Predicate Encryption for Circuits from LWE

Sergey Gorbunov* MIT
Vinod Vaikuntanathan† MIT
Hoeteck Wee‡ ENS

Abstract

In predicate encryption, a ciphertext is associated with descriptive attribute values x in addition to a plaintext μ, and a secret key is associated with a predicate f. Decryption returns plaintext μ if and only if $f(x) = 1$. Moreover, security of predicate encryption guarantees that an adversary learns nothing about the attribute x or the plaintext μ from a ciphertext, given arbitrary many secret keys that are not authorized to decrypt the ciphertext individually.

We construct a leveled predicate encryption scheme for all circuits, assuming the hardness of the subexponential learning with errors (LWE) problem. That is, for any polynomial function $d = d(\lambda)$, we construct a predicate encryption scheme for the class of all circuits with depth bounded by $d(\lambda)$, where λ is the security parameter.

*Email: sergeyg@mit.edu. Supported in part by the Northrop Grumman Cybersecurity Research Consortium (CRC) and by a Microsoft PhD Fellowship.

†Email: vinodv@csail.mit.edu. Research supported in part by DARPA Grant number FA8750- 11-2-0225, NSF Awards CNS-1350619 and CNS-1413920, an Alfred P. Sloan Research Fellowship, the Northrop Grumman Cybersecurity Research Consortium (CRC), Microsoft Faculty Fellowship, and a Steven and Renee Finn Career Development Chair from MIT.

‡E-mail: wee@di.ens.fr. CNRS, INRIA and Columbia University. Supported in part by ANR-14-CE28-0003 (EnBiD), NSF Award CNS-1445424, ERC Project CryptoCloud (FP7/2007-2013 Grant Agreement no. 339563), the Alexander von Humboldt Foundation and a Google Faculty Research Award.

1 Introduction

Predicate encryption [BW07, SBC+07, KSW08] is a new paradigm for public-key encryption that supports searching on encrypted data. In predicate encryption, ciphertexts are associated with descriptive attribute values x in addition to plaintexts μ, secret keys are associated with a predicate f, and a secret key decrypts the ciphertext to recover μ if and only if $f(x) = 1$. The security requirement for predicate encryption enforces privacy of x and the plaintext even amidst multiple secret key queries: an adversary holding secret keys for different query predicates learns nothing about the attribute x and the plaintext if none of them is individually authorized to decrypt the ciphertext.

Motivating applications. We begin with several motivating applications for predicate encryption [BW07, SBC+07]:

- For inspecting recorded log files for network intrusions, we would encrypt network flows labeled with a set of attributes from the network header, such as the source and destination addresses, port numbers, time-stamp, and protocol numbers. We could then issue auditors with restricted secret keys that can only decrypt the network flows that fall within a particular range of IP addresses and some specific time period.
- For credit card fraud investigation, we would encrypt credit card transactions labeled with a set of attributes such as time, costs and zipcodes. We could then issue investigators with restricted secret keys that decrypt transactions over $1,000 which took place in the last month and originated from a particular range of zipcodes.
- For anti-terrorism investigation, we would encrypt travel records labeled with a set of attributes such as travel destination and basic traveller data. We could then issue investigators with restricted secret keys that match certain suspicious travel patterns.
- For online dating, we would encrypt personal profiles labeled with dating preferences pertaining to age, height, weight, salary and hobbies. Secret keys are associated with specific attributes and can only decrypt profiles for which the attributes match the dating preferences.

In all of these examples, it is important that unauthorized parties do not learn the contents of the ciphertexts, nor of the meta-data associated with the ciphertexts, such as the network header or dating preferences. On the other hand, it is often okay to leak the meta-data to authorized parties. We stress that privacy of the meta-data is an additional security requirement provided by predicate encryption but not by the related and weaker notion of attribute-based encryption (ABE) [SW05, GPSW06]; the latter only guarantees the privacy of the plaintext μ and not the attribute x.

Utility and expressiveness. The utility of predicate encryption is intimately related to the class of predicates for which we could create secret keys. Ideally, we would like to support the class of all circuits. Over the past decade, substantial advances were made for the weaker primitive of ABE, culminating most recently in schemes supporting any policy computable by general circuits [GVW13, BGG+14] under the standard LWE assumption [Reg09]. However, the state-of-the-art for predicate encryption is largely limited to very simple functionalities related to computing an inner product [BW07, SBC+07, KSW08, AFV11, GMW15].

1.1 Our Contributions

In this work, we substantially advance the state of the art to obtain predicate encryption for all circuits (c.f. Figure 1):

> **Theorem (informal).** Under the LWE assumption, there exists a predicate encryption scheme for all circuits, with succint ciphertexts and secret keys independent of the size of the circuit.

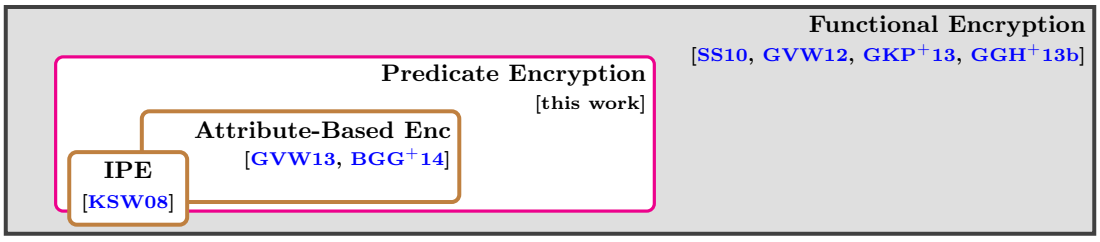

Figure 1: State of the art in functional encryption. The white region refers to functionalities for which we have constructions under standard cryptographic assumptions like LWE or decisional problems in bilinear groups: these functionalities include inner product encryption (IPE), attribute-based encryption for general circuits (ABE) and predicate encryption for general circuits. The grey region refers to functionalities beyond predicate encryption for which we only have constructions for weaker security notions like bounded collusions, or under non-standard cryptographic assumptions like obfuscation or multilinear maps.

As with prior LWE-based ABE for circuits [GVW13, BGG+14], to support circuits of depth d, the parameters of the scheme grow with $\mathrm{poly}(d)$, and we require sub-exponential $n^{\Omega(d)}$ hardness of the LWE assumption. In addition, the security guarantee is selective, but can be extended to adaptive security via complexity leveraging [BB04].

Privacy guarantees. The privacy notion we achieve is a *simulation-based* variant of "attribute-hiding" from the literature [SBC+07, OT10, AFV11]. That is, we guarantee privacy of the attribute x and the plaintext μ against collusions holding secret keys for functions f such that $f(x) = 0$. An even stronger requirement would be to require privacy of x even against authorized keys corresponding to functions f where $f(x) = 1$; in the literature, this stronger notion is referred to as "full attribute-hiding" [BW07, KSW08]. This stronger requirement is equivalent to "full-fledged" functional encryption [BSW11], for which we cannot hope to achieve simulation-based security for all circuits as achieved in this work [BSW11, AGVW13].

Relation to prior works. Our result subsumes all prior works on functional encryption under standard cryptographic assumptions, apart from a few exceptions pertaining to the inner product predicate [BW07, KSW08, OT12]. In a recent break-through work, Garg et al. [GGH+13b] gave a beautiful candidate construction of functional encryption for general circuits; however, the construction relies on "multi-linear maps" [GGH13a, CLT13], for which we have few candidates, along with complex intractability assumptions which are presently poorly understood and not extensively studied in cryptanalysis. In contrast, if we consider collusions of a priori *bounded* size, a weaker guarantee that is still meaningful for many applications, then it is possible to obtain functional encryption for general circuits under a large class of standard assumptions [SS10, GVW12, GKP+13]. See Figure 1 for a pictorial summary.

1.2 Overview of Our Construction

Our starting point is the work of Goldwasser, Kalai, Popa, Vaikuntanathan and Zeldovich [GKP+13] who show how to convert an attribute-based encryption (ABE) scheme into a *single key secure* functional encryption (FE) scheme. Recall that in an attribute-based encryption scheme [GPSW06], a ciphertext is associated with a descriptive value (a public "attribute") x and plaintext μ, and it hides μ, but not x. The observation of Goldwasser et al. [GKP+13] is to hide x by encrypting it using a fully homomorphic encryption (FHE) scheme [Gen09, BV11b], and then using the resulting FHE ciphertext as the public "attribute" in an ABE scheme for general circuits [GVW13, BGG+14]. This has the dual benefit of guaranteeing privacy of x, while at the same time allowing homomorphic computation of predicates f on the encryption of x.

This initial idea quickly runs into trouble. The decryptor who is given the predicate secret key for f and a predicate encryption of (x, μ) can indeed compute an *FHE encryption* of $f(x)$. However, the decryption process is confronted with a decision, namely whether to release the message μ or not, and this decision

depends on whether the *plaintext* $f(x)$ is 0 or 1.[1] Clearly, resolving this conundrum requires obtaining $f(x)$, which requires knowledge of the FHE secret key. Goldwasser et al. [GKP+13] solved this by employing a (single use) Yao garbling of the FHE decryption circuit, however this limited them to obtaining *single key secure* predicate/functional encryption schemes.[2]

Our first key idea is to embed the FHE secret key as part of the attributes in the ABE ciphertext. That is, in order to encrypt a plaintext μ with attributes x in the predicate encryption scheme, we first choose a symmetric key $\mathsf{fhe.sk}$ for the FHE scheme, encrypt x into a FHE ciphertext $\hat{x}$, and encrypt μ using the ABE scheme with $(\mathsf{fhe.sk}, \hat{x})$ as the attributes to obtain an ABE ciphertext ct. Our predicate encryption ciphertext is then given by

$$(\hat{x}, \mathsf{ct})$$

To generate the predicate secret key for a function f, one simply generates the ABE secret key for the function g that takes as input $(\mathsf{fhe.sk}, \hat{x})$ and computes

$$g(\mathsf{fhe.sk}, \hat{x}) = \mathsf{FHE.Dec}(\mathsf{fhe.sk}; \mathsf{FHE.Eval}(f, \hat{x}))$$

That is, g first homomorphically computes a FHE encryption of $f(x)$, and then decrypts it using the FHE secret key to output $f(x)$.

At first glance, this idea evokes strong and conflicting emotions as it raises two problems. The first pertains to correctness: we can no longer decrypt the ciphertext since the ABE decryption algorithm needs to know all of the attributes ($\hat{x}$ and $\mathsf{fhe.sk}$), but $\mathsf{fhe.sk}$ is missing. The second pertains to security: the ABE ciphertext ct is not guaranteed to protect the privacy of the attributes, and could leak all of $\mathsf{fhe.sk}$ which together with $\hat{x}$ would leak all of x. Solving both of these problems seems to require designing a predicate encryption scheme from scratch!

Our next key observation is that the bulk of the computation in g, namely the homomorphic evaluation of the function f, is performed on the *public* attribute $\hat{x}$. The only computation performed on the secret value $\mathsf{fhe.sk}$ is FHE decryption which is a fairly lightweight computation. In particular, with all known FHE schemes [Gen09, BV11b, BV11a, BGV12, GSW13, BV14, AP14], decryption corresponds to computing an inner product followed by a threshold function. Furthermore, we do know how to construct lattice-based predicate encryption schemes for threshold of inner product [AFV11, GMW15]. We stress that the latter do not correspond to FHE decryption since the inner product is computed over a vector in the ciphertext and one in the key, whereas FHE decryption requires computing an inner product over two vectors in the ciphertext; nonetheless, we will build upon the proof techniques in achieving attribute-hiding in [AFV11, GMW15] in the proof of security.

In other words, if we could enhance ABE with a modicum of secrecy so that it can perform a heavyweight computation on public attributes followed by a lightweight privacy-preserving computation on *secret* attributes, we are back in business. Our first contribution is to define such an object, that we call *partially hiding predicate encryption*.

Partially Hiding Predicate Encryption. We introduce the notion of partially hiding predicate encryption (PHPE), an object that interpolates between attribute-based encryption and predicate encryption (analogously to partial garbling in [IW14]). In PHPE, the ciphertext, encrypting message μ, is associated with an attribute (x, y) where x is private but y is always public. The secret key is associated with a function f, and decryption succeeds iff $f(x, y) = 1$. On the one extreme, considering a dummy x or functions f that

[1] In fact, there is a syntactic mismatch since $\hat{f}(\cdot)$ is not a predicate, as it outputs an FHE ciphertext.

[2] A reader familiar with [GKP+13] might wonder whether replacing single-use garbled circuits in their construction with reusable garbled circuits (also from [GKP+13]) might remove this limitation. We remark that this does not seem possible, essentially because the construction in [GKP+13] relies crucially on the simplicity of computing garbled inputs from the "garbling key". In particular, in Yao's garbled circuit scheme, the garbling key is (many) pairs of "strings" L_0 and L_1, and a garbling of an input bit b is simply L_b. This fits perfectly with the semantics of ABE (rather, a variant termed two-input ABE in [GKP+13]) that releases one of two possible "messages" L_0 or L_1 depending on the outcome of a computation. In contrast, computing a garbled input in the reusable garbling scheme is a more complex and randomized function of the garbling key, and does not seem to align well with the semantics of ABE.

ignore x and compute on y, we recover attribute-based encryption. On the other end, considering a dummy y or functions f that ignore y and compute on x, we recover predicate encryption.

