distribution of $\mathbf{L}_{B(w)} \leftarrow (\mathcal{D}_{\mathbb{Z}^m,\sigma})^m$, given $\mathbf{C}_{B(w)}\mathbf{L}_{B(w)} = \mathbf{C}_w \mod q$ is statistically close to the one in the real game. Finally, $\mathcal{B}$ uses the trapdoor $\mathbf{Y}_{A(w)}$ and `Sample` algorithm to get $\mathbf{L}_{A(w)} \leftarrow \texttt{Sample}(\mathbf{C}_{A(w)}, \mathbf{Y}_{A(w)}, \mathbf{C}_w, \sigma)$. $\mathcal{B}$ sets $K_w = (\mathbf{L}_{A(w)}, \mathbf{L}_{B(w)})$.
  If $f_w(x^*) = 0$, then $f_{A(w)}(x^*) = f_{B(w)}(x^*) = 0$. $\mathcal{B}$ knows $\mathbf{Y}_{A(w)}$ and $\mathbf{Y}_{B(w)}$ the trapdoors of $\mathbf{C}_{A(w)}$ and $\mathbf{C}_{B(w)}$ respectively. $\mathcal{B}$ first generates $(\mathbf{C}_w, \mathbf{Y}_w) \leftarrow \texttt{TrapGen}(q, n, m)$, then uses the trapdoors $\mathbf{Y}_{A(w)}, \mathbf{Y}_{B(w)}$ to sample $\mathbf{L}_{A(w)}$ and $\mathbf{L}_{B(w)}$ just as in the real game. $\mathcal{B}$ sets $K_w = (\mathbf{L}_{A(w)}, \mathbf{L}_{B(w)})$.
  We remark that when $f_{A(w)}(x^*) = f_{B(w)}(x^*) = 1$, then $\mathcal{B}$ does not know how to answer queries. Since $\mathcal{B}$ does not know any of the trapdoors of $\mathbf{C}_{A(w)}$ and $\mathbf{C}_{B(w)}$, $\mathcal{B}$ then can not sample small matrices $\mathbf{L}_1, \mathbf{L}_2$ such that $\mathbf{C}_{A(w)}\mathbf{L}_1 = \mathbf{C}_{B(w)}\mathbf{L}_2$. This is the major reason we have to restrict to OR-$\mathcal{C}_\ell$ circuits.

- AND Gate: Consider a wire w such that $\texttt{GateType}(w)$ =AND. If $f_w(x^*) = 1$, then $f_{A(w)}(x^*) = f_{B(w)}(x^*) = 1$. $\mathcal{B}$ samples $\mathbf{L}_w \leftarrow (\mathcal{D}_{\mathbb{Z}^{2m},\sigma})^m$ and set $\mathbf{C}_w = [\mathbf{C}_{A(w)}\|\mathbf{C}_{B(w)}]\mathbf{L}_w \mod q$. By Lemma 3 and 4, $\mathbf{C}_w$ is statistically close to uniform and the conditional distribution of $\mathbf{L}_w$ given $[\mathbf{C}_{A(w)}\|\mathbf{C}_{B(w)}]\mathbf{L}_w = \mathbf{C}_w \mod q$ is statistically close to the one in the real game. $\mathcal{B}$ sets $K_w = \mathbf{L}_w$.
  If $f_w(x^*) = 0$, without loss of generality, we assume $f_{A(w)}(x^*) = 0$. $\mathcal{B}$ knows $\mathbf{Y}_{A(w)}$ the trapdoor of $\mathbf{C}_{A(w)}$, $\mathcal{B}$ first generates $(\mathbf{C}_w, \mathbf{Y}_w) \leftarrow \texttt{TrapGen}(q, n, m)$, and invokes the `Sample` algorithm to get $\mathbf{L}_w \leftarrow \texttt{Sample}(\mathbf{C}_{A(w)}\|\mathbf{C}_{B(w)}, \mathbf{Y}_{A(w)}, \mathbf{C}_w, \sigma)$. $\mathcal{B}$ sets $K_w = \mathbf{L}_w$.

Finally, since $f_{k+r}(x^*) = 0$, $\mathcal{B}$ knows $\mathbf{Y}_{k+r}$ the trapdoor of $\mathbf{C}_{k+r}$, and $\mathcal{B}$ samples $\mathbf{e}_{k+r} \leftarrow \texttt{Sampe}(\mathbf{C}_{k+r}, \mathbf{Y}_{k+r}, \mathbf{u}, \sigma)$. $\mathcal{B}$ sets $K_H = \mathbf{e}_{k+r}$ and returns $SK_f = (f, K_H, \{K_w\}_{1\le w<k+r})$ to $\mathcal{A}$.

