Attribute-Based Encryption for Circuits

Sergey Gorbunov[*] Vinod Vaikuntanathan[†] Hoeteck Wee[‡]

May 31, 2013

Abstract

In an attribute-based encryption (ABE) scheme, a ciphertext is associated with an ℓ-bit *public index* ind and a message m, and a secret key is associated with a Boolean predicate P. The secret key allows to decrypt the ciphertext and learn m iff $P(\mathsf{ind}) = 1$. Moreover, the scheme should be secure against collusions of users, namely, given secret keys for polynomially many predicates, an adversary learns nothing about the message if none of the secret keys can individually decrypt the ciphertext.

We present attribute-based encryption schemes for circuits of any arbitrary polynomial size, where the public parameters and the ciphertext grow linearly with the depth of the circuit. Our construction is secure under the standard learning with errors (LWE) assumption. Previous constructions of attribute-based encryption were for Boolean formulas, captured by the complexity class NC^1.

In the course of our construction, we present a new framework for constructing ABE schemes. As a by-product of our framework, we obtain ABE schemes for polynomial-size branching programs, corresponding to the complexity class *LOGSPACE*, under quantitatively better assumptions.

[*]University of Toronto. Email: `sgorbunov@cs.toronto.edu`. Supported by Ontario Graduate Scholarship (OGS).

[†]University of Toronto. Email: `vinodv@cs.toronto.edu`. Supported by an NSERC Discovery Grant and by DARPA under Agreement number FA8750-11-2-0225. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or the U.S. Government.

[‡]George Washington University. Email: `hoeteck@alum.mit.edu`. Supported by NSF CAREER Award CNS-1237429.

1 Introduction

Attribute-based encryption [SW05, GPSW06] is an emerging paradigm for public-key encryption which enables fine-grained control of access to encrypted data. In traditional public-key encryption, access to the encrypted data is all or nothing: given the secret key, one can decrypt and read the entire message, but without it, nothing about the message is revealed (other than its length). In attribute-based encryption, an encryption of a message m is labeled with a *public* attribute vector ind (also called the "index"), and secret keys are associated with predicates P. A secret key sk_P decrypts the ciphertext and recovers the message m if and only if ind satisfies the predicate, namely if and only if $P(\mathsf{ind}) = 1$.

Attribute-based encryption captures as a special case previous cryptographic notions such as identity-based encryption (IBE) [Sha84, BF01, Coc01] and fuzzy IBE [SW05]. It has also found applications in scenarios that demand complex policies to control access to encrypted data, as well as in designing cryptographic protocols for verifiably outsourcing computations [PRV12].

The crucial component in the security requirement for attribute-based encryption stipulates that it resists *collusion attacks*, namely any group of users collectively learns nothing about the message m if none of them is individually authorized to decrypt the ciphertext.

In the past few years, there has been significant progress in attribute-based encryption in terms of efficiency, security guarantees, and diversifying security assumptions [GPSW06, Wat09, LW10, LOS+10, CHKP12, ABB10a, OT10]. On the other hand, little progress has been made in terms of supporting larger classes of predicates. The state of the art is Boolean formulas [GPSW06, LOS+10, OT10], which is a subclass of log-space computations. Constructing a secure attribute-based encryption for all polynomial-time predicates was posed as a central challenge by Boneh, Sahai and Waters [BSW11]. We resolve this problem affirmatively in this work.

2 Our Contributions

We construct attribute-based encryption schemes for circuits of every a-priori bounded depth, based on the learning with errors (LWE) assumption. In the course of our construction, we present a new framework for constructing attribute-based encryption schemes, based on a primitive that we call "two-to-one recoding" (TOR). Our methodology departs significantly from the current line of work on attribute-based encryption [GPSW06, LOS+10] and instead, builds upon the connection to garbled circuits developed in the context of *bounded* collusions [SS10b, GVW12]. Along the way, we make the first substantial progress towards the 25-year-old open problem of constructing (fully) reusable garbled circuits. In a follow-up work, Goldwasser et al. [GKP+13] completely resolved this open problem; moreover, their construction relies crucially on our ABE scheme as an intermediate building block. More details follow.

2.1 Attribute-based encryption

For every class of predicate circuits with depth bounded by a polynomial function $d = d(\lambda)$ (where λ is the security parameter), we construct an ABE scheme that supports this class of circuits, under the learning with errors (LWE) assumption. Informally, the (decisional) LWE problem [Reg09] asks to distinguish between "noisy" random linear combinations of n numbers $\mathbf{s} = (s_1, \ldots, s_n) \in \mathbb{Z}_q^n$ from uniformly random numbers over $\mathbb{Z}_q$.

Regev [Reg09] showed that solving the LWE problem *on the average* is as hard as (quantumly) solving several notoriously difficult lattice problems *in the worst case*. Since then, the LWE assumption has become a central fixture in cryptography. We now have a large body of work building cryptographic schemes under the LWE assumption, culminating in the construction of a fully homomorphic encryption scheme [BV11].

The key parameter that determines the hardness of LWE is the ratio between the modulus q and the maximum absolute value of the noise B; as such, we refer to q/B as the hardness factor of LWE. The problem becomes easier as this ratio grows, but is believed to be hard for 2^{n^ϵ}-time algorithms when $q/B = 2^{O(n^\epsilon)}$, where $0 < \epsilon < 1/2$. Our results will hold as long as the latter holds for *some constant* ϵ.

In particular, we show:

Theorem 2.1 (informal)**.** *Assume that there is a constant* $0 < \epsilon < 1$ *for which the LWE problem is hard for a* $\exp(n^\epsilon)$ *factor in dimension* n*, for all large enough* n*. Then, for any polynomial* d*, there is a* selectively secure *attribute encryption scheme for general circuits of depth* d*.*

Moreover, our scheme has succinct ciphertexts, in the sense that the ciphertext size depends polynomially on the depth d and the length ℓ of the attribute vector ind, but not on the size of the circuits in the class. The construction as stated achieves the weaker notion of selective security, but we can easily obtain a fully secure scheme following [BB04] (but using sub-exponential hardness in a crucial way):

Corollary 2.2. *Assume that there is a constant* $0 < \epsilon < 1/2$ *such that the LWE problem with a factor of* $\exp(n^\epsilon)$ *is hard in dimension* n *for* $\exp(n^\epsilon)$*-time algorithms. Then, for any polynomial* d*, there is a* fully secure *attribute-based encryption scheme for general circuits of depth* d*.*

We also obtain a new ABE scheme for branching programs (which correspond to the complexity class $LOGSPACE$) under the weaker quasi-polynomial hardness of LWE:

Theorem 2.3 (informal)**.** *There exist attribute-based encryption schemes for the class of branching programs under either (1) the hardness of the LWE problem with an* $n^{\omega(1)}$ *factor, or (2) the bilinear decisional Diffie-Hellman assumption.*

Here, there is no a-prori bound on the size or the depth of the branching program. In addition, we achieve succinct ciphertexts of size $O(\ell)$ where ℓ is the number of bits in the index. Prior to this work, we only knew how to realize IBE and inner product encryption under $n^{\omega(1)}$-hardness of LWE [CHKP12, ABB10a, AFV11], whereas our bilinear construction is a different way to achieve the results of Goyal et al. [GPSW06] which uses secret-sharing for general access structures. Our construction exploits a combinatorial property of branching programs to overcome limitations of previous approaches based on secret sharing for monotone formulas (c.f. [ABV$^+$12]). The construction is inspired by a pairings-based scheme for regular languages in [Wat12].

We now move on to provide a technical roadmap of our construction: first, we define a new primitive that we call a *two-to-one recoding* (TOR) scheme; we then show how TOR gives us an attribute-based encryption scheme for circuits, and how to construct a TOR scheme from the LWE assumption.

2.2 New Framework: TOR

A Two-to-One Recoding (TOR) scheme is a family of (probabilistic) functions $\{\mathsf{Encode}(\mathsf{pk}, \cdot)\}$ indexed by pk, together with a "two-to-one" recoding mechanism. The basic computational security guarantee for $\mathsf{Encode}(\mathsf{pk}, \cdot)$ is that of (correlated) pseudorandomness [RS10]: $\mathsf{Encode}(\mathsf{pk}, s)$ should be pseudorandom given $\mathsf{Encode}(\mathsf{pk}_i, s)$ for polynomially many pk_i's, where s is a uniformly random "seed".

The recoding mechanism guaratees that given any triple of public keys $(\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{pk}_{\text{tgt}})$, there is a recoding key rk that allows us to perform the transformation

$$(\mathsf{Encode}(\mathsf{pk}_0, s), \mathsf{Encode}(\mathsf{pk}_1, s)) \mapsto \mathsf{Encode}(\mathsf{pk}_{\text{tgt}}, s).$$

Such a recoding key rk can be generated using either of the two secret keys sk_0 or sk_1. Furthermore, the recoding mechanism must satisfy a natural simulation requirement: namely, we can generate rk given just $\mathsf{pk}_0, \mathsf{pk}_1$ (and neither of the two secret keys), if we are allowed to "program" pk_{tgt}. That is, there are three ways of generating the pair $(\mathsf{pk}_{\text{tgt}}, \mathsf{rk})$ that are (statistically) indistinguishable: (1) given pk_{tgt}, generate rk using the secret key sk_0; (2) given pk_{tgt}, generate rk using the secret key sk_1; and (3) generate rk without either secret key, by "programming" the output public key pk_{tgt}.

This requirement demonstrates the *intuitive guarantee* that we expect from a two-to-one recoding mechanism: namely, the recoding key is "useless" given only one encoding, but not both encodings. For example, it is easy to see that given $\mathsf{Encode}(\mathsf{pk}_0, s)$ and rk (but not $\mathsf{Encode}(\mathsf{pk}_1, s)$), the output $\mathsf{Encode}(\mathsf{pk}_{\text{tgt}}, s)$ is pseudorandom. Indeed, this is because rk could as well have been "simulated" using sk_1, in which case it is of no help in the distinguishing task.

The simulation requirement also rules out the trivial construction from trapdoor functions where rk is a trapdoor for inverting $\mathsf{Encode}(\mathsf{pk}_0, \cdot)$ or $\mathsf{Encode}(\mathsf{pk}_1, \cdot)$.

From TOR to Garbled Circuits. We start from the observation that our TOR primitive implies a form of *reusable* garbled circuits *with no input or circuit privacy*, but instead, with a form of *authenticity guarantee*. As we will see, this leads directly into our attribute-based encryption scheme.

Consider a two-input boolean gate with input wires u, v and output wire w, computing a function $G : \{0,1\} \times \{0,1\} \to \{0,1\}$. In Yao's garbled circuit construction, we associate each wire with a pair of strings (called "labels"), and we provide a translation table comprising of four values $v_{b,c}$ where $v_{b,c}$ allows us to perform the transformation:

$$L_{u,b}, L_{v,c} \mapsto L_{w,G(b,c)}$$

The garbled circuits construction guarantees that given the translation table and labels L_{u,b^*} and L_{v,c^*} for specific input bits b^* and c^*, we can obtain $L_{w,G(b^*,c^*)}$; however, the other label at the output, namely $L_{w,1-G(b^*,c^*)}$ remains hidden.

In our setting, we replace labels with public keys, so that each wire is associated with a pair of public keys. As before, we also provide a translation table comprising four values $\mathsf{rk}_{b,c}$ where the recoding key $\mathsf{rk}_{b,c}$ allows us to perform the transformation

$$\mathsf{Encode}(\mathsf{pk}_{u,b}, s), \mathsf{Encode}(\mathsf{pk}_{v,c}, s) \mapsto \mathsf{Encode}(\mathsf{pk}_{w,G(b,c)}, s)$$

The security properties of the TOR scheme then give us the following guarantee: Given the translation table and encodings of s corresponding to b^*, c^*, we clearly compute the encoding of s corresponding to $G(b^*, c^*)$. However, the encoding corresponding to $1 - G(b^*, c^*)$ remains pseudorandom.

Moreover, crucially, the translation table is independent of s, so we can now "reuse" the translation table by providing fresh encodings with different choices of s. In a sentence, replacing strings by functions gives us the power of reusability.

In the garbled circuits construction, the four entries of the table are permuted and thus, one can perform the translation even without knowing what the input bits b^* and c^* are. This is possible because there is an efficient way to verify when the "correct" translation key is being used. In contrast, in the reusable construction above, one has to know exactly which of the recoding keys to use. This is part of the reason why we are *unable to provide circuit or input privacy, but instead, only guarantee authenticity*, namely that an adversary can obtain only one of the two possible encodings at the output wire.

This construction forms the cornerstone of the subsequent work of Goldwasser, Kalai, Popa, Vaikuntanathan and Zeldovich [GKP+13] who construct reusable garbled circuits with input and circuit privacy, by additionally leveraging the power of fully homomorphic encryption [Gen09, BV11].

From TOR to Attribute-Based Encryption. How is all this related to attribute-based encryption? In our attribute-based encryption scheme for circuits, the encodings of s are provided in the ciphertext, and the translation tables are provided in the secret key. More precisely, each wire is associated with two TOR public keys, and the encryption of a message m under an index ind is obtained by computing $\mathsf{Encode}(\mathsf{pk}_{i,\mathsf{ind}_i}, s)$ for every input wire i. The output encoding $\mathsf{Encode}(\mathsf{pk}_{\mathrm{out}}, s)$ is then used to mask the message. We obtain the secret key corresponding to a circuit C by "stitching" multiple translation tables together, where the public keys for the input and output wires are provided in the public parameters, and we pick fresh public keys for the internal wires during key generation. In a nutshell, this gives us the guarantee that given a secret key sk_C and an encryption $\mathsf{Enc}(\mathsf{ind}, m)$ such that $C(\mathsf{ind}) = 1$, we can compute $\mathsf{Encode}(\mathsf{pk}_{\mathrm{out}}, s)$ and thus recover the message. On the other hand, this value looks pseudorandom if $C(\mathsf{ind}) = 0$.

In our outline of reusable garbled circuits with authenticity, we wanted to reuse the garbled circuit $G(C)$ across multiple encryptions with indices $\mathsf{ind}_1, \mathsf{ind}_2, \ldots$ on which C always evaluates to 0. In attribute-based encryption, we also want reusability across multiple circuits $C_1, C_2, \ldots$ all of which evaluate to 0 on a fixed index ind (in addition to multiple indices). Fortunately, the strong security properties of the TOR primitive provide us with this guarantee.

To obtain attribute-based encryption for branching programs, we are able to support a different notion of translation tables, which we can realize using a slightly weaker notion of TOR. In branching programs, the transition function depends on an input variable and the current state. The fact that one of these two values is always an input variable makes things simpler; in circuits, both of the input values to a gate could be internal wires.