We will be interested in realizing PHPE for functions ϕ of the form $\phi(x, y) = g(x, h(y))$ for some functions g and h where h may perform arbitrary heavy-weight computation on the public y and g only performs light-weight computation on the private x. Mapping back to our discussion, we would like to achieve PHPE for the "evaluate-then-decrypt" class of functions, namely where g is the FHE decryption function, h is the FHE evaluation function, x is the FHE secret key, and y is the FHE ciphertext. In general, we would like g to be simple and will allow h to be complex. It turns out that we can formalize the observation above, namely that PHPE for this class of functions gives us a predicate encryption scheme. The question now becomes: can we construct PHPE schemes for the "evaluate-then-decrypt" class of functions?

Assuming the subexponential hardness of learning with errors (LWE), we show how to construct a partially hiding predicate encryption for the class of functions $f : \mathbb{Z}_q^t \times \{0,1\}^\ell \to \{0,1\}$ of the form

$$f_\gamma(\mathbf{x}, \mathbf{y}) = \mathsf{IP}_\gamma(\mathbf{x}, h(\mathbf{y})),$$

where $h : \{0,1\}^\ell \to \{0,1\}^t$, $\gamma \in \mathbb{Z}_q$, and $\mathsf{IP}_\gamma(\mathbf{x}, \mathbf{z}) = 1$ iff $\langle \mathbf{x}, \mathbf{z} \rangle = \left(\sum_{i \in [t]} \mathbf{x}[i] \cdot \mathbf{z}[i] \right) = \gamma \mod q$.

This is almost what we want, but not quite. Recall that FHE decryption in many recent schemes [BV11b, BGV12, GSW13, BV14, AP14] is a function that checks whether an inner product of two vectors in $\mathbb{Z}_q^t$ (one of which could be over $\{0,1\}^t$) lies in a certain range. Indeed, if $\mathbf{z} \in \{0,1\}^t$ is an encryption of 1 and $\mathbf{x} \in \mathbb{Z}_q^t$ is the secret key, we know that $\langle \mathbf{x}, \mathbf{z} \rangle \in [q/2 - B, q/2 + B] \pmod q$, where B is the noise range. Applying the so-called "modulus reduction" [BV11b] transformation to all these schemes, we can assume that this range is polynomial in size.

In other words, we will manage to construct a partially hiding PE scheme for the function

$$f_\gamma(\mathbf{x}, \mathbf{y}) : \langle \mathbf{x}, h(\mathbf{y}) \rangle \stackrel{?}{=} \gamma \pmod q$$

whereas we need a partially hiding PE scheme for the FHE decryption function which is

$$f'_R(\mathbf{x}, \mathbf{y}) : \langle \mathbf{x}, h(\mathbf{y}) \rangle \stackrel{?}{\in} R \pmod q$$

where R is the polynomial size range $[q/2 - B, q/2 + B]$ from above. How do we reconcile this disparity?

The "Lazy OR" Trick. The solution, called the "lazy OR trick" [SBC+07, GMW15] is to publish secret keys for all functions f_γ for $\gamma \in R := [q/2 - B, q/2 + B]$. This will indeed allow us to test if the FHE decryption of the evaluated ciphertext is 1 (and reveal the message μ if it is), but it is also worrying. Publishing these predicate secret keys for the predicates f_γ reveals more information than whether $\langle \mathbf{x}, h(\mathbf{y}) \rangle \stackrel{?}{\in} R$. In particular, it reveals what $\langle \mathbf{x}, h(\mathbf{y}) \rangle$ is. This means that an authorized key would leak partial information about the attribute, which we do allow for predicate encryption. On the other hand, for an unauthorized key where the FHE decryption is 0, each of these $f_\gamma, \gamma \in R$ is also an unauthorized key in the PHPE and therefore leaks no information about the attribute. This extends to the collection of keys in R since the PHPE is secure against collusions. For simplicity, we assume in the rest of this overview that FHE decryption corresponds to exactly to inner product.

Asymmetry to the Rescue: Constructing Partially Hiding PE. Our final contribution is the construction of a partially hiding PE for the function class $f_\gamma(\mathbf{x}, \mathbf{y})$ above. We will crucially exploit the fact that f_γ computes an inner product on the private attribute $\mathbf{x}$. There are two challenges here: first, we need to design a decryption algorithm that knows f_γ and $\mathbf{y}$ but not $\mathbf{x}$ (this is different from decryption in ABE where the algorithm also knows $\mathbf{x}$); second, show that the ciphertext does not leak too much information about $\mathbf{x}$. We use the fully key-homomorphic encryption techniques developed by Boneh et al [BGG+14] in the context of constructing an "arithmetic" ABE scheme. The crucial observation about the ABE scheme of [BGG+14] is that while it was not designed to hide the attributes, it can be made to partially

hide them in exactly the way we want. In particular, the scheme allows us to carry out an inner product of a public attribute vector (corresponding to the evaluated FHE ciphertext) and a private attribute vector (corresponding to the FHE secret key $\mathsf{fhe.sk}$), thanks to an inherent asymmetry in homomorphic evaluation of a multiplication gate on ABE ciphertexts. More concretely, in the homomorphic evaluation of a ciphertext for a multiplication gate in [BGG+14], the decryption algorithm works even if one of the attribute remains private, and for addition gates, the decryption algorithms works even if both attributes remain private. This addresses the first challenge of a decryption algorithm that is oblivious to $\mathbf{x}$. For the second challenge of security, we rely on techniques from inner product predicate encryption [AFV11] to prove the privacy of $\mathbf{x}$ Note that in the latter, the inner product is computed over a vector in the ciphertext and one in the key, whereas in our scheme, the inner product is computed over two vectors in the ciphertext. Interestingly, the proof still goes through since the ciphertext in the ABE [BGG+14] has the same structure as the ciphertext in [AFV11]. We refer the reader to Section 3.2.1 for a detailed overview of the partial hiding PE, and to Section 4 for an overview of how we combine the partial hiding PE with FHE to obtain our main result.

Finally, we remark that exploiting asymmetry in multiplication has been used in fairly different contexts in both FHE [GSW13, BV14] and in ABE [GVW13, GV14]. In [GSW13] and in this work, the use of asymmetry was crucial for realizing the underlying cryptographic primitive; whereas in [GVW13, BV14, GV14], asymmetry was used to reduce the noise growth during homomorphic evaluation, thereby leading to quantitative improvements in the underlying assumptions and hence improved efficiency.

1.3 Discussion

Comparison with other approaches. The two main alternative approaches for realizing predicate and functional encryption both rely on multi-linear maps either implicitly, or explicitly. The first is to use indistinguishability obfuscation as in [GGH+13b], and the second is to extend the dual system encryption framework to multi-linear maps [Wat09, GGHZ14]. A crucial theoretical limitation of these approaches is that they all rely on non-standard assumptions; we have few candidates for multi-linear maps [GGH13a, CLT13, GGH15] and the corresponding assumptions are presently poorly understood and not extensively studied in cryptanalysis, and in some cases, broken [CHL+14]. In particular, the latest attack in [CHL+14] highlight the importance of obtaining constructions and developing techniques that work under standard cryptographic assumptions, as is the focus of this work.

Barriers to functional encryption from LWE. We note the two main barriers to achieving full-fledged functional encryption from LWE using our framework. First, the lazy conjunction approach to handle threshold inner product for FHE decryption leaks the exact inner product and therefore cannot be used to achieve full attribute-hiding. Second, we do not currently know of a fully attribute-hiding inner product encryption scheme under the LWE assumption, although we do know how to obtain such schemes under standard assumptions in bilinear groups [OT12, KSW08].

2 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. We use $\mathbf{x}[i]$ to refer to i'th element of vector $\mathbf{x}$. 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.

For a vector $\mathbf{x}$, we let $||\mathbf{x}||$ denote its ℓ_2 norm and $||\mathbf{x}||_\infty$ denote its infinity norm. For a matrix $\mathbf{R} \in \mathbb{Z}^{m \times m}$ we define $||\mathbf{R}||$ (resp. $||\mathbf{R}||_\infty$) as the ℓ_2 (resp. infinity) length of the longest column of $\mathbf{R}$.

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.

2.1 Lattice Preliminaries

2.1.1 Learning With Errors (LWE) Assumption

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

Definition 2.1 (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:*

$$(\mathbf{A}, \mathbf{As} + \mathbf{x}) \quad \textit{and} \quad (\mathbf{A}, \mathbf{u})$$

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

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

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

Regev and Peikert [Reg09, Pei09] showed that there is a B-bounded distribution χ such that solving $\mathsf{dLWE}_{n,m,q,\chi}$ is as hard as (quantumly) approximating certain worst case lattice problems to a factor of $\tilde{O}(n \cdot q/B)$. These lattices problems are believed to be hard to approximate even when q/B is 2^{n^ϵ} for some fixed $0 < \epsilon < 1/2$. Thus, the hardness of dLWE depends on the modulus-to-noise ratio q/B – the smaller the ratio, the harder the problem. Throughout this paper, the parameter $m = \mathrm{poly}(n)$, in which case we will shorten the notation slightly to $\mathsf{LWE}_{n,q,\chi}$. We refer to [Pei09] for a detailed account of the worst-case to average-case connection for LWE.

2.2 Lattice Algorithms

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 2.1 (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 matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and a trapdoor matrix $\mathbf{T} \in \mathbb{Z}^{m \times m}$ such that the distribution of $\mathbf{A}$ is $\mathrm{negl}(n)$-close to uniform.*

Moreover, there is an efficient algorithm $\mathsf{SampleD}$ that with overwhelming probability over all random choices, does the following: For any $\mathbf{u} \in \mathbb{Z}_q^n$, and large enough $s = \Omega(\sqrt{n \log q})$, the randomized algorithm $\mathsf{SampleD}(\mathbf{A}, \mathbf{T}, \mathbf{u}, s)$ outputs a vector $\mathbf{r} \in \mathbb{Z}^m$ with norm $||\mathbf{r}||_\infty \leq ||\mathbf{r}||_2 \leq s\sqrt{n}$ (with probability 1). Furthermore, the following distributions of the tuple $(\mathbf{A}, \mathbf{T}, \mathbf{U}, \mathbf{R})$ are within $\mathrm{negl}(n)$ statistical distance of each other for any polynomial $k \in \mathbb{N}$:

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

Lemma 2.2. *Suppose that $m \geq (n+1)\log q + \omega(\log n)$ and that $q \geq 2$ is square free. Let $\mathbf{R}$ be an $m \times k$ matrix chosen uniformly from $\{-1, 1\}^{m \times k} \mod q$ where $k = k(n)$ is polynomial in n. Let $\mathbf{A}$ and $\mathbf{B}$ be matrices chosen uniformly in $\mathbb{Z}^{n \times m}$ and $\mathbb{Z}^{n \times k}$ respectively. Then for all vectors $\mathbf{w} \in \mathbb{Z}^m$, the distribution $(\mathbf{A}, \mathbf{AR}, \mathbf{R}^T\mathbf{w})$ is statistically close to distribution $(\mathbf{A}, \mathbf{B}, \mathbf{R}^T\mathbf{w})$.*

We will use the following algorithms to sample short vectors from specific lattices. Looking ahead, the algorithm SampleLeft [ABB10, CHKP12] will be used to sample keys in the real system, while the algorithm SampleRight [ABB10] will be used to sample keys in the simulation.

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

$$\begin{array}{l} \textit{Inputs:} \text{ a full rank matrix } \mathbf{A} \text{ in } \mathbb{Z}_q^{n\times m}, \text{ a "short" basis } \mathbf{T}_A \text{ of } \Lambda_q^{\perp}(\mathbf{A}), \\ \text{a matrix } \mathbf{B} \text{ in } \mathbb{Z}_q^{n\times m_1}, \text{ a vector } \mathbf{u} \in \mathbb{Z}_q^n, \text{ and a Gaussian parameter } \alpha. \\ \\ \textit{Output:} \text{ Let } \mathbf{F} := (\mathbf{A} \parallel \mathbf{B}). \text{ The algorithm outputs a vector } \mathbf{e} \in \mathbb{Z}^{m+m_1} \\ \text{in the coset } \Lambda_{\mathbf{F}+\mathbf{u}}. \end{array} \tag{1}$$

Theorem 2.3 ([ABB10, Theorem 17], [CHKP12, Lemma 3.2])**.** *Let* $q > 2$, $m > n$ *and* $\alpha > \|\widetilde{\mathbf{T}_\mathbf{A}}\| \cdot \omega(\sqrt{\log(m+m_1)})$. *Then* $\mathsf{SampleLeft}(\mathbf{A}, \mathbf{B}, \mathbf{T_A}, \mathbf{u}, \alpha)$ *taking inputs as in (1) outputs a vector* $\mathbf{e} \in \mathbb{Z}^{m+m_1}$ *distributed statistically close to* $D_{\Lambda_{\mathbf{F}+\mathbf{u}},\alpha}$*, where* $\mathbf{F} := (\mathbf{A} \parallel \mathbf{B})$.

Where $\|\widetilde{\mathbf{T}}\|$ refers to the norm of Gram-Schmidt orthogonalisation of $\mathbf{T}$. We refer the readers to [ABB10] for more details.