Guess $\mathcal{B}$ receives back the guess b' from $\mathcal{A}$. If $b = b'$, $\mathcal{B}$ guesses that the samples are LWE samples; otherwise, it guesses that they are random samples.

This immediately shows that any adversary with non-negligible advantage in the selective security game will have an identical advantage in breaking the subexponential LWE assumption. $\square$

3.4 ABE for Shallow Circuits

In order to overcome the restriction on the circuits, we propose a partial solution of the problem in the OR-gates when simulating. The main idea is as follows. In the simulation of the above proof, the only case that $\mathcal{B}$ can not handle is that when $\mathcal{B}$ does not know any trapdoor of $\mathbf{C}_{A(w)}$ and $\mathbf{C}_{B(w)}$, $\mathcal{B}$ cannot sample small matrix $\mathbf{L}_{A(w)}$ and $\mathbf{L}_{B(w)}$ such that $\mathbf{C}_{A(w)}\mathbf{L}_{A(w)} = \mathbf{C}_{B(w)}\mathbf{L}_{B(w)}$. A possible way is that instead of one matrix, each wire now is associated with a matrix set. In an OR gate, when shifting $\mathbf{h}_{A(w)}$ or $\mathbf{h}_{B(w)}$ to $\mathbf{h}_w$, the matrices in wire w are not the same. That is in this OR-gate w, the shifting part use the fact $\mathbf{C}_{A(w)}\mathbf{L}_{A(w)} = \mathbf{C}_w^1$ and $\mathbf{C}_{B(w)}\mathbf{L}_{B(w)} = \mathbf{C}_w^2$, where $\mathbf{C}_w^1$ and $\mathbf{C}_w^2$ are in the matrix set. In this way, we can avoid the case $\mathcal{B}$ can not handle the secret key. However, the drawback now is the size of the matrix set increases very fast with the depth, which restrict the construction to only support shallow circuits, say, with depth $O(\log\log\lambda)$. We now describe the construction, the **Setup** and **Enc** algorithms are exactly the same as in the above scheme.

Setup$(1^\lambda, k, \mathcal{C}_\ell)$ The setup algorithm takes as input a security parameter λ, the maximum depth ℓ of circuits, and the number of boolean inputs k. It first generates integers $q = q(\lambda, \ell), n = n(\lambda, \ell), m = m(\lambda, \ell)$ and error rate $\alpha = \alpha(\lambda, \ell)$ and sampling parameter $\sigma = \sigma(\lambda, \ell)$. Then it generates k uniformly random matrices from $\mathbb{Z}_q^{n\times m}$ together with trapdoors $(\mathbf{A}_i, \mathbf{T}_i) \leftarrow$ `TrapGen`(q, n, m) for $i \in [k]$, chooses an uniformly random vector $\mathbf{u} \leftarrow \mathbb{Z}_q^n$. Output

$$PP = (\{\mathbf{A}_i\}_{i\in[k]}, \mathbf{u}, n, m, q, \alpha, \sigma) \quad , \quad MSK = (\{\mathbf{T}_i\}_{i\in[k]}).$$

Enc$(PP, x \in \{0,1\}^k, M \in \{0,1\})$ This algorithm takes as input the public parameters PP, an input $x \in \{0,1\}^k$ and a message $M \in \{0,1\}$. Define S to be the set of i such that $x_i = 1$. It chooses a uniformly random vector $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, and samples Gaussian error vectors $\mathbf{x}_i \leftarrow \mathcal{D}_{\mathbb{Z}^m,\alpha q}$ for $i \in S$, a Gaussian error scale $x \leftarrow \mathcal{D}_{\mathbb{Z},\alpha q}$. It computes, for $i \in S$

$$\mathbf{c}_i = \mathbf{A}_i^t\mathbf{s} + \mathbf{x}_i \mod q \quad ; \quad c = \mathbf{u}^t\mathbf{s} + x + M \cdot \lfloor q/2 \rfloor \mod q.$$

Output ciphertext $CT = (x, \{\mathbf{c}_i\}_{i\in S}, c)$.