TOR from LWE. We show how to instantiate TOR from LWE, building upon previous lattice-based IBE techniques in [GPV08, CHKP12, ABB10a, ABB10b]. The public key is given by a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, and

$$\mathsf{Encode}(\mathbf{A}, \mathbf{s}) = \mathbf{A}^T\mathbf{s} + \mathbf{e}$$

where $\mathbf{s} \in \mathbb{Z}_q^n$, $\mathbf{e} \in \mathbb{Z}_q^m$ is an error vector, and $\mathbf{A}^T$ denotes the transpose of the matrix $\mathbf{A}$. (Correlated) pseudorandomness follows directly from the LWE assumption. Given $\mathbf{A}_0, \mathbf{A}_1, \mathbf{A}_{\text{tgt}} \in \mathbb{Z}_q^{n \times m}$, the recoding key rk is given by a low-norm matrix $\mathbf{R} \in \mathbb{Z}_q^{2m \times m}$ such that

$$[\, \mathbf{A}_0 \parallel \mathbf{A}_1 \,]\, \mathbf{R} = \mathbf{A}_{\text{tgt}}$$

Note that

$$\mathbf{R}^T \begin{bmatrix} \mathbf{A}_0^T \mathbf{s} + \mathbf{e}_0 \\ \mathbf{A}_1^T \mathbf{s} + \mathbf{e}_1 \end{bmatrix} \approx \mathbf{A}_{\text{tgt}}^T \mathbf{s}$$

which gives us the recoding mechanism. There are three ways of generating the public key $\mathbf{A}_{\text{tgt}}$ together with the recoding key $\mathbf{R}$: (1) using the trapdoor for $\mathbf{A}_0$, (2) using the trapdoor for $\mathbf{A}_1$, or (3) first generating $\mathbf{R}$ and then "programming" $\mathbf{A}_{\text{tgt}} := [\mathbf{A}_0 || \mathbf{A}_1]\ \mathbf{R}$. These three ways are statistically indistinguishable by the "bonsai trick" of [CHKP12]. In fact, our recoding mechanism is very similar to the lattice delegation mechanism introduced in [ABB10b], which also uses random low norm matrices to move from one lattice to another.

The multiplicative mechanism for recoding means that the noise grows exponentially with the number of sequential recodings. This, in turn, limits the depth of the circuits we can handle. In particular, the noise grows by a multiplicative $\mathsf{poly}(n)$ factor on each recoding, which means that after depth d, it becomes $n^{O(d)}$. Since $n^{O(d)} < q/4 < 2^{n^\epsilon}$, we can handle circuits of depth $\tilde{O}(n^\epsilon)$ (here, the first inequality is for correctness and the second for security). Viewed differently, setting the LWE dimension $n = d^{1/\epsilon}$ lets us handle circuits of maximum depth $d = d(\ell)$.

Our weak TOR for branching programs uses an additive mechanism, namely the recoding key is given by a low-norm matrix $\mathbf{R} \in \mathbb{Z}_q^{m \times m}$ such that $\mathbf{A}_0 \mathbf{R} = \mathbf{A}_{\text{tgt}} - \mathbf{A}_1$. Note that $\mathbf{R}^T (\mathbf{A}_0^T \mathbf{s} + \mathbf{e}_0) + (\mathbf{A}_1^T \mathbf{s} + \mathbf{e}_1) \approx \mathbf{A}_{\text{tgt}}^T \mathbf{s}$ which gives us our recoding mechanism. Since in our branching program construction, $\mathbf{A}_0^T \mathbf{s} + \mathbf{e}_0$ will always be a fresh encoding provided in the ciphertext, the noise accumulation is additive rather than multiplicative.

2.3 Applications

Let us now explain the application of our result to the problem of publicly verifiable delegation of computation without input privacy.

A verifiable delegation scheme allows a computationally weak client to delegate expensive computations to the cloud, with the assurance that a malicious cloud cannot convince the client to accept an incorrect computation [Mic00, GKR08, GGP10, CKV10, AIK10]. Recent work of Parno, Raykova and Vaikuntanathan [PRV12] showed that any attribute-based encryption scheme for a class of circuits with encryption time at most linear in the length of the index immediately yields a two-message delegation scheme for the class in the pre-processing model. Namely, there is an initial pre-processing phase which fixes the circuit C the client wishes to compute, produces a circuit key and sends it to the server. Afterwards, to delegate computation on an input x, the client only needs to send a single message. Moreover, the ensuing delegation scheme satisfies public delegatability, namely anyone can delegate computations to the cloud; as well as public verifiability, namely anyone can check the cloud's work (given a "verification" key published by the client). The previous delegation schemes that satisfy both these properties (secure in the standard model) supported the class NC^1 [PRV12, GPSW06, LW12]. Our attribute-based encryption schemes for circuits gives us a verifiable delegation scheme for all circuits, where the computation time of the client in the online phase is polynomial in the length of its input and the depth of the circuit, but is otherwise independent of the circuit size. We note that this scheme does not guarantee privacy of the input. Building on this work, Goldwasser et al. [GKP+13] show how to achieve a publicly verifiable delegation scheme with input privacy.

2.4 Related Work

Prior to this work, the state-of-art for lattice-based predicate encryption was threshold and inner product predicates [ABV+12, AFV11]; realizing Boolean formula was itself an open problem. A different line of work considers definitional issues in the more general realm of functional encryption [BSW11, O'N10], for which general feasibility results are known for the restricted setting of a-priori bounded collusions developed from classical "one-time" garbled circuits [SS10a, GVW12] (the ciphertext size grows with both the circuit size and the collusion bound). Our methodology takes a fresh perspective on how to achieve reusability of garbled circuits with respect to authenticity. Our primitive (TOR) can be thought of as a generalization of the notion of proxy re-encryption [BBS98, AFGH06, HRSV11] which can be thought of as a one-to-one re-encryption mechanism.

Independent work. Boyen [Boy13] gave a construction of an ABE scheme for Boolean formulas based on LWE; our result for LWE-based branching program subsumes the result since Boolean formulas are a subclass of branching programs. Garg, Gentry, Halevi, Sahai and Waters [GGH+13] gave a construction of attribute-based encryption for general circuits under a DBDH-like assumption in multi-linear groups (unfortunately, there is no known candidate for realizing such an assumption), as well as a non-standard assumption in ideal lattices [GGH12]. The public parameters in the construction also grow with the depth of the circuit.

Subsequent Work. Our attribute-based encryption scheme has been used as the crucial component in the subsequent work of [GKP+13] to construct a (private index) functional encryption scheme with succinct ciphertexts. They also show a number of applications of their construction, including reusable garbled circuits *with input and circuit privacy.*

Organization. We present our TOR framework and its instantiation in Sections 4 and 5. We present our ABE scheme in Section 6. We present the scheme for branching programs in Section 7.

3 Preliminaries

Notation. Let PPT denote probabilistic polynomial-time. 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 $1 - \mathrm{negl}(n)$. The function $\lg x$ is the base 2 logarithm of x. The notation $\lfloor x \rceil$ denotes the nearest integer to x, rounding towards 0 for half-integers.

3.1 Attribute-Based Encryption

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

$\mathsf{Setup}(1^\lambda, 1^\ell) \to (\mathsf{pp}, \mathsf{mpk}, \mathsf{msk})$: The setup algorithm gets as input the security parameter λ, the length ℓ of the index, and outputs the public parameter $(\mathsf{pp}, \mathsf{mpk})$, and the master key msk. All the other algorithms get pp as part of its input.

$\mathsf{Enc}(\mathsf{mpk}, \mathsf{ind}, m) \to \mathsf{ct}_{\mathsf{ind}}$: The encryption algorithm gets as input mpk, an index $\mathsf{ind} \in \{0,1\}^\ell$ and a message $m \in \mathcal{M}$. It outputs a ciphertext $\mathsf{ct}_{\mathsf{ind}}$. Note that ind is public given $\mathsf{ct}_{\mathsf{ind}}$.

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

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

We require that for all (ind, C) such that $C(\mathsf{ind}) = 1$, all $m \in \mathcal{M}$ and $\mathsf{ct}_{\mathsf{ind}} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{ind}, m)$, $\mathsf{Dec}(\mathsf{sk}_C, \mathsf{ct}_{\mathsf{ind}}) = m$.

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

$$\Pr\left[b = b' : \begin{array}{l} \mathsf{ind} \leftarrow \mathcal{A}(1^\lambda, 1^\ell); \\ (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell); \\ (m_0, m_1) \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk}, \cdot)}(\mathsf{mpk}), |m_0| = |m_1|; \\ b \xleftarrow{\$} \{0,1\}; \\ \mathsf{ct}_{\mathsf{ind}} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{ind}, m_b); \\ b' \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk}, \cdot)}(\mathsf{ct}_{\mathsf{ind}}) \end{array} \right] - \frac{1}{2}$$

with the restriction that all queries C that $\mathcal{A}$ makes to $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ satisfies $C(\mathsf{ind}) = 0$ (that is, sk_C does not decrypt $\mathsf{ct}_{\mathsf{ind}}$). an attribute-based encryption scheme is *selectively secure* if for all PPT adversaries $\mathcal{A}$, the advantage $\mathsf{Adv}^{\mathrm{PE}}_{\mathcal{A}}(\lambda)$ 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 ind after seeing secret keys, namely, along with choosing (m_0, m_1).

3.2 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:*

$$\{\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{x}\} \quad \text{and} \quad \{\mathbf{A}, \mathbf{u}\}$$

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 poly(n)-bounded distribution χ and q as large as 2^{n^ϵ} (for any constant $0 < \epsilon < 1$). Throughout this paper, the parameter $m = \text{poly}(n)$, in which case we will shorten the notation slightly to $\mathsf{LWE}_{n,q,\chi}$.

3.3 Trapdoors for Lattices and LWE

Gaussian distributions. Let $D_{\mathbb{Z}^m,\sigma}$ be the truncated discrete Gaussian distribution over $\mathbb{Z}^m$ with parameter σ, that is, we replace the output by $\mathbf{0}$ whenever the $|| \cdot ||_\infty$ norm exceeds $\sqrt{m} \cdot \sigma$. Note that $D_{\mathbb{Z}^m,\sigma}$ is $\sqrt{m} \cdot \sigma$-bounded.

Lemma 3.1 (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* negl(n)*-close to uniform.*

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 negl(n) *statistical distance of each other for any polynomial $k \in \mathbb{N}$:*

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

4 Two-to-One Recoding Schemes

An overview is provided in Section 2.2.

Symmetric encryption. In our construction, we will use $\mathsf{Encode}(\mathsf{pk}, s)$ as a one-time key for a symmetric-key encryption scheme (E, D). If Encode is deterministic, then we could simply use a one-time pad. However, since Encode is probabilistic, the one-time pad will not guarantee correctness. Instead, we require (E, D) to satisfy a stronger correctness guarantee, namely for all messages m and for all ψ, ψ' in the support $\mathsf{Encode}(\mathsf{pk}, s)$, $\mathsf{D}(\psi', \mathsf{E}(\psi, m)) = m$.

Allowing degradation. With each recoding operation, the "quality" of encoding potentially degrades. In order to formalize this, we allow the initial global public parameters to depend on $d_{\max}$, an a-prior upper bound on the number of nested recoding operations. We then require that given any encodings ψ and ψ' that are a result of at most $d_{\max}$ nested recodings, $\mathsf{D}(\psi', \mathsf{E}(\psi, m)) = m$. We stress that we allow $d_{\max}$ to be super-polynomial, and in fact, provide such instantiations for a relaxed notion of TOR.

4.1 Definition of TOR

Formally, a TOR scheme over the input space $\mathcal{S} = \{\mathcal{S}_\lambda\}$ consists of six polynomial-time algorithms $(\mathsf{Params}, \mathsf{Keygen}, \mathsf{Encode}, \mathsf{ReKeyGen}, \mathsf{SimReKeyGen}, \mathsf{Recode})$ and a symmetric-key encryption scheme (E, D) with the following properties:

- $\mathsf{Params}(1^\lambda, d_{\max})$ is a probabilistic algorithm that takes as input the security parameter λ and an upper bound $d_{\max}$ on the number of nested recoding operations (written in binary), outputs "global" public parameters pp.
- $\mathsf{Keygen}(\mathsf{pp})$ is a probabilistic algorithm that outputs a public/secret key pair $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{Encode}(\mathsf{pk}, s)$ is a probabilistic algorithm that takes pk and an input $s \in \mathcal{S}$, and outputs an encoding ψ.

In addition, there is a recoding mechanism together with two ways to generate recoding keys: given one of the two secret keys, or by programming the output public key.

- $\mathsf{ReKeyGen}(\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{sk}_0, \mathsf{pk}_{\mathsf{tgt}})$ is a probabilistic algorithm that takes a key pair $(\mathsf{pk}_0, \mathsf{sk}_0)$, another public key pk_1, a "target" public key $\mathsf{pk}_{\mathsf{tgt}}$, and outputs a recoding key rk.
- $\mathsf{SimReKeyGen}(\mathsf{pk}_0, \mathsf{pk}_1)$ is a probabilistic algorithm that takes two public keys $\mathsf{pk}_0, \mathsf{pk}_1$ and outputs a recoding key rk together with a "target" public key $\mathsf{pk}_{\mathsf{tgt}}$.
- $\mathsf{Recode}(\mathsf{rk}, \psi_0, \psi_1)$ is a deterministic algorithm that takes the recoding key rk, two encodings ψ_0 and ψ_1, and outputs an encoding ψ_{tgt}.

Remark 4.1. *For our instantiation from lattices, we can in fact invert* $\mathsf{Encode}(\mathsf{pk}, s)$ *to recover* s *using the corresponding* sk. *However, we will not require this property in our generic constructions from TOR. Indeed, realizing this property over bilinear groups would be hard, since* s *is typically encoded in the exponent.*

Correctness. Correctness of a TOR scheme requires two things. First, for every pk and $s \in \mathcal{S}$, there exists a family of sets $\Psi_{\mathsf{pk},s,j}, j = 0, 1, \ldots, d_{\max}$:

- $\Pr[\mathsf{Encode}(\mathsf{pk}, s) \in \Psi_{\mathsf{pk},s,0}] = 1$, where the probability is taken over the coin tosses of Encode;
- $\Psi_{\mathsf{pk},s,0} \subseteq \Psi_{\mathsf{pk},s,1} \subseteq \cdots \subseteq \Psi_{\mathsf{pk},s,d_{\max}}$.
- for all $\psi, \psi' \in \Psi_{\mathsf{pk},s,d_{\max}}$ and all $m \in \mathcal{M}$, $\mathsf{D}(\psi', \mathsf{E}(\psi, m)) = m$.

Note that these properties hold trivially if Encode is deterministic and (E, D) is the one-time pad. Secondly, the correctness of recoding requires that for any triple of key pairs $(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1), (\mathsf{pk}_{\mathrm{tgt}}, \mathsf{sk}_{\mathrm{tgt}})$, and any encodings $\psi_0 \in \Psi_{\mathsf{pk}_0,s,j_0}$ and $\psi_1 \in \Psi_{\mathsf{pk}_1,s,j_1}$,

$$\mathsf{Recode}(\mathsf{rk}, \psi_0, \psi_1) \in \Psi_{\mathsf{pk}_{\mathrm{tgt}},s,\max(j_0,j_1)+1}$$

Statistical Security Properties. Note that we have three ways of sampling recoding keys: using $\mathsf{ReKeyGen}$ along with one of two secret keys sk_0 or sk_1; using $\mathsf{SimReKeyGen}$ while programming $\mathsf{pk}_{\mathrm{tgt}}$. We require that all three ways lead to the same distribution of recoding keys, up to some statistical error.

(Key Indistinguishability) : Let $(\mathsf{pk}_b, \mathsf{sk}_b) \leftarrow \mathsf{Keygen}(\mathsf{pp})$ for $b = 0, 1$ and $(\mathsf{pk}_{\mathrm{tgt}}, \mathsf{sk}_{\mathrm{tgt}}) \leftarrow \mathsf{Keygen}(\mathsf{pp})$.