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

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

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

Theorem 2.4 ([ABB10, Theorem 19])**.** *Let* $q > 2, m > n$ *and* $\alpha > \|\widetilde{\mathbf{T_B}}\| \cdot s_R \cdot \omega(\sqrt{\log m})$. *Then* $\mathsf{SampleRight}(\mathbf{A}, \mathbf{B}, \mathbf{R}, \mathbf{T_B}, \mathbf{u}, \alpha)$ *taking inputs as in (2) outputs a vector* $\mathbf{e} \in \mathbb{Z}^{m+k}$ *distributed statistically close to* $D_{\Lambda_{\mathbf{F}+\mathbf{u}},\alpha}$*, where* $\mathbf{F} := (\mathbf{A} \parallel \mathbf{AR} + \mathbf{B})$.

2.2.1 Primitive matrix

We use the primitive matrix $\mathbf{G} \in \mathbb{Z}_q^{n\times m}$ defined in [MP12]. This matrix has a trapdoor $\mathbf{T_G}$ such that $\|\mathbf{T_G}\|_\infty = 2$. We also define an algorithm $\mathbf{G}^{-1} : \mathbb{Z}_q^{n\times m} \to \mathbb{Z}_q^{m\times m}$ which deterministically derives a short pre-image $\widetilde{\mathbf{A}}$ satisfying $\mathbf{G} \cdot \widetilde{\mathbf{A}} = \mathbf{A}$; the algorithm corresponds to bit decomposition.

2.3 Fully-Homomorphic Encryption

We present a fairly minimal definition of fully homomorphic encryption (FHE) which is sufficient for our constructions. A leveled homomorphic encryption scheme is a tuple of polynomial-time algorithms $(\mathsf{HE.KeyGen}, \mathsf{HE.Enc}, \mathsf{HE.Eval}, \mathsf{HE.Dec})$:

- **Key generation.** $\mathsf{HE.KeyGen}(1^\lambda, 1^d, 1^k)$ is a probablistic algorithm that takes as input the security parameter λ, a depth bound d and message length k and outputs a secret key sk.
- **Encryption.** $\mathsf{HE.Enc}(\mathsf{sk}, \mu)$ is a probabilistic algorithm that takes as input sk and a message $\mu \in \{0,1\}^k$ and outputs a ciphertext ct.
- **Homomorphic evaluation.** $\mathsf{HE.Eval}(f, \mathsf{ct})$ is a deterministic algorithm that takes as input a boolean circuit $C : \{0,1\}^k \to \{0,1\}$ of depth at most d and a ciphertext ct and outputs another ciphertext ct'.
- **Decryption.** $\mathsf{HE.Dec}(\mathsf{sk}, \mathsf{ct}')$ is a deterministic algorithm that takes as input sk and ciphertext ct' and outputs a bit.

Correctness. We require perfect decryption correctness with respect to homomorphically evaluated ciphertexts: namely for all λ, d, k and all $\mathsf{sk} \leftarrow \mathsf{HE.KeyGen}(1^\lambda, 1^d, 1^k)$, all $\mu \in \{0,1\}^k$ and for all boolean circuits $C : \{0,1\}^k \to \{0,1\}$ of depth at most d:

$$\Pr\Big[\mathsf{HE.Dec}(\mathsf{sk}, \mathsf{HE.Eval}(C, \mathsf{HE.Enc}(\mathsf{sk}, \mu))) = C(\mu)\Big] = 1$$

where the probablity is taken over $\mathsf{HE.Enc}$ and $\mathsf{HE.KeyGen}$.

Security. We require semantic security for a single ciphertext: namely for every stateful p.p.t. adversary $\mathcal{A}$ and for all $d, k = \mathrm{poly}(\lambda)$, the following quantity

$$\Pr\left[b = b' : \begin{array}{l} \mathsf{sk} \leftarrow \mathsf{Setup}(1^\lambda, 1^d, 1^k); \\ (\mu_0, \mu_1) \leftarrow \mathcal{A}(1^\lambda, 1^d, 1^k); \\ b \xleftarrow{\$} \{0,1\}; \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{sk}, \mu_b); \\ b' \leftarrow \mathcal{A}(\mathsf{ct}) \end{array}\right] - \frac{1}{2}$$

is negligible in λ.

2.3.1 FHE from LWE

We will rely on an instantiation of FHE from the LWE assumption:

Theorem 2.5 (FHE from LWE [BV11b, BGV12, GSW13, BV14, AP14])**.** *There is a FHE scheme* $\mathsf{HE.KeyGen}, \mathsf{HE.Enc}, \mathsf{HE.Eval}, \mathsf{HE.Dec}$ *that works for any* q *with* $q \geq O(\lambda^2)$ *with the following properties:*

- $\mathsf{HE.KeyGen}$ *outputs a secret key* $\mathsf{sk} \in \mathbb{Z}_q^t$ *where* $t = \mathrm{poly}(\lambda)$*;*
- $\mathsf{HE.Enc}$ *outputs a ciphertext* $\mathsf{ct} \in \{0,1\}^\ell$ *where* $\ell = \mathrm{poly}(k, d, \lambda, \log q)$*;*
- $\mathsf{HE.Eval}$ *outputs a ciphertext* $\mathsf{ct}' \in \{0,1\}^t$*;*
- *for any boolean circuit of depth* d*,* $\mathsf{HE.Eval}(C, \cdot)$ *is computed by a boolean circuit of depth* $\mathrm{poly}(d, \lambda, \log q)$*.*
- $\mathsf{HE.Dec}$ *on input* $\mathsf{sk}, \mathsf{ct}'$ *outputs a bit* $b \in \{0,1\}$*. If* ct' *is an encryption of* 1 *then*

$$\sum_{i=1}^{t} \mathsf{sk}[i] \cdot \mathsf{ct}'[i] \in [\lfloor q/2 \rfloor - B, \lfloor q/2 \rfloor + B]$$

  for some fixed $B = \mathrm{poly}(\lambda)$*. Otherwise, if* ct' *is an encryption of* 0*, then*

$$\sum_{i=1}^{t} \mathsf{sk}[i] \cdot \mathsf{ct}'[i] \notin [\lfloor q/2 \rfloor - B, \lfloor q/2 \rfloor + B];$$

- *security relies on* $\mathsf{dLWE}_{\Theta(t),q,\chi}$*.*

We highlight several properties of the above scheme: (1) the ciphertext is a bit-string, (2) the bound B is a polynomial independent of q (here, we crucially exploit the new results in [BV14] together with the use of leveled bootstrapping)[3], (3) the size of normal ciphertexts is independent of the size of the circuit (this is the typical compactness requirement).

[3] Recall that no circular security assumption needs to be made for leveled bootstrapping.

3 Partially Hiding Predicate Encryption

3.1 Definitions

We introduce the notation of partially hiding predicate encryption (PHPE), which interpolates attribute-based encryption and predicate encryption (analogously to partial garbling in [IW14]). In PHPE, the ciphertext, encrypting message μ, is associated with an attribute (x, y) where x is private but y is always public. The secret key is associated with a predicate C, and decryption succeeds iff $C(x, y) = 1$. The requirement is that a collusion learns nothing about (x, μ) if none of them is individually authorized to decrypt the ciphertext. Attribute-based encryption corresponds to the setting where x is empty, and predicate encryption corresponds to the setting where y is empty. We refer the reader to Section A for the standard notion of predicate encryption.

Looking ahead to our construction, we show how to:

- construct PHPE for a restricted class of circuits that is "low complexity" with respect to x and allows arbitrarily polynomial-time computation on y;
- bootstrap this PHPE using FHE to obtain PE for all circuits.

Syntax. A Partially-Hiding Predicate Encryption scheme $\mathcal{PHPE}$ for a pair of input-universes $\mathcal{X}, \mathcal{Y}$, a predicate universe $\mathcal{C}$, a message space $\mathcal{M}$, consists of four algorithms $(\mathsf{PH.Setup}, \mathsf{PH.Enc}, \mathsf{PH.Keygen}, \mathsf{PH.Dec})$:

$\mathsf{PH.Setup}(1^\lambda, \mathcal{X}, \mathcal{Y}, \mathcal{C}, \mathcal{M}) \to (\mathsf{ph.mpk}, \mathsf{ph.msk})$. The setup algorithm gets as input the security parameter λ and a description of $(\mathcal{X}, \mathcal{Y}, \mathcal{C}, \mathcal{M})$ and outputs the public parameter $\mathsf{ph.mpk}$, and the master key $\mathsf{ph.msk}$.

$\mathsf{PH.Enc}(\mathsf{ph.mpk}, (x, y), \mu) \to \mathsf{ct}_y$. The encryption algorithm gets as input $\mathsf{ph.mpk}$, an attribute $(x, y) \in \mathcal{X} \times \mathcal{Y}$ and a message $\mu \in \mathcal{M}$. It outputs a ciphertext ct_y.

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

$\mathsf{PH.Dec}((\mathsf{sk}_C, C), (\mathsf{ct}_y, y)) \to \mu$. The decryption algorithm gets as input the secret key sk_C, a predicate C, and a ciphertext ct_y and the public part of the attribute y. It outputs a message $\mu \in \mathcal{M}$ or $\perp$.

Correctness. We require that for all $\mathsf{PH.Setup}(1^\lambda, \mathcal{X}, \mathcal{Y}, \mathcal{C}, \mathcal{M}) \to (\mathsf{ph.mpk}, \mathsf{ph.msk})$, for all $(x, y, C) \in \mathcal{X} \times \mathcal{Y} \times \mathcal{C}$, for all $\mu \in \mathcal{M}$,

- if $C(x, y) = 1$,
$$\Pr\left[\mathsf{PH.Dec}((\mathsf{sk}_C, C), (\mathsf{ct}_y, y)) = \mu\right] \geq 1 - \mathrm{negl}(\lambda),$$
- if $C(x, y) = 0$,
$$\Pr\left[\mathsf{PH.Dec}((\mathsf{sk}_C, C), (\mathsf{ct}_y, y)) = \perp\right] \geq 1 - \mathrm{negl}(\lambda),$$

where the probabilities are taken over $\mathsf{sk}_C \leftarrow \mathsf{PH.Keygen}(\mathsf{ph.msk}, C)$, $\mathsf{ct}_y \leftarrow \mathsf{PH.Enc}(\mathsf{ph.mpk}, (x, y), \mu)$ and coins of $\mathsf{PH.Setup}$.

Definition 3.1 (PHPE Attribute-Hiding)**.** *Fix* $(\mathsf{PH.Setup}, \mathsf{PH.Enc}, \mathsf{PH.Keygen}, \mathsf{PH.Dec})$. *For every stateful p.p.t. adversary* Adv*, and a p.p.t. simulator* Sim*, consider the following two experiments:*

$\underline{\exp^{\mathsf{real}}_{\mathcal{PHPE},\mathrm{Adv}}(1^\lambda)}$:

1: $(x, y) \leftarrow \mathrm{Adv}(1^\lambda, \mathcal{X}, \mathcal{Y}, \mathcal{C}, \mathcal{M})$
2: $(\mathsf{ph.mpk}, \mathsf{ph.msk}) \leftarrow \mathsf{PH.Setup}(1^\lambda, \mathcal{X}, \mathcal{Y}, \mathcal{C}, \mathcal{M})$
3: $\mu \leftarrow \mathrm{Adv}^{\mathsf{PH.Keygen}(\mathsf{msk},\cdot)}(\mathsf{ph.mpk})$
4: $\mathsf{ct}_y \leftarrow \mathsf{PH.Enc}(\mathsf{ph.mpk}, (x, y), \mu)$
5: $\alpha \leftarrow \mathrm{Adv}^{\mathsf{PH.Keygen}(\mathsf{ph.msk},\cdot)}(\mathsf{ct}_y)$
6: *Output* (x, y, μ, α)

$\underline{\exp^{\mathsf{ideal}}_{\mathcal{PHPE},\mathrm{Sim}}(1^\lambda)}$:

1: $(x, y) \leftarrow \mathrm{Adv}(1^\lambda, \mathcal{X}, \mathcal{Y}, \mathcal{C}, \mathcal{M})$
2: $(\mathsf{ph.mpk}, \mathsf{ph.msk}) \leftarrow \mathsf{PH.Setup}(1^\lambda, \mathcal{X}, \mathcal{Y}, \mathcal{C}, \mathcal{M})$
3: $\mu \leftarrow \mathrm{Adv}^{\mathsf{PH.Keygen}(\mathsf{ph.msk},\cdot)}(\mathsf{ph.mpk})$
4: $\mathsf{ct}_y \leftarrow \mathrm{Sim}(\mathsf{mpk}, y, 1^{|x|}, 1^{|\mu|})$
5: $\alpha \leftarrow \mathrm{Adv}^{\mathsf{PH.Keygen}(\mathsf{msk},\cdot)}(\mathsf{ct}_y)$
6: *Output* (x, y, μ, α)

We say an adversary Adv *is admissible if all oracle queries that it makes $C \in \mathcal{C}$ satisfy $C(x, y) = 0$. The Partially-Hiding Predicate Encryption scheme $\mathcal{PHPE}$ is then said to be* attribute-hiding *if there is a p.p.t. simulator* Sim *such that for every stateful p.p.t. adversary* Adv*, the following two distributions are computationally indistinguishable:*

$$\left\{ \exp^{\mathsf{real}}_{\mathcal{PHPE},\mathrm{Adv}}(1^\lambda) \right\}_{\lambda\in\mathbb{N}} \overset{c}{\approx} \left\{ \exp^{\mathsf{ideal}}_{\mathcal{PHPE},\mathrm{Sim}}(1^\lambda) \right\}_{\lambda\in\mathbb{N}}$$

Remarks. We point out some remarks of our definition (SIM-AH) when treated as a regular predicate encryption (i.e. the setting where y is empty; see Definition A.1 for completeness) and how it compares to other definitions in the literature.