KeyGen$(MSK, f = (k, r, A, B, \texttt{GateType}))$ The algorithm takes as input the master secret key MSK and a description f of a circuit. Recall that the circuit has $k+r$ wires with k input wires, r gates and the wire $k+r$ is designated as the output wire. For each wire w, we define a matrix set $\mathcal{MS}_w$, where the size of $\mathcal{MS}_w$ is defined in a recursive way according to w. If $w \in [k]$, i.e., w is the input wire, then $|\mathcal{MS}_w| = 1$; If `GateType`(w) = OR, then $|\mathcal{MS}_w| = |\mathcal{MS}_{A(w)}| + |\mathcal{MS}_{B(w)}|$; If `GateType`$(w)$ = AND, then $|\mathcal{MS}_w| = |\mathcal{MS}_{A(w)}| \cdot |\mathcal{MS}_{B(w)}|$. We note that the size of all $\mathcal{MS}_w$ is polynomial bounded if $\ell = O(\log\log\lambda)$.
The algorithm chooses random matrices together with trapdoors for each matrix set $\mathcal{MS}_w$. I.e. For each wire w, it generates $(\mathbf{C}_w^i, \mathbf{Y}_w^i) \leftarrow$ `TrapGen`(q, n, m) for $i \in [|\mathcal{MS}_w|]$. And let $\mathcal{MS}_w = \{\mathbf{C}_w^1, ..., \mathbf{C}_w^{|\mathcal{MS}_w|}\}$. The algorithm first generates a "header" component

$$K_H = \{\mathbf{e}_1, ..., \mathbf{e}_{|\mathcal{MS}_{k+r}|}\},$$

where $\mathbf{e}_i \leftarrow \texttt{Sample}(\mathbf{C}^i_{k+r}, \mathbf{Y}^i_{k+r}, \mathbf{u}, \sigma)$ for $i \in [|\mathcal{MS}_{k+r}|]$. By Lemma 3, $\mathbf{e}_i \in \mathbb{Z}^{m\times m}$ are small and $\mathbf{C}^i_{k+r}\mathbf{e}_i = \mathbf{u} \mod q$.
Next, the algorithm generates key components for every wire $w < k+r$. The structure of the key components depends upon if w is an Input wire, an OR gate, or an AND gate. We describe how it generates for each case.

- Input Wire: If $w \in [k]$, then it corresponds to the w-th input. Since now $|\mathcal{MS}_w| = 1$, we know that $\mathcal{MS}_w = \{\mathbf{C}^1_w\}$. The key generation algorithm samples $\mathbf{L}^1_w \leftarrow \texttt{Sample}(\mathbf{A}_w, \mathbf{T}_w, \mathbf{C}^1_w, \sigma)$ such that $\|\mathbf{L}^1_w\|$ is small and $\mathbf{A}_w\mathbf{L}^1_w = \mathbf{C}^1_w$. Output
$$K_w = \mathbf{L}^1_w.$$
- OR Gate: Suppose that wire w such that $\texttt{GateType}(w)$ =OR. For convenience, denote $|\mathcal{MS}_{A(w)}| = t_A$, $|\mathcal{MS}_{B(w)}| = t_B$. Then for each matrix $\mathbf{C}^i_{A(w)} \in \mathcal{MS}_{A(w)}$, it generates $\mathbf{L}^i_{A(w)} \leftarrow \texttt{Sample}(\mathbf{C}^i_{A(w)}, \mathbf{Y}^i_{A(w)}, \mathbf{C}^i_w, \sigma)$ and $\mathbf{L}^j_{B(w)} \leftarrow \texttt{Sample}(\mathbf{C}^j_{B(w)}, \mathbf{Y}^i_{B(w)}, \mathbf{C}^{j+t_A}_w, \sigma)$, for $i \in [t_A]$, $j \in [t_B]$. From Lemma 3, $\|\mathbf{L}^i_{A(w)}\|, \|\mathbf{L}^j_{B(w)}\|$ are small, $\mathbf{C}^i_{A(w)}\mathbf{L}^i_{A(w)} = \mathbf{C}^i_w \mod q$ and $\mathbf{C}^j_{B(w)}\mathbf{L}^j_{B(w)} = \mathbf{C}^{j+t_A}_w \mod q$. Output
$$K_w = (\{\mathbf{L}^i_{A(w)}\}_{i\in[t_A]}, \{\mathbf{L}^j_{B(w)}\}_{j\in[t_B]}).$$
- AND Gate: Suppose that wire w such that $\texttt{GateType}(w)$ =AND. It generates $\mathbf{L}^{i,j}_w \leftarrow \texttt{Sample}(\mathbf{C}^i_{A(w)}\|\mathbf{C}^j_{B(w)}, \mathbf{Y}^i_{A(w)}, \mathbf{C}^{(j-1)\cdot t_A+i}_w, \sigma)$, for $i \in [t_A], j \in [t_B]$. By Lemma 3, $\|\mathbf{L}^{i,j}_w\|$ are small, and $[\mathbf{C}^i_{A(w)}\|\mathbf{C}^j_{B(w)}]\mathbf{L}^{i,j}_w = \mathbf{C}^{(j-1)\cdot t_A+i}_w \mod q$. Output
$$K_w = \{\mathbf{L}^{i,j}_w; i \in [t_A], j \in [t_B]\}.$$

Finally, the algorithm outputs the secret key $SK_f = (f, K_H, \{K_w\}_{1\le w<k+r})$.