The following two ensembles must be statistically indistinguishable:

$$\Big[\mathsf{Aux}, \mathsf{ReKeyGen}(\mathsf{pk}_0, \mathsf{pk}_1, \boxed{\mathsf{sk}_0}, \mathsf{pk}_{\mathrm{tgt}})\Big] \overset{s}{\approx} \Big[\mathsf{Aux}, \mathsf{ReKeyGen}(\mathsf{pk}_1, \mathsf{pk}_0, \boxed{\mathsf{sk}_1}, \mathsf{pk}_{\mathrm{tgt}})\Big]$$

where $\mathsf{Aux} = ((\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1), (\mathsf{pk}_{\mathrm{tgt}}, \mathsf{sk}_{\mathrm{tgt}}))$. Informally, this says that sampling recoding keys using sk_0 or sk_1 yields the same distribution.

(Recoding Simulation) : Let $(\mathsf{pk}_b, \mathsf{sk}_b) \leftarrow \mathsf{Keygen}(\mathsf{pp})$ for $b = 0, 1$. Then, the following two ways of sampling the tuple $\big[(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1), \mathsf{pk}_{\mathrm{tgt}}, \mathsf{rk}\big]$ must be statistically indistinguishable:

$$\Big[(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1), \mathsf{pk}_{\mathrm{tgt}}, \mathsf{rk} : (\mathsf{pk}_{\mathrm{tgt}}, \mathsf{sk}_{\mathrm{tgt}}) \leftarrow \mathsf{Keygen}(\mathsf{pp}); \mathsf{rk} \leftarrow \mathsf{ReKeyGen}(\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{sk}_0, \mathsf{pk}_{\mathrm{tgt}})\Big] \overset{s}{\approx} \Big[(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1), \mathsf{pk}_{\mathrm{tgt}}, \mathsf{rk} : (\mathsf{pk}_{\mathrm{tgt}}, \mathsf{rk}) \leftarrow \mathsf{SimReKeyGen}(\mathsf{pk}_0, \mathsf{pk}_1)\Big]$$

In addition, we require one-time semantic security for (E, D):

(One-time Semantic Security) : For all $m_0, m_1 \in \mathcal{M}$, the following two distributions must be statistically indistinguishable:

$$\Big[\ \mathsf{E}(\psi, m_0) : \psi \overset{\$}{\leftarrow} \mathcal{K}\ \Big] \overset{s}{\approx} \Big[\ \mathsf{E}(\psi, m_1) : \psi \overset{\$}{\leftarrow} \mathcal{K}\ \Big]$$

For all three properies, computational indistinguishability is sufficient for our applications, but we will achieve the stronger statistical indistinguishability in our instantiations.

Computational Security Property. We require that given the encoding of a random s on $\ell = \mathrm{poly}(\lambda)$ keys, the evaluation at a fresh key is pseudorandom.

(Correlated Pseudorandomness) : For every polynomial $\ell = \ell(\lambda)$, let $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Keygen}(\mathsf{pp})$ for $i \in [\ell+1]$. Let $s \xleftarrow{\$} \mathcal{S}$, and let $\psi_i \leftarrow \mathsf{Encode}(\mathsf{pk}_i, s)$ for $i \in [\ell+1]$. Then, the following two ensembles must be computationally indistinguishable:

$$\Big[\ (\mathsf{pk}_i, \psi_i)_{i\in[\ell]}, \mathsf{pk}_{\ell+1}, \boxed{\psi_{\ell+1}}\ \Big] \overset{c}{\approx}$$
$$\Big[\ (\mathsf{pk}_i, \psi_i)_{i\in[\ell]}, \mathsf{pk}_{\ell+1}, \boxed{\psi} : \psi \xleftarrow{\$} \mathcal{K}\ \Big]$$

That is, we define the advantage function $\mathsf{Adv}^{\mathrm{CP}}_{\mathcal{A}}(\lambda)$ to be:

$$\Pr\left[b = b' : \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda); s \leftarrow \mathcal{S}; \\ (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Keygen}(\mathsf{pp}), \\ \psi_i \leftarrow \mathsf{Encode}(\mathsf{pk}_i, s), i = 1, \ldots, \ell; \\ \psi'_0 \leftarrow \mathsf{Encode}(\mathsf{pk}_{\ell+1}, s); \\ b \xleftarrow{\$} \{0,1\}; \psi'_1 \xleftarrow{\$} \mathcal{K} \\ b' \leftarrow \mathcal{A}(\mathsf{pk}_1, \ldots, \mathsf{pk}_{\ell+1}, \psi_1, \ldots, \psi_\ell, \psi'_b) \end{array} \right] - \frac{1}{2}$$

and we require that for all PPT $\mathcal{A}$, the advantage function $\mathsf{Adv}^{\mathrm{CP}}_{\mathcal{A}}(\lambda)$ is a negligible function in λ.

4.2 Simple Applications of TOR

First example. We revisit the example from Section 2.2. Consider a two-input boolean gate g with input wires u, v and output wire w, computing a function $G : \{0,1\} \times \{0,1\} \to \{0,1\}$. Analogous to Yao's garbled circuit, we provide a translation table Γ comprising four values

$$\Gamma := (\ \mathsf{rk}_{b,c}\ :\ b, c \in \{0,1\}\ )$$

where $\mathsf{rk}_{b,c}$ allows us to perform the transformation

$$\mathsf{Encode}(\mathsf{pk}_{u,b}, s), \mathsf{Encode}(\mathsf{pk}_{v,c}, s) \mapsto \mathsf{Encode}(\mathsf{pk}_{w,G(b,c)}, s)$$

Now, fix b^*, c^* and $d^* := G(b^*, c^*)$. Given an encoding of s corresponding to b^* and c^*, we can compute that under for d^* using the recoding key rk_{b^*,c^*}; in addition, we claim that the encoding corresponding to $1 - d^*$ remains pseudorandom. To prove this, it suffices to simulate Γ given $\mathsf{pk}_{u,b^*}, \mathsf{pk}_{v,c^*}, \mathsf{pk}_{w,1-d^*}$ as follows:

- we sample $(\mathsf{pk}_{w,d^*}, \mathsf{rk}_{b^*,c^*})$ using $\mathsf{SimReKeyGen}$;
- we sample $\mathsf{pk}_{u,1-b^*}$ and $\mathsf{pk}_{v,1-c^*}$ along with the corresponding secret keys; using these secret keys, we can sample the other three recoding keys $\mathsf{rk}_{1-b^*,c^*}, \mathsf{rk}_{b^*,1-c^*}, \mathsf{rk}_{1-b^*,1-c^*}$.

IBE from TOR. As a warm-up, we show how to build a selectively secure IBE for identity space $\{0,1\}^\ell$.

$$\mathsf{mpk} := \begin{pmatrix} \mathsf{pk}_{1,0} & \mathsf{pk}_{2,0} & \cdots & \mathsf{pk}_{\ell,0} & \mathsf{pk}_{\text{start}} \\ \mathsf{pk}_{1,1} & \mathsf{pk}_{2,1} & \cdots & \mathsf{pk}_{\ell,1} & \mathsf{pk}_{\text{out}} \end{pmatrix}$$

The ciphertext for identity ind and message m is given by:

$$\Big(\mathsf{Encode}(\mathsf{pk}_{1,\mathsf{ind}_1}, s), \ldots, \mathsf{Encode}(\mathsf{pk}_{\ell,\mathsf{ind}_\ell}, s), \mathsf{Encode}(\mathsf{pk}_{\text{start}}, s), \mathsf{E}(\mathsf{Encode}(\mathsf{pk}_{\text{out}}, s), m) \Big)$$

The secret key for identity ind is given by $(\mathsf{rk}_1, \ldots, \mathsf{rk}_\ell)$ where we first sample

$$(\mathsf{pk}'_1, \mathsf{sk}'_1), \ldots, (\mathsf{pk}'_{\ell-1}, \mathsf{sk}'_{\ell-1}) \leftarrow \mathsf{Keygen}(\mathsf{pp})$$

and then sample

$$\begin{aligned} \mathsf{rk}_1 &\leftarrow \mathsf{ReKeyGen}(\mathsf{pk}_{\text{start}}, \mathsf{pk}_{1,\mathsf{ind}_1}, \mathsf{sk}_{\text{start}}, \mathsf{pk}'_1) \\ \mathsf{rk}_2 &\leftarrow \mathsf{ReKeyGen}(\mathsf{pk}'_1, \quad \mathsf{pk}_{2,\mathsf{ind}_2}, \mathsf{sk}'_1, \quad \mathsf{pk}'_2) \\ &\vdots \\ \mathsf{rk}_\ell &\leftarrow \mathsf{ReKeyGen}(\mathsf{pk}'_{\ell-1}, \; \mathsf{pk}_{\ell,\mathsf{ind}_\ell}, \mathsf{sk}'_{\ell-1}, \; \mathsf{pk}_{\text{out}}) \end{aligned}$$

To prove selective security, we need to generate secret keys for any $\mathsf{ind} \neq \mathsf{ind}^*$, given $\mathsf{sk}_{1,1-\mathsf{ind}^*_1}, \ldots, \mathsf{sk}_{\ell,1-\mathsf{ind}^*_\ell}$ but not $\mathsf{sk}_{\text{start}}$ or sk_{out}. We can achieve this as follows: pick an i for which $\mathsf{ind}_i \neq \mathsf{ind}^*_i$;

- pick $(\mathsf{rk}_1, \mathsf{pk}'_1), \ldots, (\mathsf{rk}_{i-1}, \mathsf{pk}'_{i-1})$ using $\mathsf{SimReKeyGen}$;
- pick $(\mathsf{pk}'_i, \mathsf{sk}'_i), \ldots, (\mathsf{pk}'_{\ell-1}, \mathsf{sk}'_{\ell-1})$ using Keygen;
- pick $\mathsf{rk}_i, \mathsf{rk}_{i+1}, \ldots, \mathsf{rk}_\ell$ using $\mathsf{ReKeyGen}$ with secret keys $\mathsf{sk}_{1-\mathsf{ind}^*_i}, \mathsf{sk}'_i, \ldots, \mathsf{sk}'_{\ell-1}$ respectively.

5 TOR from LWE

In this section, we present an instantiation of TOR from LWE, building upon ideas previously introduced in [GPV08, CHKP12, ABB10a, ABB10b].

Lemma 5.1. *Assuming $\mathsf{dLWE}_{n,q,\chi}$ where $q = n^{\Theta(d_{\max})}$, there is a TOR scheme that is correct up to $d_{\max}$ levels.*

- $\mathsf{Params}(1^\lambda, d_{\max})$: First choose the LWE dimension $n = n(\lambda)$. Let the error distribution $\chi = \chi(n) = D_{\mathbb{Z},\sqrt{n}}$, the error bound $B = B(n) = O(n)$, the modulus $q = q(n) = \tilde{O}(n^2 d_{\max})^{d_{\max}} n$, the number of samples $m = m(n) = O(n \log q)$ and the Gaussian parameter $s = s(n) = O(\sqrt{n \log q})$. Output the global public parameters $\mathsf{pp} = (n, \chi, B, q, m, s)$. Define the domain $\mathcal{S}$ of the encoding scheme to be $\mathbb{Z}_q^n$.
- $\mathsf{Keygen}(\mathsf{pp})$: Run the trapdoor generation algorithm $\mathsf{TrapGen}(1^n, 1^m, q)$ to obtain a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ together with the trapdoor matrix $\mathbf{T} \in \mathbb{Z}^{m \times m}$. Output $\mathsf{pk} := \mathbf{A}$ and $\mathsf{sk} := \mathbf{T}$.
- $\mathsf{Encode}(\mathsf{pk}, s)$: Sample an error vector $\mathbf{e} \leftarrow \chi^m$ and output the encoding $\boldsymbol{\psi} := \mathbf{A}^T\mathbf{s} + \mathbf{e} \in \mathbb{Z}_q^n$.

The recoding algorithms work as follows:

- $\mathsf{ReKeyGen}(\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{sk}_b, \mathsf{pk}_{\mathrm{tgt}})$: Let $\mathsf{pk}_0 = \mathbf{A}_0$, $\mathsf{pk}_1 = \mathbf{A}_1$, $\mathsf{sk}_b = \mathbf{T}_b$ and $\mathsf{pk}_{\mathrm{tgt}} = \mathbf{A}_{\mathrm{tgt}}$. Compute the matrix $\mathbf{R} \in \mathbb{Z}^{2m \times m}$ in the following way:
  - Choose a discrete Gaussian matrix $\mathbf{R}_{1-b} \leftarrow (D_{\mathbb{Z},s})^{m \times m}$. Namely, each entry of the matrix is an independent sample from the discrete Gaussian distribution $D_{\mathbb{Z},s}$.
  - Compute $\mathbf{U} := \mathbf{A}_{\mathrm{tgt}} - \mathbf{A}_{1-b}\mathbf{R}_{1-b} \in \mathbb{Z}_q^{n \times m}$.
  - Compute the matrix $\mathbf{R}_b$ by running the algorithm $\mathsf{SampleD}$ to compute a matrix $\mathbf{R}_b \in \mathbb{Z}^{m \times m}$ as follows:
  $$\mathbf{R}_b \leftarrow \mathsf{SampleD}(\mathbf{A}_b, \mathbf{T}_b, \mathbf{U})$$

  Output
  $$\mathsf{rk}_{0,1}^{\mathrm{tgt}} := \begin{bmatrix} \mathbf{R}_0 \\ \mathbf{R}_1 \end{bmatrix} \in \mathbb{Z}^{2m \times m}$$
  (We remark that $\mathbf{A}_b\mathbf{R}_b = \mathbf{U} = \mathbf{A}_{\mathrm{tgt}} - \mathbf{A}_{1-b}\mathbf{R}_{1-b}$, and thus, $\mathbf{A}_0\mathbf{R}_0 + \mathbf{A}_1\mathbf{R}_1 = \mathbf{A}_{\mathrm{tgt}}$).

- $\mathsf{SimReKeyGen}(\mathsf{pk}_0, \mathsf{pk}_1)$: Let $\mathsf{pk}_0 = \mathbf{A}_0$ and $\mathsf{pk}_1 = \mathbf{A}_1$.
  - Sample a matrix $\mathbf{R} \leftarrow (D_{\mathbb{Z},s})^{2m \times m}$ by sampling each entry from the discrete Gaussian distribution $D_{\mathbb{Z},s}$.
  - Define
  $$\mathbf{A}_{\mathrm{tgt}} := [\mathbf{A}_0 \mid\mid \mathbf{A}_1]\ \mathbf{R} \in \mathbb{Z}_q^{n \times m}$$

  Output the pair $(\mathsf{pk}_{\mathrm{tgt}} := \mathbf{A}_{\mathrm{tgt}}, \mathsf{rk}_{0,1}^{\mathrm{tgt}} := \mathbf{R})$.