- We note the simulator for the challenge ciphertext gets y but not x; this captures the fact that y is public whereas x is private. In addition, the simulator is not allowed to program the public parameters or the secret keys. In the ideal experiment, the simulator does not explicitly learn any information about x (apart from its length); nonetheless, there is implicit leakage about x from the key queries made by an admissible adversary. Finally, we note that we can efficiently check whether an adversary is admissible.

- Our security notion is "selective", in that the adversary "commits" to (x, y) before it sees $\mathsf{ph.mpk}$. It is possible to bootstrap selectively-secure scheme to full security using standard complexity leveraging arguments [BB04, GVW13], at the price of a $2^{|x|}$ loss in the security reduction.

- Our definition refers to a single challenge message, but the definition extends readily to a setting with multiple challenge messages. Moreover, our definition composes in that security for a single message implies security with multiple messages (see Section A.2). The following remarks refer to many messages setting.

- We distinguish between two notions of indistinguishability-based (IND) definitions used in the literature: attribute-hiding (IND-AH)[4] and strong attribute-hiding (IND-SAH)[5] [BW07, SBC+07, KSW08, AFV11]. In the IND-AH, the adversary should not be able to distinguish between two pairs of attributes/messages given that it is restricted to queries which do not decrypt the challenge ciphertext (See Section A.3). It is easy to see that our SIM-AH definition is stronger than IND-AH. Furthermore, IND-SAH also ensures that adversary cannot distinguish between the attributes even when it is allowed to ask for queries that decrypt the messages (in this case, it must output $\mu_0 = \mu_1$). Our SIM-AH definition is weaker than IND-SAH, since we explicitly restrict the adversary to queries that do not decrypt the challenge ciphertext.

- In the context of arbitrary predicates, *strong* variants of definitions (that is, IND-SAH and SIM-SAH) are equivalent to security notions for functional encryption (the simulation definition must be adjusted

[4] Sometimes also referred as weak attribute-hiding.
[5] Sometimes also referred as full attribute-hiding.

to give the simulated the outputs of the queries). However, the strong variant of notion (SIM-SAH) is impossible to realize for many messages [BSW11, AGVW13]. We refer the reader to Section A.2 for a sketch of the impossibility. Hence, SIM-AH is the best-possible simulation security for predicate encryption which we realize in this work. The only problem which we leave open is to realize IND-SAH from standard LWE.

3.2 Our Construction

3.2.1 Overview

We construct a partially hiding predicate encryption for the class of predicate circuits $C : \mathbb{Z}_q^t \times \{0,1\}^\ell \to \{0,1\}$ of the form $\widehat{C} \circ \mathsf{IP}_\gamma$ where $\widehat{C} : \{0,1\}^\ell \to \{0,1\}^t$ is a boolean circuit of depth d, $\gamma \in \mathbb{Z}_q$, and

$$(\widehat{C} \circ \mathsf{IP}_\gamma)(\mathbf{x}, \mathbf{y}) = \mathsf{IP}_\gamma(\mathbf{x}, \widehat{C}(\mathbf{y})),$$

where $\mathsf{IP}_\gamma(\mathbf{x}, \mathbf{z}) = 1$ iff $\langle \mathbf{x}, \mathbf{z} \rangle = \left(\sum_{i \in [t]} \mathbf{x}[i] \cdot \mathbf{z}[i] \right) = \gamma \mod q$. We refer to circuit IP as the generic inner-product circuit of two vectors.

Looking ahead, $\widehat{C}$ corresponds to FHE evaluation of an arbitrary circuit C, whereas IP_γ corresponds to roughly to FHE decryption; in the language of the introduction in Section 1, $\widehat{C}$ corresponds to heavy-weight computation h, whereas IP_γ corresponds to light-weight computation g.

The scheme. The public parameters are matrices

$$\left(\mathbf{A},\ \mathbf{A}_1, \ldots, \mathbf{A}_\ell,\ \mathbf{B}_1, \ldots, \mathbf{B}_t \right)$$

An encryption corresponding to the attribute $(\mathbf{x}, \mathbf{y}) \in \mathbb{Z}_q^t \times \{0,1\}^\ell$ is a GPV ciphertext (an LWE sample) corresponding to the matrix

$$\left[\mathbf{A} \mid \mathbf{A}_1 + \mathbf{y}[1] \cdot \mathbf{G} \mid \cdots \mid \mathbf{A}_\ell + \mathbf{y}[\ell] \cdot \mathbf{G} \mid \mathbf{B}_1 + \mathbf{x}[1] \cdot \mathbf{G} \mid \cdots \mid \mathbf{B}_t + \mathbf{x}[t] \cdot \mathbf{G} \right]$$

To decrypt the ciphertext given $\mathbf{y}$ and a key for $\widehat{C} \circ \mathsf{IP}_\gamma$, we apply the BGGHNSVV algorithm to first transform the first part of the ciphertext into a GPV ciphertext corresponding to the matrix

$$\left[\mathbf{A} \mid \mathbf{A}_{\widehat{C}_1} + \mathbf{z}[1] \cdot \mathbf{G} \mid \cdots \mid \mathbf{A}_{\widehat{C}_t} + \mathbf{z}[t] \cdot \mathbf{G} \right]$$

where $\widehat{C}_i$ is the circuit computing the i'th bit of $\widehat{C}$ and $\mathbf{z} = \widehat{C}(\mathbf{y}) \in \{0,1\}^t$. Next, observe that

$$-\Big((\mathbf{A}_{\widehat{C}_i} + \mathbf{z}[i] \cdot \mathbf{G}) \cdot \mathbf{G}^{-1}(\mathbf{B}_i) \Big) + \mathbf{z}[i] \cdot \Big(\mathbf{B}_i + \mathbf{x}[i] \cdot \mathbf{G} \Big) = -\mathbf{A}_{\widehat{C}_i} \mathbf{G}^{-1}(\mathbf{B}_i) + \mathbf{x}[i] \cdot \mathbf{z}[i] \cdot \mathbf{G}.$$

Summing over i, we have

$$\sum_{i=1}^{\ell} -\Big((\mathbf{A}_{\widehat{C}_i} + \mathbf{z}[i] \cdot \mathbf{G}) \cdot \mathbf{G}^{-1}(\mathbf{B}_i) \Big) + \mathbf{z}[i] \cdot \Big(\mathbf{B}_i + \mathbf{x}[i] \cdot \mathbf{G} \Big) = \mathbf{A}_{\widehat{C} \circ \mathsf{IP}} + \langle \mathbf{x}, \mathbf{z} \rangle \cdot \mathbf{G}$$

where

$$\mathbf{A}_{\widehat{C} \circ \mathsf{IP}} := -\Big(\mathbf{A}_{\widehat{C}_1} \mathbf{G}^{-1}(\mathbf{B}_1) + \cdots + \mathbf{A}_{\widehat{C}_t} \mathbf{G}^{-1}(\mathbf{B}_t) \Big).$$

Therefore, given only the public matrices and $\mathbf{y}$ (but not $\mathbf{x}$), we may transform the ciphertext into a GPV ciphertext corresponding to the matrix

$$\left[\mathbf{A} \mid \mathbf{A}_{\widehat{C} \circ \mathsf{IP}} + \langle \mathbf{x}, \mathbf{z} \rangle \cdot \mathbf{G} \right].$$

The secret key corresponding to $\widehat{C} \circ \mathsf{IP}_\gamma$ is essentially a "short basis" for the matrix

$$\left[\mathbf{A} \mid \mathbf{A}_{\widehat{C} \circ \mathsf{IP}} + \gamma \cdot \mathbf{G} \right]$$

which can be sampled using a short trapdoor $\mathbf{T}$ of the matrix $\mathbf{A}$.

Proof strategy. There are two main components to the proof. Fix the selective challenge attribute $\mathbf{x}, \mathbf{y}$. First, we will simulate the secret keys without knowing the trapdoor for the matrix $\mathbf{A}$: here, we rely on the simulated key generation for the ABE [BGG+14]. Roughly speaking, we will need to generate a short basis for the matrix

$$\left[\, \mathbf{A} \mid \mathbf{A}\mathbf{R}_{\widehat{C} \circ \mathsf{IP}} + (\gamma - \widehat{C} \circ \mathsf{IP}(\mathbf{x}, \mathbf{y})) \cdot \mathbf{G} \,\right]$$

where $\mathbf{R}_{\widehat{C} \circ \mathsf{IP}}$ is a small-norm matrix known to the simulator. Now, whenever $\widehat{C} \circ \mathsf{IP}(\mathbf{x}, \mathbf{y}) \neq \gamma$ as is the case for admissible adversaries, we will be able to simultae secret keys using the puncturing techniques in [ABB10, AFV11, MP12]

Next, we will show that the attribute $\mathbf{x}$ is hidden in the challenge ciphertext. Here, we adopt the proof strategy for attribute-hiding inner product encryption in [AFV11, GMW15]. In the proof, we simulate the matrices $\mathbf{A}, \mathbf{B}_1, \ldots, \mathbf{B}_t$ using

$$\mathbf{A}, \mathbf{A}\mathbf{R}'_1 - \mathbf{x}[1]\mathbf{G}, \ldots, \mathbf{A}\mathbf{R}'_t - \mathbf{x}[t]\mathbf{G}$$

where $\mathbf{R}'_1, \ldots, \mathbf{R}'_t \xleftarrow{\$} \{\pm 1\}^{m \times m}$. In addition, we simulate the corresponding terms in the challenge ciphertext by:

$$\mathbf{c}, \mathbf{c}^\intercal \mathbf{R}'_1, \ldots, \mathbf{c}^\intercal \mathbf{R}'_t$$

where $\mathbf{c}$ is a uniformly random vector, which we switched from $\mathbf{A}^\intercal \mathbf{s} + \mathbf{e}$ using the LWE assumption. Here we crucially rely on the fact that switched to simulation of secret keys without knowing the trapdoor of $\mathbf{A}$. Going further, once $\mathbf{c}$ is random, we can switch back to simulating secret keys using the trapdoor $\mathbf{T}$. Hence, the secret keys now do not leak any information about $\mathbf{R}'_1, \ldots, \mathbf{R}'_t$. Therefore, we may then invoke the left-over hash lemma to argue that $\mathbf{x}$ is information-theoretically hidden.

3.2.2 Auxiliary evaluation algorithms

In order to formally describe our scheme, we first need to recall two algorithms $(\mathsf{Eval}_{\mathsf{pk}}, \mathsf{Eval}_{\mathsf{ct}})$ from the BGGHNSVV14 ABE [BGG+14], which we may use as a "black box" and then extend to our setting. Given a boolean predicate $C : \{0,1\}^\ell \to \{0,1\}$ and $\mathbf{y} \in \{0,1\}^\ell$, the algorithm $\mathsf{Eval}_{\mathsf{ct}}$ transforms a GPV ciphertext for the matrix

$$\left[\, \mathbf{A}_1 + \mathbf{y}[1] \cdot \mathbf{G} \mid \cdots \mid \mathbf{A}_\ell + \mathbf{y}[\ell] \cdot \mathbf{G} \right]$$

into one for the matrix

$$\left[\, \mathbf{A}_C + C(\mathbf{y}) \cdot \mathbf{G} \right],$$

where the matrix $\mathbf{A}_C$ is deterministically derived from $(C, \mathbf{A}_1, \ldots, \mathbf{A}_\ell)$ via $\mathsf{Eval}_{\mathsf{pk}}$. We then extend $\mathsf{Eval}_{\mathsf{pk}}, \mathsf{Eval}_{\mathsf{ct}}$ to handle circuits $\widehat{C} \circ \mathsf{IP}$ as outlined above, with the additional property that $\mathsf{Eval}_{\mathsf{ct}}$ is oblivious to $\mathbf{x}$. We exploit the fact that for a multiplication gate, $\mathsf{Eval}_{\mathsf{ct}}$ works even if one of the attribute remains private, and for addition gates, $\mathsf{Eval}_{\mathsf{ct}}$ works even if both attributes remain private. Concretely, $\mathsf{Eval}_{\mathsf{ct}}$ transforms a GPV ciphertext for the matrix

$$\left[\, \mathbf{A}_1 + \mathbf{y}[1] \cdot \mathbf{G} \mid \cdots \mid \mathbf{A}_\ell + \mathbf{y}[\ell] \cdot \mathbf{G} \mid \mathbf{B}_1 + \mathbf{x}[1] \cdot \mathbf{G} \mid \cdots \mid \mathbf{B}_t + \mathbf{x}[t] \cdot \mathbf{G} \,\right]$$

into one for the matrix

$$\left[\, \mathbf{A}_{\widehat{C} \circ \mathsf{IP}} + (\widehat{C} \circ \mathsf{IP})(\mathbf{x}, \mathbf{y}) \cdot \mathbf{G} \right],$$

where the matrix $\mathbf{A}_{\widehat{C} \circ \mathsf{IP}}$ is deterministically derived from $(\widehat{C}, \mathbf{A}_1, \ldots, \mathbf{A}_\ell, \mathbf{B}_1, \ldots, \mathbf{B}_t)$ via $\mathsf{Eval}_{\mathsf{pk}}$, and where $\mathsf{Eval}_{\mathsf{ct}}$ gets $\mathbf{y}$ but not $\mathbf{x}$.