Dec(SK, CT) Suppose that we are evaluating decryption for a secret key associated with a circuit $f = (k, r, A, B, \texttt{GateType})$ and a ciphertext $CT = (x, \{\mathbf{c}_i\}_{i\in S}, c)$. We will be able to decrypt if $f(x) = 1$. We will evaluate the circuit from the bottom up. Consider wire w; if $f_w(x) = 1$ then, our algorithm will compute $\mathbf{h}_w = (\mathbf{C}^i_w)^t\mathbf{s} + \hat{\mathbf{x}}_w$ for some $\mathbf{C}^i_w \in \mathcal{MS}_w$, where $\hat{\mathbf{x}}_w$ is some small error vector. If $f_w(x) = 0$ nothing needs to be computed for that wire. Our algorithm proceeds iteratively starting with computing $\mathbf{h}_1$ and proceeds in order to finally compute $\mathbf{h}_{k+r}$. We breaking the cases according to whether the wire is an Input, AND or OR gate.

- Input Wire: If $w \in [k]$, then it corresponds to the w-th input. Suppose that $f_w(x) = 1$. The algorithm computes
$$\mathbf{h}_w = (\mathbf{L}^1_w)^t\mathbf{c}_w = (\mathbf{L}^1_w)^t(\mathbf{A}^t_w\mathbf{s} + \mathbf{x}_w) = (\mathbf{C}^1_w)^t\mathbf{s} + \hat{\mathbf{x}}_w,$$
where $\hat{\mathbf{x}}_w = (\mathbf{L}^1_w)^t\mathbf{x}_w$. We note that if $f_w(x) = 1$, i.e., $x_w = 1$, we always get $\mathbf{c}_w$ from the ciphertext, otherwise if $x_w = 0$, no one can compute $\mathbf{h}_w$ because now $\mathbf{c}_w$ is not included in the ciphertext.

– OR Gate: Consider a wire w such that $\texttt{GateType}(w)$ =OR. Suppose $f_w(x) = 1$. Then, If $f_{A(w)}(x) = 1$, there exists $\mathbf{C}^i_{A(w)} \in \mathcal{MS}_{A(w)}$ such that $\mathbf{h}_{A(w)} = (\mathbf{C}^i_{A(w)})^t\mathbf{s} + \hat{\mathbf{x}}_{A(w)}$ and the algorithm knows i and the value $\mathbf{h}_{A(w)}$. It computes:

$$\mathbf{h}_w = (\mathbf{L}^i_{A(w)})^t\mathbf{h}_{A(w)} = (\mathbf{L}^i_{A(w)})^t((\mathbf{C}^i_{A(w)})^t\mathbf{s} + \hat{\mathbf{x}}_{A(w)}) = (\mathbf{C}^i_w)^t\mathbf{s} + \hat{\mathbf{x}}_w,$$

where $\hat{\mathbf{x}}_w = (\mathbf{L}^i_{A(w)})^t\hat{\mathbf{x}}_{A(w)}$. Similar operations can be done if $f_{B(w)}(x) = 1$.

– AND Gate: Consider a wire w such that $\texttt{GateType}(w)$ =AND. Suppose $f_w(x) = 1$, then $f_{A(w)}(x) = f_{B(x)}(x) = 1$. Then there exists $\mathbf{C}^i_{A(w)}, \mathbf{C}^j_{B(w)}$ such that $\mathbf{h}_{A(w)} = (\mathbf{C}^i_{A(w)})^t\mathbf{s} + \hat{\mathbf{x}}_{A(w)}$ and $\mathbf{h}_{B(w)} = (\mathbf{C}^j_{B(w)})^t\mathbf{s} + \hat{\mathbf{x}}_{B(w)}$. It computes:

$$\mathbf{h}_w = (\mathbf{L}^{i,j}_w)^t[\mathbf{h}_{A(w)}\|\mathbf{h}_{B(w)}] = (\mathbf{L}^{i,j}_w)^t([\mathbf{C}^i_{A(w)}\|\mathbf{C}^j_{B(w)}]^t\mathbf{s}+[\hat{\mathbf{x}}_{A(w)}\|\hat{\mathbf{x}}_{B(w)}]) = (\mathbf{C}^{(j-1)\cdot t_A+i}_w)^t\mathbf{s}+\hat{\mathbf{x}}_w,$$

where $\hat{\mathbf{x}}_w = (\mathbf{L}^{i,j}_w)^t[\hat{\mathbf{x}}_{A(w)}\|\hat{\mathbf{x}}_{B(w)}]$.