- $\mathsf{Recode}(\mathsf{rk}_{0,1}^{\mathrm{tgt}}, \boldsymbol{\psi}_0, \boldsymbol{\psi}_1)$: Let $\mathsf{rk}_{0,1}^{\mathrm{tgt}} = \mathbf{R}$. Compute the recoded ciphertext
$$\boldsymbol{\psi}_{\mathrm{tgt}} = [\boldsymbol{\psi}_0^T \mid\mid \boldsymbol{\psi}_1^T]\ \mathbf{R}$$

We also need a one-time symmetric encryption scheme (E, D) which we will instantiate as an *error-tolerant* version of the one-time pad with $\mathcal{K} = \mathbb{Z}_q^n, \mathcal{M} = \{0,1\}^n$, as follows:

- $\mathsf{E}(\boldsymbol{\psi}, \boldsymbol{\mu})$ takes as input a vector $\boldsymbol{\psi} \in \mathbb{Z}_q^n$ and a bit string $\boldsymbol{\mu} \in \mathcal{M}$ and outputs the encryption
$$\boldsymbol{\gamma} := \boldsymbol{\psi} + \lceil q/2 \rceil\ \boldsymbol{\mu} \pmod q$$

- $\mathsf{D}(\boldsymbol{\psi}', \boldsymbol{\gamma})$ takes as input a vector $\boldsymbol{\psi}' = (\psi_1', \ldots, \psi_n') \in \mathbb{Z}_q^n$, an encryption $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_n) \in \mathbb{Z}_q^n$ and does the following. Define a function $\mathsf{Round}(x)$ where $x \in [-(q-1)/2, \ldots, (q-1)/2]$ as:
$$\mathsf{Round}(x) = \begin{cases} 0 & \text{if } |x| < q/4 \\ 1 & \text{otherwise} \end{cases}$$

  The decryption algorithm outputs a vector $\boldsymbol{\mu} = (\mathsf{Round}(\gamma_1 - \psi_1'), \ldots, \mathsf{Round}(\gamma_n - \psi_n'))$.

  We defer the analysis of (E, D) to the full version.

5.1 Analysis

Correctness. We define the sets $\Psi_{\mathbf{A},\mathbf{s},j}$ for $\mathsf{pk} := \mathbf{A} \in \mathbb{Z}_q^{n\times m}$, $\mathbf{s} \in \mathbb{Z}_q^n$ and $j \in [1 \dots d_{\max}]$ as follows:

$$\Psi_{\mathbf{A},\mathbf{s},j} = \left\{ \mathbf{A}^T\mathbf{s} + \mathbf{e} : \; ||\mathbf{e}||_\infty \leq B \cdot (2sm\sqrt{m})^j \right\}$$

Given this definition:

- Observe that when $\mathbf{e} \leftarrow \chi^m$, $||\mathbf{e}||_\infty \leq B$ by the definition of χ and B. $\Pr[\mathsf{Encode}(\mathbf{A},\mathbf{s}) \in \Psi_{\mathbf{A},\mathbf{s},0}] = 1$.
- $\Psi_{\mathbf{A},\mathbf{s},0} \subseteq \Psi_{\mathbf{A},\mathbf{s},1} \subseteq \dots \subseteq \Psi_{\mathbf{A},\mathbf{s},d_{\max}}$, by definition of the sets above.
- For any two encodings $\boldsymbol{\psi} = \mathbf{A}^T\mathbf{s} + \mathbf{e}, \boldsymbol{\psi}' = \mathbf{A}^T\mathbf{s} + \mathbf{e}' \in \Psi_{\mathbf{A},\mathbf{s},d_{\max}}$,

  $$||\boldsymbol{\psi} - \boldsymbol{\psi}'||_\infty = ||\mathbf{e} - \mathbf{e}'||_\infty \leq 2 \cdot B \cdot (2sm\sqrt{m})^{d_{\max}} < q/4,$$

  which holds as long as $n \cdot O(n^2 \log q)^{d_{\max}} < q/4$. Thus, $\boldsymbol{\psi}$ and $\boldsymbol{\psi}'$ are "close", and by the correctness property of the symmetric encryption scheme (E,D) described above, $\mathsf{D}(\boldsymbol{\psi}', \mathsf{E}(\boldsymbol{\psi}, \boldsymbol{\mu})) = \boldsymbol{\mu}$ for any $\boldsymbol{\mu} \in \{0,1\}^n$.
- Consider two encodings $\boldsymbol{\psi}_0 \in \Psi_{\mathbf{A}_0,\mathbf{s},j_0}$ and $\boldsymbol{\psi}_1 \in \Psi_{\mathbf{A}_1,\mathbf{s},j_1}$ for any $j_0, j_1 \in \mathbb{N}$, any $\mathbf{A}_0, \mathbf{A}_1 \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{s} \in \mathbb{Z}_q^n$. Then, $\boldsymbol{\psi}_0 = \mathbf{A}_0^T\mathbf{s} + \mathbf{e}_0$ and $\boldsymbol{\psi}_1 := \mathbf{A}_1^T\mathbf{s} + \mathbf{e}_1$ where $||\mathbf{e}_0||_\infty \leq B \cdot (2sm\sqrt{m})^{j_0}$ and $||\mathbf{e}_1||_\infty \leq B \cdot (2sm\sqrt{m})^{j_1}$.

  Then, the recoded ciphertext $\boldsymbol{\psi}_{\text{tgt}}$ is computed as follows:

  $$\begin{aligned} \boldsymbol{\psi}_{\text{tgt}}^T &:= \left[\boldsymbol{\psi}_0^T \mid\mid \boldsymbol{\psi}_1^T\right] \mathbf{R}_{0,1}^{\text{tgt}} \\ &= \left[\mathbf{s}^T\mathbf{A}_0 + \mathbf{e}_0^T \mid\mid \mathbf{s}^T\mathbf{A}_1 + \mathbf{e}_1^T\right] \mathbf{R}_{0,1}^{\text{tgt}} \\ &= \mathbf{s}^T \left[\mathbf{A}_0 \mid\mid \mathbf{A}_1\right] \mathbf{R}_{0,1}^{\text{tgt}} + \left[\mathbf{e}_0^T \mid\mid \mathbf{e}_1^T\right] \mathbf{R}_{0,1}^{\text{tgt}} \\ &= \mathbf{s}^T\mathbf{A}_{\text{tgt}} + \mathbf{e}_{\text{tgt}} \end{aligned}$$

  where the last equation is because $\mathbf{A}_{\text{tgt}} = \left[\mathbf{A}_0 \mid\mid \mathbf{A}_1\right] \mathbf{R}_{0,1}^{\text{tgt}}$ and we define $\mathbf{e}_{\text{tgt}} := \left[\mathbf{e}_0^T \mid\mid \mathbf{e}_1^T\right] \mathbf{R}_{0,1}^{\text{tgt}}$. Thus,

  $$\begin{aligned} ||\mathbf{e}_{\text{tgt}}||_\infty &\leq m \cdot ||\mathbf{R}_{0,1}^{\text{tgt}}||_\infty \cdot (||\mathbf{e}_0||_\infty + ||\mathbf{e}_1||_\infty) \\ &\leq m \cdot s\sqrt{m} \cdot (B \cdot (2sm\sqrt{m})^{j_0} + B \cdot (2sm\sqrt{m})^{j_1}) \\ &\leq B \cdot (2sm\sqrt{m})^{\max(j_0,j_1)+1} \end{aligned}$$

  exactly as required. Here, the second inequality is because $||\mathbf{R}_{0,1}^{\text{tgt}}||_\infty \leq s\sqrt{m}$ by Lemma 3.1. This finishes our proof of correctness.

Key Indistinguishability. Recall that in $\mathsf{ReKeyGen}$, we given sampling $(\mathbf{R}_0, \mathbf{R}_1)$ satisfying $\mathbf{A}_0\mathbf{R}_0 + \mathbf{A}_1\mathbf{R}_1 = \mathbf{A}_{\text{tgt}}$. Key indistinguishability basically says that we obtain the same distribution whether we use a trapdoor for $\mathbf{A}_0$ or that for $\mathbf{A}_1$. Indeed, this follows directly from the following statement in [CHKP12, GPV08] (see also [CHK09, Theorem 3.4]): for every $(\mathbf{A}_0, \mathbf{T}_0)$, $(\mathbf{A}_1, \mathbf{T}_1)$ generated by $\mathsf{TrapSamp}(1^n, 1^m, q)$, every matrix $\mathbf{V} \in \mathbb{Z}_q^{n\times m}$, and any $s = \Omega(\sqrt{n \log q})$, the following two experiments generate distributions with $\text{negl}(n)$ statistical distance:

- Sample $\mathbf{R}_0 \leftarrow (D_{\mathbb{Z}^m,s})^m$, compute $\mathbf{U} := \mathbf{V} - \mathbf{A}_0\mathbf{R}_0 \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{R}_1 \leftarrow \mathsf{SampleD}(\mathbf{A}_1, \mathbf{T}_1, \mathbf{U}, s)$. Output $(\mathbf{R}_0, \mathbf{R}_1)$.
- Sample $\mathbf{R}_1 \leftarrow (D_{\mathbb{Z}^m,s})^m$, compute $\mathbf{U} := \mathbf{V} - \mathbf{A}_1\mathbf{R}_1 \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{R}_0 \leftarrow \mathsf{SampleD}(\mathbf{A}_0, \mathbf{T}_0, \mathbf{U}, s)$. Output $(\mathbf{R}_0, \mathbf{R}_1)$.

The recoding simulation property follows readily from Lemma 3.1, as is done in [CHKP12]. Correlated pseudorandomness directly from the decisional LWE assumption $\mathsf{dLWE}_{n,(\ell+1)\cdot m,q,\chi}$ where $q = n^{\Theta(d_{\max})}$.

6 Attribute-Based Encryption for Circuits

In this section, we show how to construct attribute-based encryption for circuits from any TOR scheme. Let TOR be the scheme consisting of algorithms $(\mathsf{Params}, \mathsf{Keygen}, \mathsf{Encode})$ with the "two-to-one" recoding mechanism $(\mathsf{Recode}, \mathsf{ReKeyGen}, \mathsf{SimReKeyGen})$ with input space $\mathcal{S}$. For every $d_{\max}$, let $d_{\max}$-TOR denote a secure "two-to-one" recoding scheme that is correct for $d_{\max}$ recoding levels.

Theorem 6.1. *For every ℓ and polynomial $d_{\max} = d_{\max}(\lambda)$, let $\mathcal{C}_{\ell,d_{\max}}$ denote a family of polynomial-size circuits of depth at most $d_{\max}$ that take ℓ bits of input. Assuming the existence of a $d_{\max}$-TOR scheme, there exists a selectively secure attribute-based encryption scheme $\mathcal{ABE}$ for $\mathcal{C}$.*

Combining Theorem 6.1 and Lemma 5.1, we obtain a selectively secure attribute-based encryption scheme from LWE. Furthermore, invoking an argument from [BB04, Theorem 7.1] and using subexponential hardness of LWE, we obtain a fully secure scheme:

Corollary 6.2. *For all ℓ and polynomial $d_{\max} = d_{\max}(\ell)$, there exists a selectively secure attribute-based encryption scheme $\mathcal{ABE}$ for any family of polynomial-size circuits with ℓ inputs and depth at most $d_{\max}$, assuming the hardness of $\mathsf{dLWE}_{n,q,\chi}$ for sufficiently large $n = \mathrm{poly}(\lambda, d_{\max})$, $q = n^{O(d_{\max})}$ and some $\mathrm{poly}(n)$-bounded error distribution χ.*

Moreover, assuming $2^{O(\ell)}$-hardness of $\mathsf{dLWE}_{n,q,\chi}$ for parameters $n = \mathrm{poly}(\lambda, d_{\max}, \ell)$, and q and χ as above, the attribute-based encryption scheme $\mathcal{ABE}$ is fully secure.

The reader is referred to the text after the construction for further explanation of how to choose the LWE parameters.

Observe that if we start with a TOR scheme that supports $d_{\max} = \ell^{\omega(1)}$, then our construction immediately yields an attribute-based encryption scheme for arbitrary polynomial-size circuit families (without any restriction on the depth). This can be achieved if, for example, we had an LWE-based TOR scheme where q grows polynomially instead of exponentially in $d_{\max}$ as in our LWE-based weak TOR.

We now prove Theorem 6.1.

Circuit Representation. Let $\mathcal{C}_\lambda$ be a collection of circuits each having $\ell = \ell(\lambda)$ input wires and one output wire. Define a collection $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$. For each $C \in \mathcal{C}_\lambda$, we index the wires of C in the following way. The input wires are indexed 1 to ℓ, the internal wires have indices $\ell+1, \ell+2, \ldots, |C|-1$ and the output wire has index $|C|$, which also denotes the size of the circuit. We assume that the circuit is composed of arbitrary two-to-one gates. Each gate g is indexed as a tuple (u, v, w) where u and v are the incoming wire indices, and $w > \max\{u, v\}$ is the outgoing wire index. The gate computes the function $g_w : \{0,1\} \times \{0,1\} \to \{0,1\}$. The "fan-out wires" in the circuit are given a single number. That is, if the outgoing wire of a gate feeds into the input of multiple gates, then all these wires are indexed the same. (See e.g. [BHR12, Fig 4].)

6.1 Construction from TOR

The ABE scheme $\mathcal{ABE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ is defined as follows.

$\mathsf{Setup}(1^\lambda, 1^\ell, d_{\max})$: For each of the ℓ input wires, generate two public/secret key pairs. Also, generate an additional public/secret key pair:

$$(\mathsf{pk}_{i,b}, \mathsf{sk}_{i,b}) \leftarrow \mathsf{Keygen}(\mathsf{pp}) \qquad \text{for } i \in [\ell], b \in \{0,1\}$$
$$(\mathsf{pk}_{\text{out}}, \mathsf{sk}_{\text{out}}) \leftarrow \mathsf{Keygen}(\mathsf{pp})$$

Output

$$\mathsf{mpk} := \begin{pmatrix} \mathsf{pk}_{1,0} & \mathsf{pk}_{2,0} & \dots & \mathsf{pk}_{\ell,0} & \\ \mathsf{pk}_{1,1} & \mathsf{pk}_{2,1} & \dots & \mathsf{pk}_{\ell,1} & \mathsf{pk}_{\text{out}} \end{pmatrix} \qquad \mathsf{msk} := \begin{pmatrix} \mathsf{sk}_{1,0} & \mathsf{sk}_{2,0} & \dots & \mathsf{sk}_{\ell,0} \\ \mathsf{sk}_{1,1} & \mathsf{sk}_{2,1} & \dots & \mathsf{sk}_{\ell,1} \end{pmatrix}$$

$\mathsf{Enc}(\mathsf{mpk}, \mathsf{ind}, m)$: For $\mathsf{ind} \in \{0,1\}^\ell$, choose a uniformly random $s \xleftarrow{\$} \mathcal{S}$ and encode it under the public keys specified by the index bits:

$$\psi_i \leftarrow \mathsf{Encode}(\mathsf{pk}_{i,\mathsf{ind}_i}, s) \text{ for all } i \in [\ell]$$

Encrypt the message m:

$$\tau \leftarrow \mathsf{E}(\mathsf{Encode}(\mathsf{pk}_{\text{out}}, s), m)$$

Output the ciphertext

$$\mathsf{ct}_{\mathsf{ind}} := \big(\ \psi_1,\quad \psi_2,\quad \dots,\quad \psi_\ell,\quad \tau\ \big)$$

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

1. For every non-input wire $w = \ell+1, \dots, |C|$ of the circuit C, and every $b \in \{0,1\}$, generate public/secret key pairs:

$$(\mathsf{pk}_{w,b}, \mathsf{sk}_{w,b}) \leftarrow \mathsf{Keygen}(\mathsf{pp}) \text{ if } w < |C| \text{ or } b = 0$$

and set $\mathsf{pk}_{|C|,1} := \mathsf{pk}_{\text{out}}$.