Two basic algorithms. The BGGHNSVV14 ABE provides two deterministic algorithms $\mathsf{Eval}_{\mathsf{pk}}, \mathsf{Eval}_{\mathsf{ct}}$ with the following properties:

- $\mathsf{Eval}_{\mathsf{pk}}$ takes as input ℓ matrices $\mathbf{A}_1, \ldots, \mathbf{A}_\ell \in \mathbb{Z}_q^{n \times m}$ and a predicate $C : \{0,1\}^\ell \to \{0,1\}$, outputs a matrix $\mathbf{A}_C \in \mathbb{Z}_q^{n \times m}$;

- $\mathsf{Eval}_{\mathsf{ct}}$ takes as input $\mathbf{A}_1, \ldots, \mathbf{A}_\ell$ and C as before, along with $\mathbf{y} \in \{0,1\}^\ell$ and ℓ vectors $\mathbf{u}_1, \ldots, \mathbf{u}_\ell \in \mathbb{Z}_q^m$, outputs a vector $\mathbf{u}_C \in \mathbb{Z}_q^m$.

The algorithms satisfy the following properties:

- if $(\mathbf{u}_1, \ldots, \mathbf{u}_\ell) \approx \big((\mathbf{A}_1 + \mathbf{y}[1] \cdot \mathbf{G})^\intercal \mathbf{s}, \ldots, (\mathbf{A}_\ell + \mathbf{y}[\ell] \cdot \mathbf{G})^\intercal \mathbf{s}\big)$, then $\mathbf{u}_C \approx (\mathbf{A}_C + C(\mathbf{y}) \cdot \mathbf{G})^\intercal \mathbf{s}$.
- if $(\mathbf{A}_1, \ldots, \mathbf{A}_\ell) = (\mathbf{A}\mathbf{R}_1 - \mathbf{y}[1] \cdot \mathbf{G}, \ldots, \mathbf{A}\mathbf{R}_\ell - \mathbf{y}[\ell] \cdot \mathbf{G})$ where $\mathbf{R}_1, \ldots, \mathbf{R}_\ell$ are small-norm matrices, then we have
$$\mathbf{A}_C = \mathbf{A}\mathbf{R}_C - C(\mathbf{y}) \cdot \mathbf{G}$$
where $\mathbf{R}_C$ is also a small-norm matrix with a roughly n^{2d} multiplicative blow-up.

These two properties are formalized quantitatively in the following lemma:

Lemma 3.1 (properties of $\mathsf{Eval}_{\mathsf{pk}}, \mathsf{Eval}_{\mathsf{ct}}$ [BGG+14])**.** *The algorithms $\mathsf{Eval}_{\mathsf{pk}}, \mathsf{Eval}_{\mathsf{ct}}$ satisfy the following properties. For all $\mathbf{A}_1, \ldots, \mathbf{A}_\ell \in \mathbb{Z}_q^{n \times m}$, all $\mathbf{y} \in \{0,1\}^\ell$, all boolean predicate C of depth d, let $\mathbf{A}_C := \mathsf{Eval}_{\mathsf{pk}}(\mathbf{A}_1, \ldots, \mathbf{A}_\ell, C)$. Then,*

- *for all $\mathbf{u}_1, \ldots, \mathbf{u}_\ell \in \mathbb{Z}_q^m$ and all $\mathbf{s} \in \mathbb{Z}_q^n$,*
$$\|\mathbf{u}_C - (\mathbf{A}_C + C(\mathbf{y}) \cdot \mathbf{G})^\intercal \mathbf{s}\|_\infty \leq O(\ell n \log q)^{O(d)} \cdot \max_{i \in [\ell]} \big\{\|\mathbf{u}_i - (\mathbf{A}_i + \mathbf{y}[i] \cdot \mathbf{G})^\intercal \mathbf{s}\|_\infty\big\}$$
where $\mathbf{u}_C := \mathsf{Eval}_{\mathsf{ct}}(\mathbf{A}_1, \ldots, \mathbf{A}_\ell, \mathbf{u}_1, \ldots, \mathbf{u}_\ell, \mathbf{y}, C)$.
- *if $(\mathbf{A}_1, \ldots, \mathbf{A}_\ell) = (\mathbf{A}\mathbf{R}_1 - \mathbf{y}[1] \cdot \mathbf{G}, \ldots, \mathbf{A}\mathbf{R}_\ell - \mathbf{y}[\ell] \cdot \mathbf{G})$ where $\mathbf{R}_1, \ldots, \mathbf{R}_\ell \in \mathbb{Z}_q^{m \times m}$, then we have*
$$\mathbf{A}_C = \mathbf{A}\mathbf{R}_C - C(\mathbf{y}) \cdot \mathbf{G}$$
where $\mathbf{R}_C$ is efficiently computable given $(C, \mathbf{A}, \mathbf{R}_1, \ldots, \mathbf{R}_\ell)$ and
$$\|\mathbf{R}_C\|_\infty \leq O(\ell n \log q)^{O(d)} \cdot \max\{\|\mathbf{R}_1\|_\infty, \ldots, \|\mathbf{R}_\ell\|_\infty\}$$

Extension to $\widehat{C} \circ \mathsf{IP}$. We extend the above algorithms to obtain the circuits of the form $\widehat{C} \circ \mathsf{IP}$. Let $\widehat{C}_i$ denote the circuit computing the i'th bit of $\widehat{C}$.

- $\mathsf{Eval}_{\mathsf{pk}}$ takes as input $\ell + t$ matrices $\mathbf{A}_1, \ldots, \mathbf{A}_\ell, \mathbf{B}_1, \ldots, \mathbf{B}_t \in \mathbb{Z}_q^{n \times m}$ and a circuit $\widehat{C} \circ \mathsf{IP} : \{0,1\}^\ell \times \mathbb{Z}_q^t \to \mathbb{Z}_q$, outputs a matrix $\mathbf{A}_{\widehat{C} \circ \mathsf{IP}} \in \mathbb{Z}_q^{n \times m}$ computed as follows:
  1. For $i = 1, \ldots, t$, compute $\mathbf{A}_{\widehat{C}_i} := \mathsf{Eval}_{\mathsf{pk}}(\mathbf{A}_1, \ldots, \mathbf{A}_\ell, \widehat{C}_i)$;
  2. Output $\mathbf{A}_{\widehat{C} \circ \mathsf{IP}} := -\Big(\mathbf{A}_{\widehat{C}_1} \mathbf{G}^{-1}(\mathbf{B}_1) + \cdots + \mathbf{A}_{\widehat{C}_t} \mathbf{G}^{-1}(\mathbf{B}_t)\Big)$.
- $\mathsf{Eval}_{\mathsf{ct}}$ takes as input $\mathbf{A}_1, \ldots, \mathbf{A}_\ell, \mathbf{B}_1, \ldots, \mathbf{B}_t$ and $\widehat{C} \circ \mathsf{IP}$ as before, along with $\mathbf{y} \in \{0,1\}^\ell$ and $\ell + t$ vectors $\mathbf{u}_1, \ldots, \mathbf{u}_\ell, \mathbf{v}_1, \ldots, \mathbf{v}_t \in \mathbb{Z}_q^m$, outputs a vector $\mathbf{u}_{\widehat{C} \circ \mathsf{IP}} \in \mathbb{Z}_q^m$ computed as follows:
  1. For $i = 1, \ldots, t$, compute $\mathbf{u}'_{\widehat{C}_i} := \mathsf{Eval}_{\mathsf{ct}}(\mathbf{A}_1, \ldots, \mathbf{A}_\ell, \mathbf{u}_1, \ldots, \mathbf{u}_\ell, \mathbf{y}, \widehat{C}_i)$;
  2. Output $\mathbf{u}^\intercal_{\widehat{C} \circ \mathsf{IP}} := \Big((\mathbf{z}[1] \cdot \mathbf{v}_1 - \mathbf{G}^{-1}(\mathbf{B}_1)^\intercal \cdot \mathbf{u}'_1) + \cdots + (\mathbf{z}[t] \cdot \mathbf{v}_t - \mathbf{G}^{-1}(\mathbf{B}_t)^\intercal \cdot \mathbf{u}'_{\widehat{C}_i})\Big)$, where $\mathbf{z} = \widehat{C}(\mathbf{y})$.

We stress that $\mathsf{Eval}_{\mathsf{ct}}$ does not get $\mathbf{x}$. We obtain the following extension of Lemma 3.1:

Lemma 3.2 (properties of extended $\mathsf{Eval}_{\mathsf{pk}}, \mathsf{Eval}_{\mathsf{ct}}$)**.** *The algorithms $\mathsf{Eval}_{\mathsf{pk}}, \mathsf{Eval}_{\mathsf{ct}}$ satisfy the following properties. For all $\mathbf{A}_1, \ldots, \mathbf{A}_\ell, \mathbf{B}_1, \ldots, \mathbf{B}_t \in \mathbb{Z}_q^{n \times m}$, all $(\mathbf{x}, \mathbf{y}) \in \mathbb{Z}_q^t \times \{0,1\}^\ell$, all boolean circuits $\widehat{C}$ of depth d, let $\mathbf{A}_{\widehat{C} \circ \mathsf{IP}} := \mathsf{Eval}_{\mathsf{pk}}(\mathbf{A}_1, \ldots, \mathbf{A}_\ell, \mathbf{B}_1, \ldots, \mathbf{B}_t, \widehat{C} \circ \mathsf{IP})$. Then,*

- *for all* $\mathbf{u}_1, \ldots, \mathbf{u}_\ell, \mathbf{v}_1, \ldots, \mathbf{v}_t \in \mathbb{Z}_q^m$ *and all* $\mathbf{s} \in \mathbb{Z}_q^n$,

$$\left\| \mathbf{u}_{\widehat{C} \circ \mathsf{IP}} - (\mathbf{A}_{\widehat{C} \circ \mathsf{IP}} + \langle \mathbf{x}, \widehat{C}(\mathbf{y}) \rangle \cdot \mathbf{G})^\intercal \mathbf{s} \right\|_\infty \leq O(\ell n \log q)^{O(d)} \cdot \max_{i \in [\ell]} \left\{ \left\| \mathbf{u}_i - (\mathbf{A}_i + \mathbf{y}[i] \cdot \mathbf{G})^\intercal \mathbf{s} \right\|_\infty, \ldots. \right\}$$

  where $\mathbf{u}_{\widehat{C} \circ \mathsf{IP}} := \mathsf{Eval}_{\mathsf{ct}}(\mathbf{A}_1, \ldots, \mathbf{A}_\ell, \mathbf{B}_1, \ldots, \mathbf{B}_t, \mathbf{u}_1, \ldots, \mathbf{u}_\ell, \mathbf{v}_1, \ldots, \mathbf{v}_t, \mathbf{y}, \widehat{C} \circ \mathsf{IP})$.

- *if* $(\mathbf{A}_1, \ldots, \mathbf{A}_\ell) = (\mathbf{A}\mathbf{R}_1 - \mathbf{y}[1] \cdot \mathbf{G}, \ldots, \mathbf{A}\mathbf{R}_\ell - \mathbf{y}[\ell] \cdot \mathbf{G})$ *and* $(\mathbf{B}_1, \ldots, \mathbf{B}_t) = (\mathbf{B}\mathbf{R}'_1 - \mathbf{x}[1] \cdot \mathbf{G}, \ldots, \mathbf{B}\mathbf{R}'_\ell - \mathbf{x}[t] \cdot \mathbf{G})$, *where* $\mathbf{R}_1, \ldots, \mathbf{R}_\ell, \mathbf{R}'_1, \ldots, \mathbf{R}'_t \in \mathbb{Z}_q^{m \times m}$, *then we have*

$$\mathbf{A}_{\widehat{C} \circ \mathsf{IP}} = \mathbf{A}\mathbf{R}_{\widehat{C} \circ \mathsf{IP}} + \langle \mathbf{x}, \widehat{C}(\mathbf{y}) \rangle \mathbf{G}$$

  where $\mathbf{R}_{\widehat{C} \circ \mathsf{IP}}$ *is efficiently computable and*

$$\left\| \mathbf{R}_{\widehat{C} \circ \mathsf{IP}} \right\|_\infty \leq O(\ell n \log q)^{O(d)} \cdot \max\{ \|\mathbf{R}_1\|_\infty, \ldots, \|\mathbf{R}_\ell\|_\infty, \|\mathbf{R}'_1\|_\infty, \ldots, \|\mathbf{R}'_t\|_\infty \}$$

Proof. The proof follows readily from Lemma 3.1, along with the calculation

$$\mathbf{R}_{\widehat{C} \circ \mathsf{IP}} := \mathbf{R}_{\widehat{C}_1} \mathbf{G}^{-1}(\mathbf{B}_1) + \cdots + \mathbf{R}_{\widehat{C}_t} \mathbf{G}^{-1}(\mathbf{B}_t) - \left(\hat{\mathbf{y}}[1] \mathbf{R}'_1 + \cdots + \hat{\mathbf{y}}[t] \mathbf{R}'_t \right).$$

□

3.3 Our PHPE scheme

For simplicity, we present our scheme for 1-bit message spaces.