If $f(x) = f_{k+r}(x) = 1$, then the algorithm will get $\mathbf{h}_{k+r} = (\mathbf{C}^i_{k+r})^t\mathbf{s} + \hat{\mathbf{x}}_{k+r}$ for some $i \in [|\mathcal{MS}_{k+r}|]$. It finally computes $v = c - \mathbf{e}^t_i\mathbf{h}_{k+r} \mod q$, if $|v| \leq q/4$, then returns 0; otherwise, returns 1.

3.5 Correctness and Parameters

The correctness and selection of parameters are very similar to the scheme in section 3.2. We now consider the size of the matrix set $\mathcal{MS}_w$. The size increases faster in an AND gate than in an OR gate (although it increases fast in an OR gate as well). For an AND gate w, we have that $|\mathcal{MS}_w| = |\mathcal{MS}_{A(w)}| \cdot |\mathcal{MS}_{B(w)}|$. As the depth increasing, the size of $\mathcal{MS}_{k+r}$ will be $O(2^{2^\ell})$, this restrict the scheme only support circuits with depth $O(\log\log\lambda)$. Since $\ell = O(\log\log\lambda)$, the module q can be set to be polynomial and the security of this construction will be based on the standard LWE assumption.

3.6 Security

We now prove selective security of our scheme in the standard model. The security is based on the LWE assumption.

Theorem 2. *Our scheme is selectively secure for circuits in $\mathcal{C}_\ell$ with $\ell = O(\log\log\lambda)$, if the LWE assumption holds.*

Proof. We show that if there exists a polynomial time $\mathcal{A}$ on our ABE scheme for circuits of depth ℓ and of length k inputs with non-negligible advantage, then we can construct a polynomial time algorithm $\mathcal{B}$ on the LWE assumption with non-negligible advantage. We describe how $\mathcal{B}$ interacts with $\mathcal{A}$.

Init $\mathcal{A}$ first declares challenge input $x^* \in \{0,1\}^k$.

Setup $\mathcal{B}$ sets the public parameters according to x^*. If $x^*_i = 1$, then $\mathcal{B}$ receives $(\mathbf{A}_i, \mathbf{b}_i)$ from the LWE oracle; if $x^*_i = 0$, it invokes $\texttt{TrapGen}(q, n, m)$ to generate matrix $\mathbf{A}_i$ together with a trapdoor $\mathbf{T}_i$. It receives another pair $(\mathbf{u}, w)$ from the LWE oracle and generates parameters $\alpha = \alpha(\lambda, \ell)$ and $\sigma = \sigma(\lambda, \ell)$. $\mathcal{B}$ sends $(\{\mathbf{A}_i\}_{i\in[k]}, \mathbf{u}, n, m, q, \alpha, \sigma$)to $\mathcal{A}$. According to $\texttt{TrapGen}$ algorithm, we know that the public parameters are statistically indistinguishable from the ones in the real game.

Challenge $\mathcal{A}$ submits two messages M_0, M_1. Let $S^* \subseteq [n]$ be the set of input indices where $x^* = 1$. $\mathcal{B}$ flips a random bit $b \in \{0,1\}$ and creates challenge ciphertext as:

$$CT^* = (x^*, \{\mathbf{b}_i\}_{i \in S^*}, w + M_b \cdot \lfloor q/2 \rfloor).$$

If $\mathbf{b}_i$ and w are LWE samples, then the ciphertext is the encryption of M_b; otherwise, the ciphertext is uniformly random and independent of M_b.

Key Generation Phase Both key generation phases are in the same manner by the reduction algorithm. To answer queries, we can think of the proof as having some invariant properties on each wire of the circuit. Consider a wire w, if $f_w(x^*) = 1$ then the intermediate matrices in $\mathcal{MS}_w$ is totally random, and $\mathcal{B}$ does not know *all* the trapdoors; if $f_w(x^*) = 0$ then $\mathcal{B}$ knows *all* the trapdoors of the matrices in $\mathcal{MS}_w$, i.e., $\mathbf{Y}_w^i$. This enables us to sample $\{\mathbf{e}_i\}_{i \in [|\mathcal{MS}_{k+r}|]}$, since $f_{k+r}(x^*) = 0$. We describe how to generate key for each wire w.