2. For the gate $g = (u, v, w)$ with outgoing wire w, compute the four recoding keys $\mathsf{rk}^w_{b,c}$ (for $b, c \in \{0,1\}$):

$$\mathsf{rk}^w_{b,c} \leftarrow \mathsf{ReKeyGen}\Big(\mathsf{pk}_{u,b}, \mathsf{pk}_{v,c}, \mathsf{sk}_{u,b}, \mathsf{pk}_{w,g_w(b,c)}\Big)$$

Output the secret key which is a collection of $4(|C| - \ell)$ recoding keys

$$\mathsf{sk}_C := \Big(\ \mathsf{rk}^w_{b,c}\ :\ w \in \big[\,\ell+1, |C|\,\big],\ b, c \in \{0,1\}\ \Big)$$

$\mathsf{Dec}(\mathsf{sk}_C, \mathsf{ct}_{\mathsf{ind}})$: We tacitly assume that $\mathsf{ct}_{\mathsf{ind}}$ contains the index ind. For $w = \ell+1, \ldots, |C|$, let $g = (u, v, w)$ denote the gate with outgoing wire w. Suppose wires u and v carry the values b^* and c^*, so that wire w carries the value $d^* := g_w(b^*, c^*)$. Compute

$$\psi_{w,d^*} \leftarrow \mathsf{Recode}\Big(\mathsf{rk}^w_{b^*,c^*}, \psi_{u,b^*}, \psi_{v,c^*}\Big)$$

If $C(\mathsf{ind}) = 1$, then we would have computed $\psi_{|C|,1}$. Output the message

$$m \leftarrow \mathsf{D}(\ \psi_{|C|,1}, \tau\ )$$

If $C(\mathsf{ind}) = 0$, output $\perp$.

LWE Parameters. Fix $\ell = \ell(\lambda)$ and $d_{\max} = d_{\max}(\ell)$, and suppose the $\mathsf{dLWE}_{n,m,q,\chi}$ assumption holds for $q = 2^{n^\epsilon}$ for some $0 < \epsilon < 1$. Then, in our LWE-based TOR, we will set:

$$n = \tilde{\Theta}(d_{\max}^{1/\epsilon}) \quad \text{and} \quad q = n^{\Theta(d_{\max})}$$

By Corollary 6.2, we get security under 2^{n^ϵ}-LWE.

6.2 Correctness

Lemma 6.3 (correctness). *Let $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ be family where each $\mathcal{C}_\lambda$ is a finite collection of polynomial-size circuits each of depth at most $d_{\max}$. Let* TOR *be a correct "two-to-one" recoding scheme for $d_{\max}$ levels. Then, the construction presented above is a correct attribute-based encryption scheme.*

Proof. Fix a circuit C of depth at most $d_{\max}$ and an input ind such that $C(\mathsf{ind}) = 1$. Informally, we rely on recoding correctness for $d_{\max}$ recodings to show that $w = 1, \ldots, |C|$, we have

$$\psi_{w,d^*} = \mathsf{Encode}(\mathsf{pk}_{w,d^*}, s),$$

where d^* is the value carried by the wire w and ψ_{w,d^*} is computed as in Dec. Formally, we proceed via induction on w to show that

$$\psi_{w,d^*} \in \Psi_{\mathsf{pk}_{w,d^*},s,j}.$$

where j is the depth of wire w. The base case $w = 1, \ldots, \ell$ follows immediately from correctness of Encode. For the inductive step, consider a wire w at depth j for some gate $g = (u, v, w)$ where $u, v < w$. By the induction hypothesis,

$$\psi_{u,b^*} \in \Psi_{\mathsf{pk}_{u,b^*},s,j_0}, \quad \psi_{u,c^*} \in \Psi_{\mathsf{pk}_{v,c^*},s,j_1}$$

where $j_0, j_1 < j$ denote the depths of wires u and v respectively. It follows immediately from the correctness of Recode that

$$\psi_{w,d^*} \in \Psi_{\mathsf{pk}_{w,d^*},s,\max(i_0,i_1)+1} \subseteq \Psi_{\mathsf{pk}_{w,d^*},s,j}$$

which completes the inductive proof. Since $C(\mathsf{ind}) = 1$ and $\mathsf{pk}_{|C|,1} = \mathsf{pk}_{\text{out}}$, we have $\psi_{|C|,1} \in \Psi_{\mathsf{pk}_{\text{out}},s,d_{\max}}$. Finally, by the correctness of (E, D), $\mathsf{D}(\psi_{|C|,1}, \tau) = m$. □

6.3 Security

Lemma 6.4 (selective security)**.** *For any adversary $\mathcal{A}$ against selective security of the attribute-based encryption scheme, there exist an adversary $\mathcal{B}$ against correlated pseudorandomness of TOR whose running time is essentially the same as that of $\mathcal{A}$, such that*

$$\mathsf{Adv}^{\mathrm{PE}}_{\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathrm{CP}}_{\mathcal{B}}(\lambda) + \mathrm{negl}(\lambda)$$

where $\mathrm{negl}(\lambda)$ *captures the statistical security terms in TOR.*

We begin by describing alternative algorithms, which would be useful later for constructing the adversary $\mathcal{B}$ for the correlated pseudorandomness security game.

Alternative algorithms. Fix the selective challenge ind. We get from the "outside" the challenge $\mathsf{pp}, (\mathsf{pk}_i, \psi_i)_{i\in[\ell+1]}$ for correlated pseudorandomness, The main challenge is to design an alternative algorithm KeyGen^* for answering secret key queries without knowing $\mathsf{sk}_{1,\mathsf{ind}_1}, \ldots, \mathsf{sk}_{\ell,\mathsf{ind}_\ell}$ or $\mathsf{sk}_{\mathrm{out}}$. The algorithm KeyGen^* will maintain the following invariant: on input C with $C(\mathsf{ind}) = 0$,

- for every non-output wire $w = 1, \ldots, |C| - 1$ carrying the value b^*, we will know $\mathsf{sk}_{w,1-b^*}$ but not sk_{w,b^*}.

Moreover, we do not know $\mathsf{sk}_{|C|,0}$ or $\mathsf{sk}_{|C|,1} = \mathsf{sk}_{\mathrm{out}}$.

$\mathsf{Setup}^*(\mathsf{ind}, 1^\lambda, 1^\ell, d_{\max})$: Let

$$\begin{aligned} (\mathsf{pk}_{i,1-\mathsf{ind}_i}, \mathsf{sk}_{i,1-\mathsf{ind}_i}) &\leftarrow \mathsf{Keygen}(\mathsf{pp}) \text{ for } i \in [\ell] \\ \mathsf{pk}_{\mathrm{out}} &:= \mathsf{pk}_{\ell+1} \\ \mathsf{pk}_{i,\mathsf{ind}_i} &:= \mathsf{pk}_i \text{ for } i \in [\ell] \end{aligned}$$

Output $\mathsf{mpk} = \begin{pmatrix} \mathsf{pk}_{1,0} & \mathsf{pk}_{2,0} & \ldots & \mathsf{pk}_{\ell,0} & \\ \mathsf{pk}_{1,1} & \mathsf{pk}_{2,1} & \ldots & \mathsf{pk}_{\ell,1} & \mathsf{pk}_{\mathrm{out}} \end{pmatrix}$

$\mathsf{Enc}^*(\mathsf{mpk}, \mathsf{ind}, m)$: Set $\tau \leftarrow \mathsf{E}(\psi_{\ell+1}, m)$ and output the ciphertext

$$\mathsf{ct}_{\mathsf{ind}} = \left(\ \psi_1,\quad \psi_2,\quad \ldots,\quad \psi_\ell,\quad \tau\ \right)$$

where $\psi_1, \ldots, \psi_{\ell+1}$ are provided in the challenge.

$\mathsf{KeyGen}^*(\mathsf{ind}, \mathsf{msk}, C)$: where $C(\mathsf{ind}) = 0$,

1. For each internal wire $w \in [\ell+1, |C| - 1]$ of the circuit C carrying the value b^* for input ind, generate public/secret key pairs:

$$(\mathsf{pk}_{w,1-b^*}, \mathsf{sk}_{w,1-b^*}) \leftarrow \mathsf{Keygen}(\mathsf{pp})$$

We will generate pk_{w,b^*} using $\mathsf{SimReKeyGen}$ as described next.

2. For $w = \ell + 1, \dots, |C|$, let $g = (u, v, w)$ denote the gate for which w is the outgoing wire. Suppose wires u and v carry the values b^* and c^*, so that wire w carries the value $d^* := g_w(b^*, c^*)$. By the invariant above, we know $\mathsf{sk}_{u,1-b^*}$ and $\mathsf{sk}_{v,1-c^*}$ but not sk_{u,b^*} and sk_{v,c^*}. We start by generating

$$(\mathsf{pk}_{w,d^*}, \mathsf{rk}^w_{b^*,c^*}) \leftarrow \mathsf{SimReKeyGen}(\mathsf{pk}_{u,b^*}, \mathsf{pk}_{v,c^*})$$

We generate the other three recoding keys using $\mathsf{ReKeyGen}$ as follows:

$$\begin{aligned} \mathsf{rk}^w_{1-b^*,c^*} &\leftarrow \mathsf{ReKeyGen}\big(\mathsf{pk}_{u,1-b^*}, \mathsf{pk}_{v,c^*}, \quad \mathsf{sk}_{u,1-b^*}, \mathsf{pk}_{w,g_w(1-b^*,c^*)}\big) \\ \mathsf{rk}^w_{b^*,1-c^*} &\leftarrow \mathsf{ReKeyGen}\big(\mathsf{pk}_{v,1-c^*}, \mathsf{pk}_{u,b^*}, \quad \mathsf{sk}_{v,1-c^*}, \mathsf{pk}_{w,g_w(b^*,1-c^*)}\big) \\ \mathsf{rk}^w_{1-b^*,1-c^*} &\leftarrow \mathsf{ReKeyGen}\big(\mathsf{pk}_{u,1-b^*}, \mathsf{pk}_{v,1-c^*}, \mathsf{sk}_{u,1-b^*}, \mathsf{pk}_{w,g_w(1-b^*,1-c^*)}\big) \end{aligned}$$

Note that $\mathsf{rk}^w_{1-b^*,c*}, \mathsf{rk}^w_{1-b^*,1-c*}$ are generated the same way in both KeyGen and KeyGen^* using $\mathsf{sk}_{u,1-b^*}$.

Output the secret key

$$\mathsf{sk}_C := \Big(\ \mathsf{rk}^w_{b,c} \ : \ w \in \big[\, \ell + 1, |C| \,\big], \ b, c \in \{0, 1\} \ \Big)$$

Informally, the recoding key $\mathsf{rk}^w_{b^*,1-c^*}$ looks the same as in Keygen because of key indistinguishability, and $\mathsf{rk}^w_{b^*,c^*}$ (together with the simulated pk_{w,d^*}) looks the same as in Keygen because of the recoding simulation property.

Game sequence. Next, consider the following sequence of games. We use $\mathsf{Adv}_0, \mathsf{Adv}_1, \dots$ to denote the advantage of the adversary $\mathcal{A}$ in Games 0, 1, etc. Game 0 is the real experiment.

Game i for $i = 1, 2, \dots, q$. As in Game 0, except the challenger answers the first $i - 1$ key queries using KeyGen^* and the remaining $q - i$ key queries using KeyGen. For the i'th key query C_i, we consider sub-Games $i.w$ as follows:

Game $i.w$, for $w = \ell + 1, \dots, |C_i|$. The challenger switches $(\mathsf{rk}^w_{b,c} : b, c \in \{0, 1\})$ from KeyGen to KeyGen^*. More precisely:

- First, we switch $(\mathsf{pk}_{w,d^*}, \mathsf{rk}^w_{b^*,c*})$ from KeyGen to KeyGen^*. This relies on recoding simulation.
- Next, we switch $\mathsf{rk}^w_{b^*,1-c*}$ from KeyGen to KeyGen^*. This relies on key indistinguishability, w.r.t. sk_{b^*} and sk_{1-c^*}.
- The other two keys $\mathsf{rk}^w_{1-b^*,c*}, \mathsf{rk}^w_{1-b^*,1-c*}$ are generated the same way in both KeyGen and KeyGen^*.

By key indistinguishability and recoding simulation, we have

$$|\mathsf{Adv}_{i,w} - \mathsf{Adv}_{i,w+1}| \leq \text{negl}(\lambda) \text{ for all } i, w$$

Note that in Game q, the challenger runs Setup^* and answers all key queries using KeyGen^* with the selective challenge ind and generates the challenge ciphertext using Enc.

Game $q+1$. Same as Game q, except the challenger generates the challenge ciphertext using Enc^* with $\psi_{\ell+1} = \mathsf{Encode}(\mathsf{pk}_{\ell+1}, s)$. Clearly,

$$\mathsf{Adv}_{q+1} = \mathsf{Adv}_q$$

Game $q+2$. Same as Game $q+1$, except $\psi_{\ell+1} \xleftarrow{\$} \mathcal{K}$. It is straight-forward to construct an adversary $\mathcal{B}$ such that

$$|\mathsf{Adv}_{q+1} - \mathsf{Adv}_{q+2}| \leq \mathsf{Adv}^{\mathrm{CP}}_{\mathcal{B}}(\lambda)$$

Finally, $\mathsf{Adv}_{q+2} \leq \mathrm{negl}(\lambda)$ by the one-time semantic security of (E, D). The lemma then follows readily.

7 Attribute-Based Encryption for Branching Programs

In this section, we present weak TOR and attribute-based encryption for branching programs, which capture the complexity class log-space. As noted in Section 2.2, we exploit the fact that in branching programs, the transition function depends on an input variable and the current state; this means that one of the two input encodings during recoding is always a "depth 0" encoding.

Branching programs. Recall that a branching program Γ is a directed acyclic graph in which every nonterminal node has exactly two outgoing edges labeled $(i, 0)$ and $(i, 1)$ for some $i \in [\ell]$. Moreover, there is a distinguished terminal accept node. Every input $x \in \{0,1\}^\ell$ naturally induces a subgraph Γ_x containing exactly those edges labeled (i, x_i). We say that Γ accepts x iff there is a path from the start node to the accept node in Γ_x. At the cost of possibly doubling the number of edges and vertices, we may assume that there is at most one edge connecting any two nodes in Γ.

7.1 Weak TOR

A weak "two-to-one" encoding (wTOR) scheme consists of the same algorithms as TOR, except that $\mathsf{Keygen}(\mathsf{pp}, j)$ takes an additional input $j \in \{0,1\}$. That is, Keygen may produce different distribution of public/secret key pairs depending on j. Moreover, in $\mathsf{ReKeyGen}$, the first public key is always generated using $\mathsf{Keygen}(\mathsf{pp}, 0)$ and the second using $\mathsf{Keygen}(\mathsf{pp}, 1)$; similarly, in Recode, the first encoding is always generated with respect to a public key from $\mathsf{Keygen}(\mathsf{pp}, 0)$ and the second from $\mathsf{Keygen}(\mathsf{pp}, 1)$. Similarly, the correctness and statistical security properties are relaxed.