- $\mathsf{PH.Setup}\left(1^\lambda, 1^t, 1^\ell, 1^d\right)$: The setup algorithm takes the security parameter λ, the length of the secret attribute t, the length of the public attribute ℓ, and the circuit depth bound d. Define the lattice parameters $n = n(\lambda), m = m(n, d), q = q(n, d), \chi = \chi(n)$ as per Section 3.6.

  1. Choose random matrices $\mathbf{A}_i \in \mathbb{Z}_q^{n \times m}$ for $i = 1, \ldots, \ell$, $\mathbf{B}_i \in \mathbb{Z}_q^{n \times m}$ for $i = 1, \ldots, t$ and $\mathbf{P} \in \mathbb{Z}_q^{n \times m}$.[6]
  2. Sample a matrix with associated trapdoor:

$$\left(\mathbf{A}, \mathbf{T}\right) \leftarrow \mathsf{TrapGen}(1^m, 1^n, q)$$

  3. Let $\mathbf{G} \in \mathbb{Z}_q^{n \times m}$ be the powers-of-two matrix with a public trapdoor basis $\mathbf{T}_\mathbf{G}$.
  4. Output the master public key $\mathsf{ph.mpk} := \left(\{\mathbf{A}_i\}, \{\mathbf{B}_i\}, \mathbf{A}, \mathbf{P}\right)$ and the master secret key as $\mathsf{ph.msk} := (\mathsf{mpk}, \mathbf{T})$.

- $\mathsf{PH.Keygen}\left(\mathsf{ph.msk}, \widehat{C} \circ \mathsf{IP}_\gamma\right)$: The key-generation algorithms takes as input the master secret key msk, a circuit $\widehat{C} \circ \mathsf{IP}_\gamma$. It outputs a secret key $\mathsf{sk}_{\widehat{C} \circ \mathsf{IP}_\gamma}$ computed as follows.

  1. Let

$$\mathbf{A}_{\widehat{C} \circ \mathsf{IP}} \leftarrow \mathsf{Eval}_{\mathsf{pk}}\left(\{\mathbf{A}_i\}, \{\mathbf{B}_i\}, \widehat{C} \circ \mathsf{IP}\right)$$

     be the homomorphically computed "public key" as per the evaluation algorithm in Section 3.2.2.
  2. Sample a matrix $\mathbf{R} \in \mathbb{Z}_q^{2m \times m}$ such that $[\mathbf{A} | \mathbf{A}_{\widehat{C} \circ \mathsf{IP}} + \gamma \cdot \mathbf{G}] \cdot \mathbf{R} = \mathbf{P} \mod q$, where

$$\mathbf{R} \leftarrow \mathsf{SampleLeft}(\mathbf{A}, \mathbf{A}_{\widehat{C} \circ \mathsf{IP}} + \gamma \cdot \mathbf{G}, \mathbf{T}, \mathbf{P}, s)$$

  3. Output the secret key $\mathsf{sk}_{\widehat{C} \circ \mathsf{IP}_\gamma} := (\mathbf{R})$.

[6] To simplify notation, we always denote the collections $\{\mathbf{A}_i\} := \{\mathbf{A}_i\}_{i \in [\ell]}$ and $\{\mathbf{B}_i\} = \{\mathbf{B}_i\}_{i \in [t]}$.

- $\mathsf{PH.Enc}\big(\mathsf{ph.mpk}, (\mathbf{x}, \mathbf{y}), \mu\big)$: The encryption algorithm takes as input the public key $\mathsf{ph.mpk}$, attribute vectors $\mathbf{x} \in \mathbb{Z}_q^t$, $\mathbf{y} \in \{0,1\}^\ell$ and a message $\mu \in \{0,1\}$. It computes ciphertext $\mathsf{ct}_\mathbf{y}$ as follows.

  1. Choose a secret vector $\mathbf{s} \leftarrow (\chi)^n$ and error terms $\mathbf{e}, \mathbf{e}' \leftarrow (\chi)^m$.
  2. Let $\mathbf{b} = \big[0, \ldots, 0, \lceil q/2 \rceil \mu\big]^\intercal \in \mathbb{Z}_q^m$. Compute encodings
  $$\beta_0 = (\mathbf{A})^\intercal \mathbf{s} + \mathbf{e} \text{ and } \beta_1 = \mathbf{P}^\intercal \mathbf{s} + \mathbf{e}' + \mathbf{b}$$
  3. For all $i = 1, \ldots, \ell$ compute an encoding
  $$\mathbf{u}_i = (\mathbf{A}_i + \mathbf{y}[i] \cdot \mathbf{G})^\intercal \mathbf{s} + \mathbf{R}_i^\intercal \mathbf{e}$$
  where $\mathbf{R}_i \leftarrow \{-1, 1\}^{m \times m}$.
  4. For all $i = 1, \ldots, t$ compute an encoding
  $$\mathbf{v}_i = \big(\mathbf{B}_i + \mathbf{x}[i] \cdot \mathbf{G}\big)^\intercal \mathbf{s} + (\mathbf{R}_i')^\intercal \mathbf{e}_i$$
  where $\mathbf{R}_i' \leftarrow \{-1, 1\}^{m \times m}$.
  5. Output the ciphertext
  $$\mathsf{ct}_\mathbf{y} := \Big(\{\mathbf{u}_i\}_{i \in [\ell]}, \{\mathbf{v}_i\}_{i \in [t]}, \beta_0, \beta_1\Big)$$

- $\mathsf{PH.Dec}\big((\mathsf{sk}_{\widehat{C} \circ \mathsf{IP}_\gamma}, \widehat{C} \circ \mathsf{IP}_\gamma), (\mathsf{ct}_\mathbf{y}, \mathbf{y})\big)$: The decryption algorithm takes as input the secret key $\mathsf{sk}_{\widehat{C} \circ \mathsf{IP}_\gamma}$ for a circuit $\widehat{C} \circ \mathsf{IP}_\gamma$ and the ciphertext $\mathsf{ct}_\mathbf{y}$ along with the public attribute $\mathbf{y}$. It proceeds as follows.

  1. Using $\{\mathbf{u}_i\}, \{\mathbf{v}_i\}$ and $\mathbf{y}$, apply the encoding evaluation algorithm (See Section 3.2.2) to obtain a ciphertext
  $$\mathbf{u}_{\widehat{C} \circ \mathsf{IP}} \leftarrow \mathsf{Eval}_{\mathsf{ct}}\big(\{\mathbf{A}_i, \mathbf{u}_i\}, \{\mathbf{B}_i, \mathbf{v}_i\}, \widehat{C} \circ \mathsf{IP}, \mathbf{y}\big)$$
  where $\mathbf{u}_{\widehat{C} \circ \mathsf{IP}} \approx (\mathbf{A}_{\widehat{C} \circ \mathsf{IP}} + \rho \cdot \mathbf{G})^\intercal \mathbf{s} + \mathbf{e}$ for some $\rho \in \mathbb{Z}_q$.
  2. Now, compute
  $$\eta = \beta_1 - \mathbf{R}^\intercal \cdot \begin{bmatrix} \beta_0 \\ \mathbf{u}_{\widehat{C} \circ \mathsf{IP}} \end{bmatrix} \in \mathbb{Z}_q^m$$
  Output $\mu = \mathsf{Round}(\eta[m])$ if $\big[\mathsf{Round}(\eta[1]), \ldots, \mathsf{Round}(\eta[m-1])\big] = \mathbf{0}$, where
  $$\mathsf{Round}(c) = \begin{cases} 0 & \text{if } |c| < q/4 \\ 1 & \text{otherwise} \end{cases}$$
  Otherwise, output $\perp$.

3.4 Analysis and Correctness

Lemma 3.3. *Let $\mathcal{C}$ be a family of circuits bounded by depth d and let $\mathcal{PHPE}$ be our scheme defined above. Assume that for LWE dimension $n = n(\lambda)$, the parameters are instantiated as follows:*

$$\begin{aligned} \chi &= D_{\mathbb{Z}, \sqrt{n}} \\ q &= \tilde{O}(tnd)^{O(d)} \\ m &= O(n \log q) \\ B &= B(n) \\ s &= O(tn \log q)^{O(d)} \end{aligned}$$

Then, the scheme is correct according to Definition 3.1.

Proof. We proceed proving correctness of the scheme in two steps. First, we bound the error term $\mathbf{e}$ in the final homomorphically computed encoding $\mathbf{u}_{\widehat{C} \circ \mathsf{IP}}$. By Lemma 3.2, the error in $\mathbf{u}_{\widehat{C} \circ \mathsf{IP}}$ satisfies

$$\left\| \mathbf{u}_{\widehat{C} \circ \mathsf{IP}} - (\mathbf{A}_{\widehat{C} \circ \mathsf{IP}} + \langle \mathbf{x}, \widehat{C}(\mathbf{y}) \rangle \cdot \mathbf{G})^\intercal \mathbf{s} \right\|_\infty \leq O(tB \cdot O(n \log q)^{O(d+1)}).$$

Recall that $[\mathbf{A} \mid \mathbf{A}_{\widehat{C} \circ \mathsf{IP}} + \gamma \cdot \mathbf{G}] \cdot \mathbf{R} = \mathbf{P} \mod q$ and $\|\mathbf{R}^\intercal\|_\infty \leq s\sqrt{m}$. After multiplying by $\mathbf{R}^\intercal$, we obtain the final error bound of $O(tB \cdot O(n \log q)^{O(d+1)})$. We then consider two cases:

- if $\langle \mathbf{x}, \widehat{C}(\mathbf{y}) \rangle = \gamma \mod q$, then

  $$||\eta = \beta_1 - \mathbf{R}^\intercal \cdot \begin{bmatrix} \beta_0 \\ \mathbf{u}_{\widehat{C} \circ \mathsf{IP}} \end{bmatrix}||_\infty = O(tB \cdot O(n \log q)^{O(d+1)}) \leq q/4$$

  in the first $m-1$ entries for sufficiently large $q = \tilde{O}(tnd)^{O(d)}$. Hence, the message μ is recovered correctly.

- Otherwise, say $\langle \mathbf{x}, \widehat{C}(\mathbf{y}) \rangle = \gamma' \neq \gamma \mod q$ and $\gamma' = \gamma + \gamma^*$. Then, then multiplying by $\mathbf{R}^\intercal = [\mathbf{R}_1, \mathbf{R}_2]$ we obtain

  $$\eta = \beta_1 - \mathbf{R}^\intercal \cdot \begin{bmatrix} \beta_0 \\ \mathbf{u}_{\widehat{C} \circ \mathsf{IP}} \end{bmatrix} = \mathbf{R}_2^\intercal \cdot \gamma^* \cdot \mathbf{G} + \mathbf{e}^*$$

  for some error vector $\mathbf{e}^*$. Hence, with all but negligible probability all first $m-1$ coefficients of η will be below $q/4$.

This concludes the correctness proof. □

3.5 Security

Theorem 3.4. *Let $\mathcal{PHPE}$ be our partially-hiding predicate encryption scheme. Then, it is secure according to Definition 3.1 assuming hardness of Learning With Errors problem.*

Proof. We describe a p.p.t. simulator Sim algorithm and then claim that the output of the ideal experiment is indistinguishable from real via a series of hybrids.

- $\mathsf{Sim}(\mathsf{ph.mpk}, \mathbf{y})$: samples $\beta_0, \beta_1, \mathbf{u}_i, \mathbf{v}_i$ randomly and independently from $\mathbb{Z}_q^m$ and outputs the ciphertext

  $$\mathsf{ct} := \left(\{\mathbf{u}_i\}_{i \in [\ell]}, \{\mathbf{v}_i\}_{i \in [t]}, \mathbf{y}, \beta_0, \beta_1 \right)$$

Hybrid Sequence. We now claim that security of our scheme scheme via a series of hybrids, where Hybrid 0 corresponds to the real experiment and Hybrid 6 corresponds to the simulated experiment using algorithms Sim.

- **Hybrid 0:** The real experiment.
- **Hybrid 1:** The real game algorithms $\mathsf{PH.Setup}, \mathsf{PH.Enc}$ are replaced with $\mathsf{PH.Setup}_1^*, \mathsf{PH.Enc}_1^*$ defined below. Informally, these algorithms use the knowledge of $\mathbf{x}, \mathbf{y}$ to setup the public parameters in a special form.
- **Hybrid 2:** The real game $\mathsf{PH.Keygen}$ is replaced with $\mathsf{PH.Keygen}_1^*$, where instead of using the trapdoor $\mathbf{T}$ of the matrix $\mathbf{A}$, the secret keys are sampled using the public trapdoor $\mathbf{T}_\mathbf{G}$ along with the trapdoor information generated using $\mathsf{PH.Setup}_1^*$.
- **Hybrid 3:** Same as above, except the $\mathsf{PH.Enc}_1^*$ is replaced with $\mathsf{PH.Enc}_2^*$ defined below.
- **Hybrid 4:** Same as above, except $\mathsf{PH.Keygen}_1^*$ is replaced with real key-generation $\mathsf{PH.Keygen}$.

- **Hybrid 5:** Same as above, except $\mathsf{PH.Enc}_2^*$ is replaced with $\mathsf{PH.Enc}_3^*$ define below, which informally replaces all ciphertext components with random elements.
- **Hybrid 6:** The simulated experiment, that is, same as above, except $\mathsf{PH.Setup}_1^*$ is replaced with $\mathsf{PH.Setup}$.

Auxiliary Algorithms. We now define the auxiliary algorithms similar to those in [AFV11, GMW15] and then argue that the hybrids are either statistically or computationally indistinguishable.