- Input Wire: Suppose $w \in [k]$, if $f_w(x^*) = 1$, then sample $\mathbf{L}_w^1 \leftarrow (\mathcal{D}_{\mathbb{Z}^m,\sigma})^m$, and set $\mathbf{C}_w^1 = \mathbf{A}_w \mathbf{L}_w^1 \mod q$. By Lemma 4, $\mathbf{C}_w^1$ is statistically close to uniformly random in $\mathbb{Z}_q^{n \times m}$. From Lemma 4 and 3 the conditional distribution of $\mathbf{L}_w^1$ given $\mathbf{A}_w^1 \mathbf{L}_w^1 = \mathbf{C}_w^1 \mod q$ is statistically close to the one in the real game. $\mathcal{B}$ sets $K_w = \mathbf{L}_w^1$.
  If $f_w(x^*) = 0$, $\mathcal{B}$ first generates $(\mathbf{C}_w^1, \mathbf{Y}_w^1) \leftarrow \mathtt{TrapGen}(q, n, m)$, and then invokes the $\mathtt{Sample}$ algorithm to get $\mathbf{L}_w^1 \leftarrow \mathtt{Sample}(\mathbf{A}_w, \mathbf{T}_w, \mathbf{C}_w^1, \sigma)$. Note that in the **Setup** phase, $\mathcal{B}$ knows the trapdoor $\mathbf{T}_w$ if $x_w^* = 0$. $\mathcal{B}$ sets $K_w = \mathbf{L}_w^1$.

- OR Gate: Suppose wire w such that $\mathtt{GateType}(w)$ =OR. If $f_w(x^*) = 1$, at least one of the sets $\mathcal{MS}_{A(w)}$, $\mathcal{MS}_{B(w)}$ $\mathcal{B}$ does not know *all* the trapdoors of the matrices. If $\mathcal{B}$ knows the trapdoor of some matrix in $\mathcal{MS}_{A(w)}$ or $\mathcal{MS}_{B(w)}$, it generates the key component exactly as in the real game. For each matrix $\mathbf{C}_{A(w)}^i$, $\mathcal{B}$ does not know the trapdoor, it samples $\mathbf{L}_{A(w)}^i \leftarrow (\mathcal{D}_{\mathbb{Z}^m,\sigma})^m$, and sets $\mathbf{C}_w^i = \mathbf{C}_{A(w)}^i \mathbf{L}_{A(w)}^i \mod q$. Similarly, for each matrix $\mathbf{C}_{B(w)}^j$, $\mathcal{B}$ does not know the trapdoor, it samples $\mathbf{L}_{B(w)}^j \leftarrow (\mathcal{D}_{\mathbb{Z}^m,s})^m$, and sets $\mathbf{C}_w^{j+t_A} = \mathbf{C}_{B(w)}^j \mathbf{L}_{B(w)}^j \mod q$. By the same analysis as before, the distribution of $\mathbf{L}_{A(w)}^i$, $\mathbf{L}_{B(w)}^j$, $\mathbf{C}_w^i$ and $\mathbf{C}_w^{j+t_A}$ are statistically close to the real game. $\mathcal{B}$ sets $K_w = (\{\mathbf{L}_{A(w)}^i\}_{i \in [t_A]}, \{\mathbf{L}_{B(w)}^j\}_{j \in [t_B]})$.
  If $f_w(x^*) = 0$, then $f_{A(w)}(x^*) = f_{B(w)}(x^*) = 0$. $\mathcal{B}$ knows all the trapdoors of matrices in $\mathcal{MS}_{A(w)}$ and $\mathcal{MS}_{B(w)}$. For every $i \in [t_A]$, $j \in [t_B]$, $\mathcal{B}$ first generates $(\mathbf{C}_w^i, \mathbf{Y}_w^i) \leftarrow \mathtt{TrapGen}(q, n, m)$, and $(\mathbf{C}_w^{j+t_A}, \mathbf{Y}_w^{j+t_A}) \leftarrow \mathtt{TrapGen}(q, n, m)$. Then it uses the trapdoors to sample $\mathbf{L}_{A(w)}^i \leftarrow \mathtt{Sample}(\mathbf{C}_{A(w)}^i, \mathbf{Y}_{A(w)}^i, \mathbf{C}_w^i, \sigma)$ and $\mathbf{L}_{B(w)}^j \leftarrow \mathtt{Sample}(\mathbf{C}_{B(w)}^j, \mathbf{Y}_{B(w)}^j, \mathbf{C}_w^{j+t_A}, \sigma)$. $\mathcal{B}$ sets $K_w = (\{\mathbf{L}_i^{A(w)}\}_{i \in [t_A]}, \{\mathbf{L}_{i \in [t_B]}^{B(w)}\})$.