Correctness. First, for every pk and $s \in \mathcal{S}$, there exists a family of sets $\Psi_{\mathsf{pk},s,j}, j = 0, 1, \ldots, d_{\max}$:

- $\Psi_{\mathsf{pk},s,1} \subseteq \cdots \subseteq \Psi_{\mathsf{pk},s,d_{\max}}$.
- for all $\psi, \psi' \in \Psi_{\mathsf{pk},s,d_{\max}}$ and all $m \in \mathcal{M}$,

$$\mathsf{D}(\psi', \mathsf{E}(\psi, m)) = m$$

Secondly, the correctness of recoding requires that for any triple of key pairs $(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1), (\mathsf{pk}_{\mathrm{tgt}}, \mathsf{sk}_{\mathrm{tgt}})$ respectively in the support of $\mathsf{Keygen}(\mathsf{pp}, 0), \mathsf{Keygen}(\mathsf{pp}, 1), \mathsf{Keygen}(\mathsf{pp}, 1)$ and any encodings $\psi_0 \in \mathsf{Encode}(\mathsf{pk}_0, s)$ and $\psi_1 \in \Psi_{\mathsf{pk}_1,s,j_1}$ where $0 < j_1$,

$$\mathsf{Recode}(\mathsf{rk}, \psi_0, \psi_1) \in \Psi_{\mathsf{pk}_{\mathrm{tgt}},s,j_1+1}$$

Statistical Security Properties. We require recoding simulation as before, but not key indistinguishability. However, we require the following additional property:

(Back-tracking) : For all $(\mathsf{pk}_0, \mathsf{sk}_0) \leftarrow \mathsf{Keygen}(\mathsf{pp}, 0)$ and all $(\mathsf{pk}_1, \mathsf{sk}_1), (\mathsf{pk}_{\mathrm{tgt}}, \mathsf{sk}_{\mathrm{tgt}}) \leftarrow \mathsf{Keygen}(\mathsf{pp}, 1)$, the following distributions are identical:

$$\mathsf{ReKeyGen}(\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{sk}_0, \mathsf{pk}_{\mathrm{tgt}}) \equiv -\mathsf{ReKeyGen}(\mathsf{pk}_0, \mathsf{pk}_{\mathrm{tgt}}, \mathsf{sk}_0, \mathsf{pk}_1)$$

Informally, this says that switching the order of pk_1 and $\mathsf{pk}_{\mathrm{tgt}}$ as inputs to $\mathsf{ReKeyGen}$ is the same as switching the "sign" of the output. In our instantiations, the output of $\mathsf{ReKeyGen}$ lies in a group, so negating the output simply refers to applying the group inverse operation.

Remark 7.1. *Due to the additional back-tracking property, it is not the case that a TOR implies a weak TOR. However, we are able to instantiate weak TOR under weaker and larger classes of assumptions than TOR.*

Computational Security Property. We define the advantage function $\mathsf{Adv}^{\mathrm{CP}}_{\mathcal{A}}(\lambda)$ (modified to account for the additional input to Keygen) to be the absolute value of:

$$\Pr\left[b = b' : \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda); s \leftarrow \mathcal{S}; \\ (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Keygen}(\mathsf{pp}, 0), \\ \psi_i \leftarrow \mathsf{Encode}(\mathsf{pk}_i, s), i = 1, \ldots, \ell; \\ (\mathsf{pk}_{\ell+1}, \mathsf{sk}_{\ell+1}) \leftarrow \mathsf{Keygen}(\mathsf{pp}, 1); \\ \psi'_0 \leftarrow \mathsf{Encode}(\mathsf{pk}_{\ell+1}, s); \\ b \xleftarrow{\$} \{0,1\}; \psi'_1 \xleftarrow{\$} \mathcal{K} \\ b' \leftarrow \mathcal{A}(\mathsf{pk}_1, \ldots, \mathsf{pk}_{\ell+1}, \psi_1, \ldots, \psi_\ell, \psi'_b) \end{array} \right] - \frac{1}{2}$$

and we require that for all PPT $\mathcal{A}$, the advantage function $\mathsf{Adv}^{\mathrm{CP}}_{\mathcal{A}}(\lambda)$ is a negligible function in λ.

7.2 Weak TOR from LWE

We provide an instantiation of weak TOR from LWE. The main advantage over our construction of TOR in Section 5 is that the dependency of q on $d_{\max}$ is linear in $d_{\max}$ instead of exponential. Therefore, if q is quasi-polynomial, we can handle any polynomial $d_{\max}$, as opposed to an a-prior bounded $d_{\max}$.

Lemma 7.1. *Assuming $\mathsf{dLWE}_{n,(\ell+2)m,q,\chi}$ where $q = O(d_{\max} n^3 \log n)$, there is a weak TOR scheme that is correct up to $d_{\max}$ levels.*

Note that the parameters here are better than in Lemma 5.1. The construction of weak TOR from learning with errors follows:

- $\mathsf{Params}(1^\lambda, d_{\max})$: First choose the LWE dimension $n = n(\lambda)$. Let the error distribution $\chi = \chi(n) = D_{\mathbb{Z},\sqrt{n}}$, the error bound $B = B(n) = O(n)$, the modulus $q = q(n) = d_{\max} \cdot O(n^3 \log n)$, the number of samples $m = m(n) = O(n \log q)$ and the Gaussian parameter $s = s(n) = O(\sqrt{n \log q})$. Output the global public parameters $\mathsf{pp} = (n, \chi, B, q, m, s)$. Define the domain $\mathcal{S}$ of the encoding scheme to be $\mathbb{Z}_q^n$.

- $\mathsf{Keygen}(\mathsf{pp}, j)$: Run the trapdoor generation algorithm $\mathsf{TrapGen}(1^n, 1^m, q)$ to obtain a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ together with the trapdoor $\mathbf{T}$. Output

$$\mathsf{pk} = \mathbf{A}; \quad \mathsf{sk} = \mathbf{T}.$$

- $\mathsf{Encode}(\mathbf{A}, \mathbf{s})$: Sample an error vector $\mathbf{e} \leftarrow \chi^m$ and output the encoding $\boldsymbol{\psi} := \mathbf{A}^T\mathbf{s} + \mathbf{e} \in \mathbb{Z}_q^n$.
- $\mathsf{ReKeyGen}(\mathbf{A}_0, \mathbf{A}_1, \mathbf{A}_{\mathrm{tgt}}, \mathbf{T})$: Outputs a low-norm matrix $\mathbf{R}$ such that $\mathbf{A}_0\mathbf{R} = \mathbf{A}_{\mathrm{tgt}} - \mathbf{A}_1$. In particular,

$$\mathbf{R} \leftarrow \mathsf{SampleD}(\mathbf{A}_0, \mathbf{T}_0, \mathbf{A}_{\mathrm{tgt}} - \mathbf{A}_1, s)$$

- $\mathsf{SimReKeyGen}(\mathbf{A}_0, \mathbf{A}_1)$: Sample a matrix $\mathbf{R} \leftarrow (D_{\mathbb{Z},s})^{m \times m}$ by sampling each entry from the discrete Gaussian distribution $D_{\mathbb{Z},s}$. Output

$$\mathsf{rk} := \mathbf{R}; \quad \mathbf{A}_{\mathrm{tgt}} := \mathbf{A}_0\mathbf{R} + \mathbf{A}_1$$

- $\mathsf{Recode}(\mathsf{rk}, \boldsymbol{\psi}_0, \boldsymbol{\psi}_1)$: Outputs $\mathsf{rk}^T\boldsymbol{\psi}_0 + \boldsymbol{\psi}_1$.

Correctness. We define the sets $\Psi_{\mathbf{A},\mathbf{s},j}$ for $\mathsf{pk} := \mathbf{A} \in \mathbb{Z}_q^{n \times m}$, $\mathbf{s} \in \mathbb{Z}_q^n$ and $j \in [1 \ldots d_{\max}]$ as follows:

$$\Psi_{\mathbf{A},\mathbf{s},j} = \left\{\mathbf{A}^T\mathbf{s} + \mathbf{e} : \ ||\mathbf{e}||_\infty \leq B \cdot j \cdot (sm\sqrt{m})\right\}$$

The analysis is similar to that in the previous section. In particular, we observe right away that

- $\Psi_{\mathbf{A},\mathbf{s},1} \subseteq \Psi_{\mathbf{A},\mathbf{s},1} \subseteq \ldots \subseteq \Psi_{\mathbf{A},\mathbf{s},d_{\max}}$.
- For any two encodings $\boldsymbol{\psi}, \boldsymbol{\psi}' \in \Psi_{\mathbf{A},\mathbf{s},d_{\max}}$ and $\boldsymbol{\mu} \in \{0,1\}^n$, $\mathsf{D}(\boldsymbol{\psi}', \mathsf{E}(\boldsymbol{\psi}, \boldsymbol{\mu})) = \boldsymbol{\mu}$, as long as

$$B \cdot d_{\max} \cdot (sm\sqrt{m}) \leq q/4.$$

- Consider two encodings $\mathbf{A}^T\mathbf{s} + \mathbf{e} \in \mathsf{Encode}(\mathbf{A}, \mathbf{s})$ and $\boldsymbol{\psi}_1 \in \Psi_{\mathbf{A}_1,\mathbf{s},j_1}$ for any $j_1 \in \mathbb{N}$. Then, $\boldsymbol{\psi}_0 = \mathbf{A}_0^T\mathbf{s} + \mathbf{e}_0$ and $\boldsymbol{\psi}_1 := \mathbf{A}_1^T\mathbf{s} + \mathbf{e}_1$ where $||\mathbf{e}_0||_\infty \leq B$ and $||\mathbf{e}_1||_\infty \leq j_1 \cdot B \cdot (sm\sqrt{m})$.
  Then, the recoded ciphertext $\boldsymbol{\psi}_{\mathrm{tgt}}$ is computed as follows:

$$\begin{aligned}\boldsymbol{\psi}_{\mathrm{tgt}} &:= \mathbf{R}^T\boldsymbol{\psi}_0 + \boldsymbol{\psi}_1 \\ &= \mathbf{R}^T(\mathbf{A}_0^T\mathbf{s} + \mathbf{e}_0) + (\mathbf{A}_1^T\mathbf{s} + \mathbf{e}_1) \\ &= \mathbf{A}_{\mathrm{tgt}}^T\mathbf{s} + \mathbf{e}_{\mathrm{tgt}}\end{aligned}$$

  where the last equation is because $\mathbf{A}_{\mathrm{tgt}} = \mathbf{A}_0\mathbf{R} + \mathbf{A}_1$ and we define $\mathbf{e}_{\mathrm{tgt}} := \mathbf{R}^T\mathbf{e}_0 + \mathbf{e}_1$. Thus,

$$\begin{aligned}||\mathbf{e}_{\mathrm{tgt}}||_\infty &\leq m \cdot ||\mathbf{R}||_\infty ||\mathbf{e}_0||_\infty + ||\mathbf{e}_1||_\infty \\ &\leq m \cdot s\sqrt{m} \cdot B + B \cdot j_1 \cdot (sm\sqrt{m}) \\ &= (j_1 + 1) \cdot B \cdot (sm\sqrt{m})\end{aligned}$$

  exactly as required. Here, the second inequality is because $||\mathbf{R}||_\infty \leq s\sqrt{m}$ by Lemma 3.1. This finishes our proof of correctness.

Security. Correlated pseudorandomness follows from $\mathsf{dLWE}_{n,(\ell+2)m,q,\chi}$ where $q = n \cdot d_{\max}$. Recoding simulation follows from Lemma 3.1 by an argument identical to the one for the construction of TOR in Section 5. For back-tracking, negation is simply the additive inverse over $\mathbb{Z}_q^m$.

7.3 Weak TOR from Bilinear Maps

We use asymmetric groups for maximal generality and for conceptual clarity. We consider cyclic groups G_1, G_2, G_T of prime order q and $e : G_1 \times G_2 \to G_T$ is a non-degenerate bilinear map. We require that the group operations in G and G_T as well the bilinear map e are computable in deterministic polynomial time with respect to λ. Let g_1, g_2 denote random generators of G_1, G_2 respectively. The DBDH Assumption says that, given $g_1, g_2, g_1^a, g_2^a, g_2^b$ and g_1^s, $e(g_1, g_2)^{abs}$ is pseudorandom.

- $\mathsf{Params}(1^\lambda, d_{\max})$: Outputs $\mathsf{pp} := (g_1, g_2, g_1^a, g_2^a)$.
- $\mathsf{Keygen}(\mathsf{pp}, j)$:
  - If $j = 0$, then samples $t \xleftarrow{\$} \mathbb{Z}_q$ and outputs
  $$(\mathsf{pk}, \mathsf{sk}) := ((g_1^{a/t}, g_2^{a/t}), t)$$
  - If $j \geq 1$, output $\mathsf{pk} \xleftarrow{\$} G_2$.
- $\mathsf{Encode}(\mathsf{pk}, s)$:
  - If $\mathsf{pk} = (g_1^{a/t}, g_2^{a/t}) \in G_1 \times G_2$, output $(g_1^{a/t})^s$
  - If $\mathsf{pk} \in G_2$, output $e(g_1^a, \mathsf{pk})^s$
- $\mathsf{Recode}(\mathsf{rk}, c_0, , c_1)$: Outputs $e(c_0, \mathsf{rk}) \cdot c_1$.
- $\mathsf{ReKeyGen}((g_1^{a/t}, g_2^{a/t}), \mathsf{pk}_1, \mathsf{pk}_{\mathrm{tgt}}, t)$: Outputs $\mathsf{rk} := (\mathsf{pk}_{\mathrm{tgt}} \cdot \mathsf{pk}_1^{-1})^t \in G_2$.
- $\mathsf{SimReKeyGen}((g_1^{a/t}, g_2^{a/t}), \mathsf{pk}_1)$: Picks $z \xleftarrow{\$} Z_q$ and outputs
$$\mathsf{rk} := (g_2^{a/t})^z, \quad \mathsf{pk}_{\mathrm{tgt}} := \mathsf{pk}_1 \cdot (g_2^a)^z$$

Correctness. Define $\Psi_{\mathsf{pk},s,j} := \{\mathsf{Encode}(\mathsf{pk}, s)\}$. For recoding, observe that:

$$\begin{aligned}
&\mathsf{Recode}((\mathsf{pk}_{\mathrm{tgt}} \cdot \mathsf{pk}_1^{-1})^t, g_1^{as/t}, e(g_1^a, \mathsf{pk}_1)^s \\
&= e(g_1^{as/t}, (\mathsf{pk}_{\mathrm{tgt}} \cdot \mathsf{pk}_1^{-1})^t) \cdot e(g_1^a, \mathsf{pk}_1)^s \\
&= e(g_1^a, (\mathsf{pk}_{\mathrm{tgt}} \cdot \mathsf{pk}_1^{-1})^s) \cdot e(g_1^a, \mathsf{pk}_1)^s \\
&= e(g_1^a, \mathsf{pk}_{\mathrm{tgt}})^s = \mathsf{Encode}(\mathsf{pk}_{\mathrm{tgt}}, s)
\end{aligned}$$

For back-tracking, negation is simply the multiplicative inverse over G_q.