- $\mathsf{PH.Setup}_1^*\big(1^\lambda, 1^d, \mathbf{x}, \mathbf{y}\big)$: In addition to the system parameters, the setup algorithms use the knowledge of the challenge attribute vectors $(\mathbf{x}, \mathbf{y})$. Define the lattice parameters $n = n(\lambda), m = m(n,d), q = q(n,d), \chi = \chi(n)$ (See Section 3.6).

  1. Sample a matrix with associated trapdoor:
  $$\big(\mathbf{A}, \mathbf{T}\big) \leftarrow \mathsf{TrapGen}(1^m, 1^n, q)$$
  Let $\mathbf{G}$ be the powers-of-two matrix with a public trapdoor $\mathbf{T}_{\mathbf{G}}$.
  2. Let $\mathbf{A}_i = \mathbf{A} \cdot \mathbf{R}_i - \mathbf{y}[i] \cdot \mathbf{G} \in \mathbb{Z}_q^{n \times m}$ for $i = 1, \ldots, \ell$ where $\mathbf{R}_i \xleftarrow{\$} \{-1, 1\}^{m \times m}$.
  3. Let $\mathbf{B}_i = \mathbf{A} \cdot \mathbf{R}'_i - \mathbf{x}[i] \cdot \mathbf{G} \in \mathbb{Z}_q^{n \times m}$ for $i = 1, \ldots, t$ where $\mathbf{R}'_i \xleftarrow{\$} \{-1, 1\}^{m \times m}$.
  4. Choose a random matrix $\mathbf{P} \in \mathbb{Z}_q^{n \times m}$.
  5. Output the master public key $\mathsf{ph.mpk} := \big(\{\mathbf{A}_i\}, \{\mathbf{B}_i\}, \mathbf{A}, \mathbf{P}\big)$ and the master secret key as $\mathsf{ph.msk} := (\mathsf{mpk}, \mathbf{T}, \{\mathbf{R}_i\}, \{\mathbf{R}'_i\})$.

  In Hybrids 1, 4, 5, we will generate secret keys using $\mathsf{PH.Keygen}$, which requires knowing $\mathbf{T}$. In Hybrids 2 and 3, we will generate secret keys using $\mathsf{PH.Keygen}_1^*$, which does not require knowing $\mathbf{T}$.

- $\mathsf{PH.Keygen}_1^*\big(\mathsf{ph.msk}, \widehat{C} \circ \mathsf{IP}_\gamma\big)$: The key-generation algorithms takes as input the master secret key $\mathsf{ph.msk}$, a pair of circuits C, IP_γ. It outputs a secret key $\mathsf{sk}_{\widehat{C} \circ \mathsf{IP}_\gamma}$ computed as follows.

  1. Let
  $$\mathbf{A}_{\widehat{C} \circ \mathsf{IP}} \leftarrow \mathsf{Eval}_{\mathsf{pk}}\big(\{\mathbf{A}_i\}, \{\mathbf{B}_i\}, \widehat{C} \circ \mathsf{IP}\big)$$
  be the homomorphically computed "public key" as per the evaluation algorithm in Section 3.2.2.
  2. By Lemma 3.2, $\mathbf{A}_{\widehat{C} \circ \mathsf{IP}} = \mathbf{A} \cdot \mathbf{R}_{\widehat{C} \circ \mathsf{IP}} - \langle \mathbf{x}, \widehat{C}(\mathbf{y}) \rangle \cdot \mathbf{G}$. Now, an admissible adversary is restricted to queries on circuits $\widehat{C} \circ \mathsf{IP}_\gamma$ such that $\langle \mathbf{x}, \widehat{C}(\mathbf{y}) \rangle \neq \gamma$. Hence, we can sample $\mathbf{R} \in \mathbb{Z}_q^{2m \times m}$ such that
  $$[\mathbf{A} \mid \mathbf{A}_{\widehat{C} \circ \mathsf{IP}} + \gamma \cdot \mathbf{G}] \cdot \mathbf{R} = [\mathbf{A} \mid \mathbf{A}\mathbf{R}_{\widehat{C} \circ \mathsf{IP}} + (\gamma - \langle \mathbf{x}, \widehat{C}(\mathbf{y}) \rangle) \cdot \mathbf{G}] \cdot \mathbf{R} = \mathbf{P} \mod q,$$
  using
  $$\mathbf{R} \leftarrow \mathsf{SampleRight}(\mathbf{A}, (\gamma - \langle \mathbf{x}, \widehat{C}(\mathbf{y}) \rangle) \cdot \mathbf{G}, \mathbf{R}_{\widehat{C} \circ \mathsf{IP}}, \mathbf{T}_{\mathbf{G}}, \mathbf{P}, s)$$
  3. Output the secret key $\mathsf{sk}_{\widehat{C} \circ \mathsf{IP}_\gamma} := (\mathbf{R})$.

- $\mathsf{PH.Enc}_1^*\big(\mathsf{ph.mpk}, (\mathbf{x}, \mathbf{y}), \mu\big)$: The encryption algorithm takes as input the public key $\mathsf{ph.mpk}$, challenge vectors $\mathbf{x} \in \mathbb{Z}_q^t$, $\mathbf{y} \in \mathbb{Z}_q^\ell$ and a message μ. It computes ciphertext $\mathsf{ct}_{\mathbf{y}}$ as follows.

  1. Choose a vector $\mathbf{s} \in (\chi)^n$ and compute encodings
  $$\beta_0 = (\mathbf{A})^\intercal \mathbf{s} + \mathbf{e} \text{ and } \beta_1 = \mathbf{P}^\intercal \mathbf{s} + \mathbf{e}' + \mathbf{b}$$
  where $\mathbf{b} = [0, \ldots, 0, \lceil q/2 \rceil \mu]^\intercal \in \mathbb{Z}_q^m$.

2. For all $i = 1, \ldots, \ell$ compute an encoding

$$\begin{aligned} \mathbf{u}_i &= \mathbf{R}_i^\intercal \cdot \beta_0 \\ &= (\mathbf{A} \cdot \mathbf{R}_i)^\intercal \mathbf{s} + \mathbf{R}_i^\intercal \cdot \mathbf{e} \\ &= (\mathbf{A}_i + \mathbf{y}[i] \cdot \mathbf{G})^\intercal \mathbf{s} + \mathbf{R}_i^\intercal \cdot \mathbf{e} \end{aligned}$$

3. For all $i = 1, \ldots, t$ compute an encoding

$$\begin{aligned} \mathbf{v}_i &= (\mathbf{R}'_i)^\intercal \cdot \beta_0 \\ &= (\mathbf{A} \cdot \mathbf{R}'_i)^\intercal \mathbf{s} + (\mathbf{R}'_i)^\intercal \cdot \mathbf{e} \\ &= (\mathbf{B}_i + \mathbf{x}[i] \cdot \mathbf{G})^\intercal \mathbf{s} + (\mathbf{R}'_i)^\intercal \cdot \mathbf{e} \end{aligned}$$

4. Output the ciphertext

$$\mathsf{ct} := \left(\{\mathbf{u}_i\}_{i \in [\ell]}, \{\mathbf{v}_i\}_{i \in [t]}, \mathbf{y}, \beta_0, \beta_1 \right)$$

- $\mathsf{PH.Enc}_2^*\big(\mathsf{ph.mpk}, (\mathbf{x}, \mathbf{y}), \mu\big)$:
  1. Choose random elements β_0, β_1 from $\mathbb{Z}_q^m$.
  2. For all $i = 1, \ldots, \ell$ compute an encoding $\mathbf{u}_i = \mathbf{R}_i^\intercal \cdot \beta_0$.
  3. For all $i = 1, \ldots, t$ compute an encoding $\mathbf{v}_i = (\mathbf{R}'_i)^\intercal \cdot \beta_0$.
  4. Output the ciphertext
$$\mathsf{ct} := \left(\{\mathbf{u}_i\}_{i \in [\ell]}, \{\mathbf{v}_i\}_{i \in [t]}, \mathbf{y}, \beta_0, \beta_1 \right)$$
- $\mathsf{PH.Enc}_3^*\big(\mathsf{ph.mpk}, (\mathbf{x}, \mathbf{y}), \mu\big)$: The final auxiliary encryption algorithm computes ciphertext $\mathsf{ct}_\mathbf{y}$ as a collection of random encodings. That is, $\beta_0, \beta_1, \mathbf{u}_i, \mathbf{v}_i$ are all randomly and independently chosen from $\mathbb{Z}_q^m$. Output the ciphertext
$$\mathsf{ct} := \left(\{\mathbf{u}_i\}_{i \in [\ell]}, \{\mathbf{v}_i\}_{i \in [t]}, \mathbf{y}, \beta_0, \beta_1 \right)$$

Lemma 3.5. *The output of Hybrid 0 is statistically indistinguishable from the output of Hybrid 1.*

Proof. The proof follows closely to [AFV11, Lemma 4.3]. For completeness, we first summarize the difference between the two hybrids:

1. In Hybrid 0, matrices $\mathbf{A}_i, \mathbf{B}_i$ are uniformly chosen in $\mathbb{Z}_q^{n \times m}$. However, in Hybrid 1, $\mathbf{A}_i = \mathbf{A} \cdot \mathbf{R}_i - \mathbf{y}[i] \cdot \mathbf{G}$ and $\mathbf{B}_i = \mathbf{A} \cdot \mathbf{R}'_i - \mathbf{x}[i] \cdot \mathbf{G}$.
2. In Hybrid 0, the ciphertext encodings are computed as

$$\mathbf{u}_i = (\mathbf{A}_i + \mathbf{y}[i] \cdot \mathbf{G})^\intercal \mathbf{s} + \mathbf{R}_i^\intercal \mathbf{e} \quad \text{and} \quad \mathbf{v}_i = (\mathbf{A}_i + \mathbf{x}[i] \cdot \mathbf{G})^\intercal \mathbf{s} + (\mathbf{R}'_i)^\intercal \mathbf{e}$$

   whereas in Hybrid 1 it is computed as

$$\mathbf{u}_i = \mathbf{R}_i^\intercal \beta_0 \quad \text{and} \quad \mathbf{v}_i = (\mathbf{R}'_i)^\intercal \beta_0$$

   where $\beta_0 = (\mathbf{A})^\intercal \mathbf{s} + \mathbf{e}$.

We now argue that the joint distribution of the public parameters, the ciphertext and the secret keys

$$\left(\mathbf{A}, \{\mathbf{A}_i\}, \{\mathbf{B}_i\}, \{\mathbf{u}_i\}, \{\mathbf{v}_i\}, \{\mathsf{sk}_{\widehat{C} \circ \mathsf{IP}}\} \right)$$

is statistically indistinguishable between the two hybrids. Note that the secret keys are produced in both using trapdoor $\mathbf{T}$ and the public matrices. Now, observe that by Lemma 2.2,

$$\left(\mathbf{A}, \mathbf{A}\cdot\mathbf{R}_i - \mathbf{y}[i]\cdot\mathbf{G}, \mathbf{R}_i^\intercal\mathbf{e}, \mathbf{T}\right) \overset{s}{\approx} \left(\mathbf{A}, \mathbf{A}_i, \mathbf{R}_i^\intercal\mathbf{e}, \mathbf{T}\right).$$

This holds for matrices $\mathbf{B}_i$ as well. And since for all i, $\mathbf{R}_i$ (resp. $\mathbf{R}'_i$) is randomly and independently chosen, it follows that

$$\left(\mathbf{A}, \{\mathbf{A}_i\}, \{\mathbf{B}_i\}, \{\mathbf{R}_i^\intercal\mathbf{e}\}, \{(\mathbf{R}'_i)^\intercal\mathbf{e}\}, \mathbf{T}\right) \overset{s}{\approx} \left(\mathbf{A}, \{\mathbf{A}_i\mathbf{R}_i - \mathbf{y}[i]\cdot\mathbf{G}\}, \{\mathbf{B}_i\mathbf{R}'_i - \mathbf{x}[i]\cdot\mathbf{G}\}, \{\mathbf{R}_i^\intercal\mathbf{e}\}, \{(\mathbf{R}'_i)^\intercal\mathbf{e}\}, \mathbf{T}\right).$$

The ciphertext components $\mathbf{u}_i$ and $\mathbf{v}_i$ are derived simply by adding $(\mathbf{A}_i + \mathbf{y}[i]\cdot\mathbf{G})^\intercal\mathbf{s}$ and $(\mathbf{B}_i + \mathbf{x}[i]\cdot\mathbf{G})^\intercal\mathbf{s}$ to $\mathbf{R}_i^\intercal\mathbf{e}$ and $(\mathbf{R}'_i)^\intercal\mathbf{e}$, respectively. And the secret keys are generated from the matrices and the trapdoor $\mathbf{T}$. Since applying a function to two statistically indistinguishable distributions produces two statistically indistinguishable distributions, this shows that the public parameters, the ciphertext and the secret keys are statistically close in both hybrids. □

Lemma 3.6. *The output of Hybrid 1 is statistically indistinguishable from the output of Hybrid 2.*

Proof. From Hybrid 1 to Hybrid 2, we switch between between Keygen and $\mathsf{PH.Keygen}^*$. Fix a secret key query $\widehat{C}\circ\mathsf{IP}_\gamma$ made by an admissible adversary:

- In Hybrid 1, the secret key is sampled using $\mathsf{SampleLeft}$ with trapdoor $\mathbf{T}$, and its distribution only depends on the public matrices in $\mathsf{ph.mpk}$ by Theorem 2.3 (provided the parameter s is sufficiently large);
- In Hybrid 2, the secret key is sampled using $\mathsf{SampleRight}$ with trapdoor $\mathbf{T}_\mathbf{G}$ along with $\mathbf{R}_i, \mathbf{R}'_i$ (which we can do since the adversary is admissible) and its distribution only depends on the public matrices in $\mathsf{ph.mpk}$ by Theorem 2.4 (again, provided s is sufficiently large).