- AND Gate: Consider a wire w such that $\mathtt{GateType}(w)$ =AND. If $f_w(x^*) = 1$, then $f_{A(w)}(x^*) = f_{B(w)}(x^*) = 1$. Therefore, $\mathcal{B}$ does not know all the trapdoors of the matrices in $\mathcal{MS}_{A(w)}$, $\mathcal{MS}_{B(w)}$. For any $i \in [t_A]$, $j \in [t_B]$, it samples $\mathbf{L}_w^{i,j} \leftarrow (\mathcal{D}_{\mathbb{Z}^{2m},s})^m$ and set $\mathbf{C}_w^{(j-1)\cdot t_A + i} = [\mathbf{C}_{A(w)}^i \| \mathbf{C}_{B(w)}^j] \mathbf{L}_w^{i,j} \mod q$. By a similar analysis, the distributions of the $\mathbf{L}_w^{i,j}$ and $\mathbf{C}_w^{(j-1)\cdot t_A + i}$ are statistically close to the real game. $\mathcal{B}$ sets $K_w = \{\mathbf{L}_w^{i,j}; i \in [t_A], j \in [t_B]\}$.
  If $f_w(x^*) = 0$, w.l.o.g, we assume $f_{A(w)}(x^*) = 0$, then $\mathcal{B}$ knows all trapdoors of the matrices in $\mathcal{MS}_{A(w)}$. $\mathcal{B}$ first generates $(\mathbf{C}_w^{(j-1)\cdot t_A + i}, \mathbf{Y}_w^{(j-1)\cdot t_A + i}) \leftarrow \mathtt{TrapGen}(q, n, m)$, for $i \in [t_A]$, $j \in [t_B]$, then invokes the $\mathtt{Sample}$ algorithm to get $\mathbf{L}_w^{i,j} \leftarrow \mathtt{Sample}(\mathbf{C}_{A(w)}^i \| \mathbf{C}_{B(w)}^j, \mathbf{Y}_{A(w)}^i, \mathbf{C}_w^{(j-1)\cdot t_A + i}, \sigma)$. $\mathcal{B}$ sets $K_w = \{\mathbf{L}_w^{i,j}; i \in [t_A], j \in [t_B]\}$.

Finally, since $f_{k+r}(x^*) = 0$, $\mathcal{B}$ knows all the trapdoors of the matrices in $\mathcal{MS}_{k+r}$, and $\mathcal{B}$ can sample $\mathbf{e}_i \leftarrow \texttt{Sample}(\mathbf{C}^r_{k+r}, \mathbf{Y}^i_{k+r}, \mathbf{u}, \sigma)$. $\mathcal{B}$ sets $K_H = (\mathbf{e}_1, ..., \mathbf{e}_{|\mathcal{MS}_{k+r}|})$ and returns $SK_f = (f, K_H, \{K_w\}_{1 \leq w < k+r})$ to $\mathcal{A}$.

Guess $\mathcal{B}$ receives back the guess b' from $\mathcal{A}$. If $b = b'$, $\mathcal{B}$ guesses that the samples are LWE samples; otherwise, it guesses that they are random samples.

This immediately shows that any adversary with non-negligible advantage in the selective security game will have an identical advantage in breaking the LWE assumption. □

Acknowledgements. We are grateful to Xavier Boyen and Jintai Ding for their helpful discussions.