Security. Correlation pseudorandomness follows readily from the DBDH assumption and its random self-reducibility.

7.4 Attribute-Based Encryption from weak TOR

$\mathsf{Setup}(1^\lambda, 1^\ell, d_{\max})$: For each one of ℓ input bits, generate two public/secret key pairs. Also, generate a public/secret key pair for the start and accept states:

$$\begin{aligned}(\mathsf{pk}_{i,b}, \mathsf{sk}_{i,b}) &\leftarrow \mathsf{Keygen}(\mathsf{pp}, 0) \quad \text{for } i \in [\ell], b \in \{0,1\}\\ (\mathsf{pk}_{\text{start}}, \mathsf{sk}_{\text{start}}) &\leftarrow \mathsf{Keygen}(\mathsf{pp}, 1)\\ (\mathsf{pk}_{\text{accept}}, \mathsf{sk}_{\text{accept}}) &\leftarrow \mathsf{Keygen}(\mathsf{pp}, 1)\end{aligned}$$

Output

$$\mathsf{mpk} := \begin{pmatrix} \mathsf{pk}_{1,0} & \mathsf{pk}_{2,0} & \dots & \mathsf{pk}_{\ell,0} & \mathsf{pk}_{\text{start}} \\ \mathsf{pk}_{1,1} & \mathsf{pk}_{2,1} & \dots & \mathsf{pk}_{\ell,1} & \mathsf{pk}_{\text{accept}} \end{pmatrix}$$

$$\mathsf{msk} := \begin{pmatrix} \mathsf{sk}_{1,0} & \mathsf{sk}_{2,0} & \dots & \mathsf{sk}_{\ell,0} & \mathsf{sk}_{\text{start}} \\ \mathsf{sk}_{1,1} & \mathsf{sk}_{2,1} & \dots & \mathsf{sk}_{\ell,1} & \mathsf{sk}_{\text{accept}} \end{pmatrix}$$

$\mathsf{Enc}(\mathsf{mpk}, \mathsf{ind}, m)$: For $\mathsf{ind} \in \{0,1\}^\ell$, choose a uniformly random $s \xleftarrow{\$} \mathcal{S}$ and encode it under the public keys specified by the index bits and the start state:

$$\begin{aligned}\psi_i &\leftarrow \mathsf{Encode}(\mathsf{pk}_{i,\mathsf{ind}_i}, s) \text{ for all } i \in [\ell]\\ \psi_{\text{start}} &\leftarrow \mathsf{Encode}(\mathsf{pk}_{\text{start}}, s)\end{aligned}$$

Encrypt the message:

$$\tau \leftarrow \mathsf{E}(\mathsf{Encode}(\mathsf{pk}_{\text{accept}}, s), m)$$

Output the ciphertext:

$$\mathsf{ct}_{\mathsf{ind}} = \begin{pmatrix} \psi_1, & \psi_2, & \dots, & \psi_\ell, & \psi_{\text{start}}, & \tau \end{pmatrix}$$

$\mathsf{KeyGen}(\mathsf{msk}, \Gamma)$: $\Gamma : \{0,1\}^\ell \to \{0,1\}$ is a branching program that takes a ℓ-bit input and outputs a single bit.

- For every node u, except the start and accept nodes, sample public/secret key pair:

$$(\mathsf{pk}_u, \mathsf{sk}_u) \leftarrow \mathsf{Keygen}(\mathsf{pp}, 1)$$

- For every edge (u, v) labeled (i, b) in Γ, sample a recoding key $\mathsf{rk}_{u,v}$ as follows:

$$\mathsf{rk}_{u,v} \leftarrow \mathsf{ReKeyGen}\Big(\mathsf{pk}_{i,b}, \mathsf{pk}_u, \mathsf{sk}_{i,b}, \mathsf{pk}_v\Big)$$

The secret key sk_Γ is the collection of all the recoding keys $\mathsf{rk}_{u,v}$ for every edge (u, v) in Γ.

$\mathsf{Dec}(\mathsf{sk}_\Gamma, \mathsf{ct}_{\mathsf{ind}})$: Suppose $\Gamma(\mathsf{ind}) = 1$; output $\perp$ otherwise. Let Π denote the (directed) path from the start node to the accept node in Γ_{ind}. For every edge (u, v) labeled (i, ind_i) in Π, apply the recoding algorithm on the two encodings ψ_i, ψ_u and the recoding key $\mathsf{rk}_{u,v}$:

$$\psi_v \leftarrow \mathsf{Recode}\Big(\mathsf{rk}_{u,v}, \psi_i, \psi_u\Big)$$

If $\Gamma(\mathsf{ind}) = 1$, we obtain ψ_{accept}. Decrypt and output the message:

$$m \leftarrow \mathsf{D}(\psi_{\text{accept}}, \tau)$$

7.4.1 Correctness

Lemma 7.2 (correctness)**.** *Let $\mathcal{G} = \{\Gamma\}_\lambda$ be a collection of polynomial-size branching programs of depth at most $d_{\max}$ and let* wTOR *be a weak "two-to-one" recoding scheme for $d_{\max}$ levels. Then, the construction presented above is a correct attribute-based encryption scheme for $\mathcal{G}$.*

Proof. Let Π denote the directed path from the start to the accept nodes in Γ_{ind}. We show via induction on nodes v along the path Π that

$$\psi_v \in \Psi_{\mathsf{pk}_v,s,j}$$

where j is the depth of node v along the path. The base case for $v :=$ start node follows immediately from correctness of Encode. For the inductive step, consider a node v along the path Π at depth j for some edge (u, v) labeled (i, ind_i). By the induction hypothesis,

$$\psi_u \in \Psi_{\mathsf{pk}_u,s,j_0}$$

where $j_0 < j$ denote the depths of node u. Also by the correctness of the Encode algorithm, for all $i \in [\ell]$

$$\psi_i \in \Psi_{\mathsf{pk}_{i,\mathsf{ind}_i},s,0}$$

It follows immediately from the correctness of Recode that

$$\psi_v \in \Psi_{\mathsf{pk}_v,s,j_0+1} \subseteq \Psi_{\mathsf{pk}_v,s,j}$$

which completes the inductive proof. Since $C(\mathsf{ind}) = 1$, we have

$$\psi_{\text{accept}} \in \Psi_{\mathsf{pk}_{\text{accept}},s,d_{\max}}$$

Recall that $\tau \leftarrow \mathsf{E}(\mathsf{Encode}(\mathsf{pk}_{\text{accept}}, s), m)$. Finally, by the correctness of (E, D),

$$\mathsf{D}(\psi_{\text{accept}}, \tau) = m$$ □

7.4.2 Selective Security

Lemma 7.3 (selective security)**.** *For any adversary $\mathcal{A}$ against selective security of the attribute-based encryption scheme for branching programs, there exist an adversary $\mathcal{B}$ against correlated pseudorandomness of wTOR whose running time is essentially the same as that of $\mathcal{A}$, such that*

$$\mathsf{Adv}^{\text{PE}}_{\mathcal{A}}(\lambda) \le \mathsf{Adv}^{\text{CP}}_{\mathcal{B}}(\lambda) + \text{negl}(\lambda)$$

where $\text{negl}(\lambda)$ *captures the statistical security terms in TOR.*

In the proof of security, we will rely crucially on the following combinatorial property of branching programs: for any input x, the graph Γ_x does not contain any cycles as an *undirected* graph.

Alternative algorithms. Fix the selective challenge ind. We also get a collection of public keys, corresponding encodings from the "outside": $(\mathsf{pk}_i, \psi_i)_{i\in[\ell+2]}$, where the challenge is to decide whether $\psi_{\ell+1}$ is $\mathsf{Encode}(\mathsf{pk}_{\ell+2}, s)$ or random. The main challenge is design an alternative algorithm KeyGen^* for answering secret key queries without knowing $\mathsf{sk}_{1,\mathsf{ind}_1}, \ldots, \mathsf{sk}_{\ell,\mathsf{ind}_\ell}$ or $\mathsf{sk}_{\text{start}}, \mathsf{sk}_{\text{accept}}$. We consider the following "alternative" algorithms.

$\mathsf{Setup}^*(1^\lambda, 1^\ell, d_{\max})$: Let

$$\begin{array}{rcl} (\mathsf{pk}_{i,1-\mathsf{ind}_i}, \mathsf{sk}_{i,1-\mathsf{ind}_i}) & \leftarrow & \mathsf{Keygen}(\mathsf{pp}, 0) \text{ for } i \in [\ell] \\ \mathsf{pk}_{i,\mathsf{ind}_i} & := & \mathsf{pk}_i \text{ for } i \in [\ell] \\ \mathsf{pk}_{\text{start}} & := & \mathsf{pk}_{\ell+1} \\ \mathsf{pk}_{\text{accept}} & := & \mathsf{pk}_{\ell+2} \end{array}$$

Define and output the master public key as follows:

$$\mathsf{mpk} = \begin{pmatrix} \mathsf{pk}_{1,0} & \mathsf{pk}_{2,0} & \cdots & \mathsf{pk}_{\ell,0} & \mathsf{pk}_{\text{start}} \\ \mathsf{pk}_{1,1} & \mathsf{pk}_{2,1} & \cdots & \mathsf{pk}_{\ell,1} & \mathsf{pk}_{\text{accept}} \end{pmatrix}$$

$\mathsf{Enc}^*(\mathsf{mpk}, \mathsf{ind}, m)$: Define

$$\begin{array}{rcl} \psi_{i,\mathsf{ind}_i} & := & \psi_i \quad \text{for all } i \in [\ell] \\ \psi_{\text{start}} & := & \psi_{\ell+1} \\ \psi_{\text{accept}} & := & \psi_{\ell+2} \end{array}$$

Encrypt the message m:

$$\tau \leftarrow \mathsf{E}(\psi_{\text{accept}}, b)$$

Output the simulated ciphertext

$$\mathsf{ct}_{\mathsf{ind}} = \begin{pmatrix} \psi_1, & \psi_2, & \ldots, & \psi_\ell, & \psi_{\text{start}}, & \tau \end{pmatrix}$$

$\mathsf{KeyGen}^*(\mathsf{msk}, \Gamma)$: Let Γ'_{ind} denote the *undirected* graph obtained from Γ_{ind} by treating every directed edge as an undirected edge (while keeping the edge label). Observe that Γ'_{ind} satisfies the following properties:

- Γ'_{ind} contains no cycles. This is because Γ_{ind} is acyclic and every nonterminal node contains exactly one outgoing edge.
- The start node and the accept node lie in different connected components in Γ'_{ind}, since $\Gamma(\mathsf{ind}) = 0$.

Simulation invariant: for each "active" edge labeled (i, ind_i) from node u to node v, simulate the recoding key. Choose our own public/secret key pair for each "inactive" edges $(i, 1 - \mathsf{ind}_i)$ and generate the recoding key honestly.

- Run a DFS in Γ'_{ind} starting from the start node. Whenever we visit a new node v from a node u along an edge labeled (i, ind_i), we set:

$$\begin{array}{lcll} (\mathsf{pk}_v, \mathsf{rk}_{u,v}) & \leftarrow & \mathsf{SimReKeyGen}\big(\mathsf{pk}_{i,\mathsf{ind}}, \mathsf{pk}_u\big) & \text{if } (u,v) \text{ is a directed edge in } \Gamma \\ (\mathsf{pk}_v, -\mathsf{rk}_{v,u}) & \leftarrow & \mathsf{SimReKeyGen}\big(\mathsf{pk}_{i,\mathsf{ind}}, \mathsf{pk}_u\big) & \text{if } (v,u) \text{ is a directed edge in } \Gamma \end{array}$$

  Here, we exploit the back-tracking property in wTOR.

  Note that since $\Gamma(\mathsf{ind}) = 0$, then the accept node is not assigned a public key by this process.
- For all nodes u without an assignment, run $(\mathsf{pk}_u, \mathsf{sk}_u) \leftarrow \mathsf{Keygen}(\mathsf{pp}, 1)$.
- For every remaining edge (u,v) labeled $(i, 1 - \mathsf{ind}_i)$ in Γ, sample a recoding key $\mathsf{rk}_{u,v}$ as in KeyGen using $\mathsf{sk}_{i,1-\mathsf{ind}}$ as follows:

$$\mathsf{rk}_{u,v} \leftarrow \mathsf{ReKeyGen}\Big(\mathsf{pk}_{i,1-\mathsf{ind}}, \mathsf{pk}_u, \mathsf{sk}_{i,1-\mathsf{ind}}, \mathsf{pk}_v\Big)$$

The secret key sk_Γ is simply the collection of all the recoding keys $\mathsf{rk}_{u,v}$ for every edge (u,v) in Γ.

Game sequence. Next, consider the following sequence of games. We use $\mathsf{Adv}_0, \mathsf{Adv}_1, \ldots$ to denote the advantage of the adversary $\mathcal{A}$ in Games 0, 1, etc. Let n denote the number of edges in a branching program Γ labeled (i, ind_i) for some i, and for all $j \in [n]$ let e_j denote the actual edge.

Game 0. Real experiment.

Game i for $i = 1, 2, \ldots, q$. As in Game 0, except the challenger answers the first $i-1$ key queries using KeyGen^* and the remaining $q - i$ key queries using KeyGen. For the i'th key query Γ_i, we consider sub-Games $i.e$ as follows:

Game $i.j$, for $j = 1, \ldots, n$. For edge $e_j = (u,v)$ labeled (i, ind_i), the challenger switches the simulated recoding key $\mathsf{rk}_{u,v}$ from KeyGen to KeyGen^*. We rely on recoding simulation and back-tracking properties simultaneously.

By recoding simulation and back-tracking, we have:

$$|\mathsf{Adv}_{i,e} - \mathsf{Adv}_{i,e+1}| \le \text{negl}(\lambda) \text{ for all } i, e$$

Note that in Game q, the challenger runs Setup^* and answers all key queries using KeyGen^* with the selective challenge ind and generates the challenge ciphertext using Enc.

Game $q+1$. Same as Game q, except the challenger generates the challenge ciphertext using Enc^* with $\psi_{\ell+2} \leftarrow \mathsf{Encode}(\mathsf{pk}_{\ell+2}, s)$.

$$\mathsf{Adv}_{q+1} = \mathsf{Adv}_q$$

Game $q+2$. Same as Game $q+1$, except $\psi_{\ell+2} \xleftarrow{\$} \mathcal{K}$. It is straight-forward to construct an adversary $\mathcal{B}$ such that

$$|\mathsf{Adv}_{q+1} - \mathsf{Adv}_{q+2}| \le \mathsf{Adv}^{\text{CP}}_{\mathcal{B}}(\lambda)$$

Finally, $\mathsf{Adv}_{q+2} \le \text{negl}(\lambda)$ by the one-time semantic security of (E, D). The lemma then follows readily.

Acknowledgments.

We thank Dan Boneh and Shafi Goldwasser for helpful comments and insightful conversations.

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. Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE. In *CRYPTO*, pages 98–115, 2010.

[ABV$^+$12] Shweta Agrawal, Xavier Boyen, Vinod Vaikuntanathan, Panagiotis Voulgaris, and Hoeteck Wee. Functional encryption for threshold functions (or, fuzzy IBE) from lattices. In *Public Key Cryptography*, pages 280–297, 2012.