References

1. S. Agrawal, D. Boneh, and X. Boyen. Efficient lattice (h) ibe in the standard model. In *Advances in Cryptology–EUROCRYPT 2010*, pages 553–572. Springer, 2010.
2. S. Agrawal, D. Boneh, and X. Boyen. Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical ibe. In *Advances in Cryptology–CRYPTO 2010*, pages 98–115. Springer, 2010.
3. S. Agrawal, X. Boyen, V. Vaikuntanathan, P. Voulgaris, and H. Wee. Functional encryption for threshold functions (or fuzzy ibe) from lattices. In *Public Key Cryptography*, pages 280–297, 2012.
4. S. Agrawal, D. Freeman, and V. Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In *Advances in Cryptology–ASIACRYPT 2011*, pages 21–40. Springer, 2011.
5. J. Alwen and C. Peikert. Generating shorter bases for hard random lattices. In *Symposium on Theoretical Aspects of Computer Science STACS 2009*, pages 75–86, 2009.
6. M. Bellare, V.T. Hoang, and P. Rogaway. Foundations of garbled circuits. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 784–796. ACM, 2012.
7. D. Boneh and X. Boyen. Secure identity based encryption without random oracles. In *Advances in Cryptology-CRYPTO 2004*, pages 197–206. Springer, 2004.
8. D. Boneh, X. Boyen, and E.J. Goh. Hierarchical identity based encryption with constant size ciphertext. *Advances in Cryptology–EUROCRYPT 2005*, pages 440–456, 2005.
9. D. Boneh, A. Sahai, and B. Waters. Functional encryption: Definitions and challenges. In *Theory of Cryptography*, pages 253–273. Springer, 2011.
10. Dan Boneh and Xavier Boyen. Efficient selective identity-based encryption without random oracles. *Journal of Cryptology (JoC)*, 24(4):659–693, 2011. early version in Eurocrypt 2004.
11. Xavier Boyen. Attribute-based functional encryption on lattices. In *Theory of Cryptography*, pages 122–142. Springer, 2013.
12. Z. Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In *Advances in Cryptology–CRYPTO 2012*, pages 868–886. Springer, 2012.
13. Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 309–325. ACM, 2012.
14. Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. In *Foundations of Computer Science (FOCS) 2011*, pages 97–106. IEEE, 2011.
15. D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert. Bonsai trees, or how to delegate a lattice basis. *Advances in Cryptology–EUROCRYPT 2010*, pages 523–552, 2010.
16. S. Garg, C. Gentry, and S. Halevi. Candidate multilinear maps from ideal lattices and applications. *Advances in Cryptology–EUROCRYPT 2013*, 2013.
17. S. Garg, C. Gentry, S. Halevi, A. Sahai, and B. Waters. Attribute-based encryption for circuits from multilinear maps. CRYPTO, 2013.
18. S. Garg, C. Gentry, A. Sahai, and B. Waters. Witness encryption and its applications. STOC, 2013.
19. C. Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.
20. C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the 40th annual ACM symposium on Theory of computing*, pages 197–206. ACM, 2008.
21. S. Goldwasser, Y. Kalai, R. Popa, V. Vaikuntanathan, and N. Zeldovich. Succinct functional encryption and applications: Reusable garbled circuits and beyond. STOC, 2013.
22. S. Gorbunov, V. Vaikuntanathan, and H. Wee. Predicate encryption for circuits. STOC, 2013.

23. V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM conference on computer and communications security*, pages 89–98. ACM, 2006.
24. M. A. Hamburg. *Spatial encryption.* Stanford University, 2011.
25. J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *Advances in Cryptology–EUROCRYPT 2008*, pages 146–162. Springer, 2008.
26. A. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. *Advances in Cryptology–EUROCRYPT 2010*, pages 62–91, 2010.
27. A. Lewko and B. Waters. Unbounded hibe and attribute-based encryption. In *Advances in Cryptology–EUROCRYPT 2011*, pages 547–567. Springer, 2011.
28. A. Lewko and B. Waters. New proof methods for attribute-based encryption: Achieving full security through selective techniques. In *Advances in Cryptology–CRYPTO 2012*, pages 180–198. Springer, 2012.
29. D. Micciancio and C. Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *Advances in Cryptology–EUROCRYPT 2012*, pages 700–718. Springer, 2012.
30. T. Okamoto and K. Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. *Advances in Cryptology–CRYPTO 2010*, pages 191–208, 2010.
31. T. Okamoto and K. Takashima. Adaptively attribute-hiding (hierarchical) inner product encryption. *Advances in Cryptology–EUROCRYPT 2012*, 2012.
32. A. O'Neill. Definitional issues in functional encryption. Technical report, Cryptology ePrint Archive, Report 2010/556, 2010.
33. B. Parno, M. Raykova, and V. Vaikuntanathan. How to delegate and verify in public: Verifiable computation from attribute-based encryption. In *Theory of Cryptography*, pages 422–439. Springer, 2012.
34. C. Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, pages 333–342. ACM, 2009.
35. O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 84–93. ACM, 2005.
36. A. Sahai and B. Waters. Fuzzy identity-based encryption. *Advances in Cryptology–EUROCRYPT 2005*, pages 457–473, 2005.
37. B. Waters. Dual system encryption: Realizing fully secure ibe and hibe under simple assumptions. *Advances in Cryptology-CRYPTO 2009*, pages 619–636, 2009.
38. B. Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In *Public Key Cryptography*, pages 53–70. Springer, 2011.
39. B. Waters. Functional encryption for regular languages. *Advances in Cryptology–CRYPTO 2012*, pages 218–235, 2012.