[AFGH06] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Trans. Inf. Syst. Secur.*, 9(1):1–30, 2006.

[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.

[AIK10] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. From secrecy to soundness: Efficient verification via secure computation. In *ICALP (1)*, pages 152–163, 2010.

[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.

[BB04] Dan Boneh and Xavier Boyen. Efficient selective-id secure identity-based encryption without random oracles. In *EUROCRYPT*, pages 223–238, 2004.

[BBS98] Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In *EUROCRYPT*, pages 127–144, 1998.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In *CRYPTO*, pages 213–229, 2001.

[BHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In *ACM CCS*, pages 784–796, 2012.

[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.

[CHK09] David Cash, Dennis Hofheinz, and Eike Kiltz. How to delegate a lattice basis. Cryptology ePrint Archive, Report 2009/351, 2009.

[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.

[CKV10] Kai-Min Chung, Yael Kalai, and Salil P. Vadhan. Improved delegation of computation using fully homomorphic encryption. In *CRYPTO*, pages 483–501, 2010.

[Coc01] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *IMA Int. Conf.*, pages 360–363, 2001.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.

[GGH12] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices and applications. Cryptology ePrint Archive, Report 2012/610, 2012.

[GGH$^+$13] Sanjam Garg, Craig Gentry, Shai Halevi, Amit Sahai, and Brent Waters. Attribute-based encryption for circuits from multilinear maps. Cryptology ePrint Archive, Report 2013/128, 2013.

[GGP10] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *CRYPTO*, pages 465–482, 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$^+$13] Shafi Goldwasser, Yael Kalai, Raluca Ada Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Succinct functional encryption and its power: Reusable garbled circuits and beyond. In *STOC*, 2013. To appear.

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

[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.

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

[HRSV11] Susan Hohenberger, Guy N. Rothblum, Abhi Shelat, and Vinod Vaikuntanathan. Securely obfuscating re-encryption. *J. Cryptology*, 24(4):694–719, 2011.

[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.

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

[Mic00] Silvio Micali. Computationally sound proofs. *SIAM J. Comput.*, 30(4):1253–1298, 2000.

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

[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.

[O'N10] Adam O'Neill. Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556, 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.

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

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

[RS10] Alon Rosen and Gil Segev. Chosen-ciphertext security via correlated products. *SIAM J. Comput.*, 39(7):3058–3088, 2010.

[Sha84] Adi Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO*, pages 47–53, 1984.

[SS10a] Amit Sahai and Hakan Seyalioglu. Worry-free encryption: functional encryption with public keys. In *ACM CCS*, pages 463–472, 2010.

[SS10b] Damien Stehlé and Ron Steinfeld. Faster fully homomorphic encryption. In *ASIACRYPT*, pages 377–394, 2010.

[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.

[Wat12] Brent Waters. Functional encryption for regular languages. In *CRYPTO*, pages 218–235, 2012.

A Extensions

A.1 Outsourcing Decryption

In this section we show how to modify our main construction of attribute-based encryption to support outsourcing of decryption circuits, similar to [GHW11]. We require that the Keygen algorithm returns two keys:

- the evaluation key ek_C, that is given to a computationally powerful proxy,
- and a decryption key dk, given to the client.

Given a ciphertext $\mathsf{ct}_{\mathsf{ind}}$, the proxy must perform the "bulk" of the computation and return a new ciphertext $\mathsf{ct}'_{\mathsf{ind}}$ that is forwarded to the client. Using the decryption key dk, the client can decrypt and learn the message m iff the predicate $C(\mathsf{ind})$ is satisfied. We emphasize that that amount of computation the client needs to perform to decrypt the message must be independent on the circuit size. Intuitively, the security ensures that an adversary should learn nothing about the message, conditioned on that it queries for decryption keys dk's for predicates that are not satisfied by the challenge index (note, the adversary *can* query for evaluation keys separately for predicates that are satisfied).

Intuitively, we modify the main construction as follows. As before, the key-generation algorithm assigns two keys for each circuit wire. The evaluation key consists of all the recoding keys for the circuit. In addition, the output wire has another key $\mathsf{pk}_{\mathrm{out}}$ which now plays a special role. The recoding key from $\mathsf{pk}_{|C|,1}$ to $\mathsf{pk}_{\mathrm{out}}$ is only given to the client as the decryption key. If $C(\mathsf{ind}) = 1$, the the proxy computes an encoding under the $\mathsf{pk}_{|C|,1}$ and forwards it to the client. The client applies the transformation, and decrypts the message. For technical reasons, since we are using "two-to-one" recoding mechanism, we need to introduce an auxiliary public key $\mathsf{pk}_{\mathrm{in}}$ and a corresponding encoding.

$\mathsf{Setup}(1^\lambda, 1^\ell, d_{\max})$: For each of the ℓ input wires, generate two public/secret key pairs. Also, generate an additional public/secret key pair:

$$\begin{aligned} (\mathsf{pk}_{i,b}, \mathsf{sk}_{i,b}) &\leftarrow \mathsf{Keygen}(\mathsf{pp}) \qquad \text{for } i \in [\ell], b \in \{0,1\} \\ (\mathsf{pk}_{\mathrm{out}}, \mathsf{sk}_{\mathrm{out}}) &\leftarrow \mathsf{Keygen}(\mathsf{pp}) \\ (\mathsf{pk}_{\mathrm{in}}, \mathsf{sk}_{\mathrm{in}}) &\leftarrow \mathsf{Keygen}(\mathsf{pp}) \end{aligned}$$

Output

$$\mathsf{mpk} := \begin{pmatrix} \mathsf{pk}_{1,0} & \mathsf{pk}_{2,0} & \dots & \mathsf{pk}_{\ell,0} & \mathsf{pk}_{\mathrm{in}} \\ \mathsf{pk}_{1,1} & \mathsf{pk}_{2,1} & \dots & \mathsf{pk}_{\ell,1} & \mathsf{pk}_{\mathrm{out}} \end{pmatrix} \qquad \mathsf{msk} := \begin{pmatrix} \mathsf{sk}_{1,0} & \mathsf{sk}_{2,0} & \dots & \mathsf{sk}_{\ell,0} & \mathsf{sk}_{\mathrm{in}} \\ \mathsf{sk}_{1,1} & \mathsf{sk}_{2,1} & \dots & \mathsf{sk}_{\ell,1} & \mathsf{sk}_{\mathrm{out}} \end{pmatrix}$$

$\mathsf{Enc}(\mathsf{mpk}, \mathsf{ind}, m)$: For $\mathsf{ind} \in \{0,1\}^\ell$, choose a uniformly random $s \xleftarrow{\$} \mathcal{S}$ and encode it under the public keys specified by the index bits:

$$\psi_i \leftarrow \mathsf{Encode}(\mathsf{pk}_{i,\mathsf{ind}_i}, s) \text{ for all } i \in [\ell]$$

Encode s under the input public key:

$$\psi_{\text{in}} \leftarrow \mathsf{Encode}(\mathsf{pk}_{\text{in}}, s)$$

Encrypt the message m:

$$\tau \leftarrow \mathsf{E}(\mathsf{Encode}(\mathsf{pk}_{\text{out}}, s), m)$$

Output the ciphertext

$$\mathsf{ct}_{\mathsf{ind}} := (\ \psi_1,\quad \psi_2,\quad \ldots,\quad \psi_\ell,\quad \psi_{\text{in}},\quad \tau\ )$$

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

1. For every non-input wire $w = \ell + 1, \ldots, |C|$ of the circuit C, and every $b \in \{0,1\}$, generate public/secret key pairs:
$$(\mathsf{pk}_{w,b}, \mathsf{sk}_{w,b}) \leftarrow \mathsf{Keygen}(\mathsf{pp})$$
2. For the gate $g = (u, v, w)$ with output wire w, compute the four recoding keys $\mathsf{rk}^w_{b,c}$ (for $b, c \in \{0,1\}$):
$$\mathsf{rk}^w_{b,c} \leftarrow \mathsf{ReKeyGen}\Big(\mathsf{pk}_{u,b}, \mathsf{pk}_{v,c}, \mathsf{sk}_{u,b}, \mathsf{pk}_{w,g_w(b,c)}\Big)$$
3. Also, compute the recoding key
$$\mathsf{rk}^{\text{out}} \leftarrow \mathsf{ReKeyGen}\Big(\mathsf{pk}_{|C|,1}, \mathsf{pk}_{\text{in}}, \mathsf{sk}_{|C|,1}, \mathsf{pk}_{\text{out}}\Big)$$

Output the evaluation key which is a collection of $4(|C| - \ell)$ recoding keys

$$\mathsf{ek}_C := \Big(\ \mathsf{rk}^w_{b,c}\ :\ w \in \big[\,\ell + 1, |C|\,\big],\ b, c \in \{0,1\}\ \Big)$$

and the decryption key $\mathsf{dk} := \mathsf{rk}^{\text{out}}$.

$\mathsf{Eval}(\mathsf{ek}_C, \mathsf{ct}_{\mathsf{ind}})$: We tacitly assume that $\mathsf{ct}_{\mathsf{ind}}$ contains the index ind. For $w = \ell + 1, \ldots, |C|$, let $g = (u, v, w)$ denote the gate with output wire w. Suppose wires u and v carry the values b^* and c^*, so that wire w carries the value $d^* := g_w(b^*, c^*)$. Compute

$$\psi_{w,d^*} \leftarrow \mathsf{Recode}\Big(\mathsf{rk}^w_{b^*,c^*}, \psi_{u,b^*}, \psi_{v,c^*}\Big)$$

If $C(\mathsf{ind}) = 1$, then we would have computed $\psi_{|C|,1}$. Output

$$\mathsf{ct}'_{\mathsf{ind}} := (\psi_{|C|,1}, \psi_{\text{in}}, \tau)$$

If $C(\mathsf{ind}) = 0$, output $\perp$.

$\mathsf{Dec}(\mathsf{dk}, \mathsf{ct}'_{\mathsf{ind}})$: Apply the transformation

$$\psi_{\text{out}} \leftarrow \mathsf{Recode}\Big(\mathsf{rk}^{\text{out}}, \psi_{\text{in}}, \psi_{|C|,1}\Big)$$

and output the message

$$m \leftarrow \mathsf{D}(\ \psi_{\text{out}}, \tau\ )$$

Security. We informally state how to modify the simulator in the proof of security in Section-6.4. The simulator gets $\{\mathsf{pk}_i, \psi_i\}_{i\in[\ell+2]}$ from the "outside". It assigns $\mathsf{pk}_1, \ldots, \mathsf{pk}_\ell$ as the public keys specified by the bits of ind and $\mathsf{pk}_{\text{in}} := \mathsf{pk}_{\ell+1}, \mathsf{pk}_{\text{out}} := \mathsf{pk}_{\ell+2}$. It is easy to see how to simulate the ciphertext: all the input encodings become a part of it, as well as an encryption of the message using $\psi_{\text{out}} := \psi_{\ell+2}$. Now, the evaluation key ek is simulated by applying the TOR simulator.

- For query C such that $C(\mathsf{ind}) = 0$, the simulator can choose $(\mathsf{pk}_{|C|,1}, \mathsf{sk}_{|C|,1})$ by itself (the public key $\mathsf{pk}_{|C|,0}$ is "fixed" by the TOR simulator). Hence, the decryption key dk can be computed using $\mathsf{sk}_{|C|,1}$.
- On the other hand, for query C such that $C(\mathsf{ind}) = 1$, the adversary is not allowed to obtain the decryption key dk, hence there is not need to simulate it.

A.2 Extending Secret Keys

Consider the following problem: a users holds two (or more) secret keys sk_{C_1} and sk_{C_2}. C_1 allows to decrypt all ciphertexts addressed to *human resources* department and C_2 allows to decrypt ciphertexts addressed to *share holders*. The user wishes to create (and delegate) another secret key sk_{C^*} that allows to decrypt ciphertexts addressed to *human resources* and *share holders*. The question that we study is whether it is possible to allow the user to compute sk_{C^*} without calling the authority holding the master secret key msk. More formally, given $\{\mathsf{sk}_{C_i}\}_{i\in[q]}$ a users should be able to compute a secret key sk_{C^*} for any circuit C^* that is an black-box monotone composition of C_i's. Note that only monotone compositions are realizable, since otherwise a users holding a secret keys sk_{C_1} where $C_1(\mathsf{ind}) = 0$ could come up with a secret key for $\overline{C_1}$ and hence break any notion of security.

To suppose monotone extensions, it is enough to show how to obtain (1) $\mathsf{sk}_{C_1 \text{ AND } C_2}$ given $\mathsf{sk}_{C_1}, \mathsf{sk}_{C_2}$, and (2) $\mathsf{sk}_{C_1 \text{ OR } C_2}$ given $\mathsf{sk}_{C_1}, \mathsf{sk}_{C_2}$. We start from the construction presented in Section-A.1. We note that the security of that construction does not break if we give the secret key associated with the output value 0 ($\mathsf{sk}_{|C_i|,1}$) as a part of the secret key sk_{C_i}. This is because our simulation proceeds by sampling $(\mathsf{pk}_{|C_i|,1}, \mathsf{sk}_{|C_i|,1})$ honestly using Keygen algorithm and the fact the adversary is restricted to quires C_i such that $C_i(\mathsf{ind}) = 0$. Hence, given $\mathsf{sk}_{|C_1|,1}$ and $\mathsf{sk}_{|C_2|,1}$, let $C^* = C_1$ AND C_2. The user computes sk_{C^*} as $(\mathsf{ek}_{C_1}, \mathsf{ek}_{C_2})$ plus four recoding keys $\mathsf{rk}^{C^*}_{b,c}$ (for $b, c \in \{0,1\}$):

$$\begin{aligned}
(\mathsf{pk}_{|C^*|,0}, \mathsf{rk}^{C^*}_{0,0}) &\leftarrow \mathsf{SimReKeyGen}(\mathsf{pk}_{|C_1|,0}, \mathsf{pk}_{|C_2|,0}) \\
\mathsf{rk}^{C^*}_{0,1} &\leftarrow \mathsf{ReKeyGen}\Big(\mathsf{pk}_{|C_1|,0}, \mathsf{pk}_{|C_2|,1}, \mathsf{sk}_{|C_2|,1}, \mathsf{pk}_{|C^*|,0}\Big) \\
\mathsf{rk}^{C^*}_{1,0} &\leftarrow \mathsf{ReKeyGen}\Big(\mathsf{pk}_{|C_1|,1}, \mathsf{pk}_{|C_2|,0}, \mathsf{sk}_{|C_1|,1}, \mathsf{pk}_{|C^*|,0}\Big) \\
\mathsf{rk}^{C^*}_{1,1} &\leftarrow \mathsf{ReKeyGen}\Big(\mathsf{pk}_{|C_1|,1}, \mathsf{pk}_{|C_2|,1}, \mathsf{sk}_{|C_1|,1}, \mathsf{pk}_{\text{out}}\Big)
\end{aligned}$$

As before, the message is encrypted under the encoding $\psi_{\text{out}} \leftarrow \mathsf{Encode}(\mathsf{pk}_{\text{out}}, s)$. The construction extends similarly to support OR compositions. Furthermore, arbitrary monotone structures can be realized by sampling keys associated with value 1 $(\mathsf{pk}_1, \mathsf{sk}_1)$ honestly and computing the recoding keys as above, until the final wire is assigned to pk_{out}.