In particular, in both hybrids, the distribution of the secret key only depends on the matrices $\mathbf{A}, \mathbf{A}_{\widehat{C}\circ\mathsf{IP}} + \gamma\cdot\mathbf{G}, \mathbf{P}$ and are in turn completely determined by $\mathsf{ph.mpk}$. Since $\mathsf{ph.mpk}$ has exactly the same distribution in Hybrids 1 and 2, it follows that the output of both hybrids are statistically indistinguishable. □

Lemma 3.7. *The output of Hybrid 2 is computationally indistinguishable from the output of Hybrid 3, under the LWE assumption.*

Proof. We show how to break the security of LWE given an adversary that distinguishes between the two hybrids. We are given matrices $(\mathbf{A}, \mathbf{P}) \in \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^{n\times m}$ and samples $\mathbf{u}, \mathbf{w} \in \mathbb{Z}_q^m \times \mathbb{Z}_q^m$ which are either LWE samples for some secret vector $\mathbf{s}$ or randomly chosen. We simulate the experiments as follows.

- Runs $\mathsf{PH.Setup}_1^*, \mathsf{PH.Keygen}_1^*$ algorithms using the matrices $\mathbf{A}, \mathbf{P}$ from the challenge.
- To simulate the ciphertext encodings, let $\beta_0 = \mathbf{u}$ and $\beta_1 = \mathbf{w} + \mathbf{b}$, where $\mathbf{b} = \left[0, \ldots, 0, \lceil q/2\rceil\mu\right]^\intercal \in \mathbb{Z}_q^m$. The ciphertext encodings $\mathbf{u}_i, \mathbf{v}_i$ are computed using $\mathbf{R}_i, \mathbf{R}'_i$ as

  $$\mathbf{u}_i = \mathbf{R}_i^\intercal\beta_0 \text{ and } \mathbf{v}_i = (\mathbf{R}'_i)^\intercal\beta_0$$

  Output

  $$\mathsf{ct} := \left(\{\mathbf{u}_i\}_{i\in[\ell]}, \{\mathbf{v}_i\}_{i\in[t]}, \mathbf{y}, \beta_0, \beta_1\right)$$

Now clearly, if $\mathbf{u} = \mathbf{A}^\intercal\mathbf{s} + \mathbf{e}$ and $\mathbf{w} = \mathbf{P}^\intercal\mathbf{s} + \mathbf{e}'$, then the simulation is identical to Hybrid 2. Otherwise, if $\mathbf{u}, \mathbf{w}$ are random elements then the experiment corresponds exactly to Hybrid 3. Hence, given an adversary that distinguishes between Hybrids 2 and 3, we can break the security of the standard LWE problem. □

Lemma 3.8. *The output of Hybrid 3 is statistically indistinguishable from the output of Hybrid 4.*

Proof. The proof follows similarly to that of Lemma 3.6, where we switch between $\mathsf{PH.Keygen}^*$ and Keygen. □

Lemma 3.9. *The output of Hybrid 4 is statistically indistinguishable from the output of Hybrid 5.*

Proof. In Hybrid 4, the ciphertext encodings are computed as $\mathbf{u}_i = \mathbf{R}_i^\intercal \cdot \beta_0$ and $\mathbf{v}_i = (\mathbf{R}'_i)^\intercal \cdot \beta_0$. However, in Hybrid 5 these are randomly chosen from the encoding space. The indistinguishability of two hybrids follows from the standard leftover hash lemma, since:

- the secret keys are generated using $\mathsf{PH.Keygen}$, which do not use any information about $\mathbf{R}_i, \mathbf{R}'_i$;
- the only additional leakage on $\mathbf{R}_i, \mathbf{R}'_i$ comes from $(\mathbf{AR}_i, \mathbf{BR}'_i)$ in $\mathsf{ph.mpk}$.

Therefore, for all $\mathbf{A}, \mathbf{B}, \beta_0$ and for all i,

$$\left(\mathbf{A}, \mathbf{B}, \beta_0, \{\mathbf{AR}_i, \mathbf{R}_i^\intercal \cdot \beta_0\}, \{\mathbf{BR}'_i, (\mathbf{R}'_i)^\intercal \cdot \beta_0\}\right) \overset{s}{\approx} \left(\mathbf{A}, \mathbf{B}, \beta_0, \{\mathbf{AR}_i, \mathbf{u}_i\}, \{\mathbf{BR}'_i, \mathbf{v}_i\}\right)$$

for randomly chosen $\mathbf{R}_i, \mathbf{R}'_i, \mathbf{u}_i, \mathbf{v}_i$ and sufficiently large $m = O(n \log q)$. □

Lemma 3.10. *The output of Hybrid 5 is statistically indistinguishable from the output of Hybrid 6.*

Proof. The proof follows similarly to that of Lemma 3.5, where we switch between $\mathsf{PH.Setup}_1^*$ and $\mathsf{PH.Setup}$. □

This completes the security proof. □

3.6 Parameters Selection

We must set the parameters to satisfy correctness and security of the scheme. For correctness, we must ensure that the magnitude of the final error $\mathbf{e}$ is below $q/4$. For security, we must ensure the statistical indistingushability of matrix $\mathbf{A}$ from uniform and indistinguishability of $\mathsf{SampleLeft}$ and $\mathsf{SampleRight}$ algorithms by setting parameter s large enough. We start by setting LWE dimension $n = n(\lambda)$, the error distribution $\chi = \chi(n) = D_{\mathbb{Z},\sqrt{n}}$ and the error bound $B = B(n) = O(n)$. We set the modulus $q = \tilde{O}(tnd)^{O(d)}$ and lattice dimension $m = O(n \log q)$ to apply Lemma 2.1. Finally, we set $s = O(tn \log q)^{O(d)}$ to apply Lemmas 2.3, 2.4. As shown in Section 3.4, these parameters also satisfy the correctness requirements of the scheme. That is, the master public key, ciphertext and secret keys all have size $\mathrm{poly}(\lambda, t, \ell, d)$ where $(1^\lambda, 1^t, 1^\ell, 1^d)$ is the input to $\mathsf{PH.Setup}$ and we achieve security under $\mathsf{LWE}_{n,q,\chi}$ where $q = \tilde{O}(tnd)^{O(d)}$ and the modulus-to-noise ratio is $\tilde{O}(tnd)^{O(d)}$.

4 Predicate Encryption for Circuits

In this section, we present our main construction of predicate encryption for circuits by bootstrapping on top of the partially-hiding predicate encryption. That is,

- We construct a Predicate Encryption scheme $\mathcal{PE} = (\mathsf{Setup}, \mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec})$ for boolean predicate family $\mathcal{C}$ bounded by depth d over k bit inputs.

starting from

- an FHE scheme $\mathcal{FHE} = (\mathsf{HE.KeyGen}, \mathsf{HE.Enc}, \mathsf{HE.Dec}, \mathsf{HE.Eval})$ with properties as described in Section 2.3. Define ℓ as the size of the initial ciphertext encrypting k bit messages, and t as the size of the FHE secret key and evaluated ciphertext vectors;

- a partially-hiding predicate encryption scheme $\mathcal{PHPE} = (\mathsf{PH.Setup}, \mathsf{PH.Keygen}, \mathsf{PH.Enc}, \mathsf{PH.Dec})$ for the class $\mathcal{C}_{\mathsf{PHPE}}$ of predicates bounded by some depth parameter $d' = \mathrm{poly}(d, \lambda, \log q)$. Recall that

$$(\widehat{C} \circ \mathsf{IP}_\gamma)(\mathbf{x} \in \mathbb{Z}_q^t, \mathbf{y} \in \{0,1\}^t) = 1 \text{ iff } \left(\sum_{i \in [t]} \mathbf{x}[i] \cdot \widehat{C}(\mathbf{y})[i] \right) = \gamma \mod q$$

  where $\widehat{C} : \{0,1\}^\ell \to \{0,1\}^t$ is a circuit of depth at most d'.

Overview. At a high level, the construction proceeds as follows:

- the $\mathcal{PE}$ ciphertext corresponding to an attribute $\mathbf{a} \in \{0,1\}^k$ is a $\mathcal{PHPE}$ ciphertext corresponding to an attribute $(\mathsf{fhe.sk}, \mathsf{fhe.ct})$ where $\mathsf{fhe.sk} \xleftarrow{\$} \mathbb{Z}_q^t$ is private and $\mathsf{fhe.ct} := \mathsf{HE.Enc}(\mathbf{a}) \in \{0,1\}^\ell$ is public;
- the $\mathcal{PE}$ secret key for a predicate $C : \{0,1\}^k \to \{0,1\} \in \mathcal{C}$ is a collection of $2B+1$ $\mathcal{PHPE}$ secret keys for the predicates $\{\widehat{C} \circ \mathsf{IP}_\gamma : \mathbb{Z}_q^t \times \{0,1\}^\ell \to \{0,1\}\}_{\gamma = \lfloor q/2 \rfloor - B, \ldots, \lfloor q/2 \rfloor + B}$ where $\widehat{C} : \{0,1\}^\ell \to \{0,1\}$ is the circuit:

$$\widehat{C}(\mathsf{fhe.ct}) := \mathsf{HE.Eval}(\mathsf{fhe.ct}, C),$$

  so $\widehat{C}$ is a circuit of depth at most $d' = \mathrm{poly}(d, \lambda, \log q)$;
- decryption works by trying all possible $2B+1$ secret keys.

Note that the construction relies crucially on the fact that B (the bound on the noise in the FHE evaluated ciphertexts) is polynomial. For correctness, observe that for all $C, \mathbf{a}$:

$$\begin{aligned}
& C(\mathbf{a}) = 1 \\
& \quad \Leftrightarrow \quad \mathsf{HE.Dec}(\mathsf{fhe.sk}, \mathsf{HE.Eval}(C, \mathsf{fhe.ct})) = 1 \\
& \quad \Leftrightarrow \quad \exists\, \gamma \in [\lfloor q/2 \rfloor - B, \lfloor q/2 \rfloor + B] \text{ such that } \left(\sum_{i \in [t]} \mathsf{fhe.sk}[i] \cdot \mathsf{fhe.ct}[i] \right) = \gamma \mod q \\
& \quad \Leftrightarrow \quad \exists\, \gamma \in [\lfloor q/2 \rfloor - B, \lfloor q/2 \rfloor + B] \text{ such that } (\widehat{C} \circ \mathsf{IP}_\gamma)(\mathsf{fhe.sk}, \mathsf{fhe.ct}) = 1
\end{aligned}$$

where $\mathsf{fhe.sk}, \mathsf{fhe.ct}, \hat{C}$ are derived from $C, \mathbf{a}$ as in our construction.

4.1 Our Predicate Encryption scheme

Our construction proceeds as follows:

- $\mathsf{Setup}(1^\lambda, 1^k, 1^d)$: The setup algorithm takes the security parameter λ, the attribute length k and the predicate depth bound d.
  1. Run the partially-hiding PE scheme for family $\mathcal{C}_{\mathsf{PHPE}}$ to obtain a pair of master public and secret keys:

$$(\mathsf{ph.mpk}, \mathsf{ph.msk}) \leftarrow \mathsf{PH.Setup}(1^\lambda, 1^t, 1^\ell, 1^{d'})$$

     where for k-bit messages and depth d circuits: t is the length of FHE secret key, ℓ is the bit-length of the initial FHE ciphertext and d' is the bound on FHE evaluation circuit (as described at the beginning of this section).
  2. Output $(\mathsf{mpk} := \mathsf{ph.mpk}, \mathsf{msk} := \mathsf{ph.msk})$.
- $\mathsf{Keygen}(\mathsf{msk}, C)$: The key-generation algorithms takes as input the master secret key msk and a predicate C. It outputs a secret key sk_C computed as follows.
  1. Let $\widehat{C}(\cdot) := \mathsf{HE.Eval}(\cdot, C)$ and let $(\widehat{C} \circ \mathsf{IP}_\gamma)$ be the predicates for $\gamma = \lfloor q/2 \rfloor - B, \ldots, \lfloor q/2 \rfloor + B$.

2. For all $\gamma = \lfloor q/2 \rfloor - B, \ldots, \lfloor q/2 \rfloor + B$, compute

$$\mathsf{sk}_{\widehat{C} \circ \mathsf{IP}_\gamma} \leftarrow \mathsf{PH.Keygen}\big(\mathsf{ph.msk}, \widehat{C} \circ \mathsf{IP}_\gamma\big)$$

3. Output the secret key as $\mathsf{sk}_C := \big(\{\mathsf{sk}_{\widehat{C} \circ \mathsf{IP}}\}_{\gamma = \lfloor q/2 \rfloor - B, \ldots, \lfloor q/2 \rfloor + B}\big)$.

- $\mathsf{Enc}(\mathsf{mpk}, \mathbf{a}, \mu)$: The encryption algorithm takes as input the public key mpk, the input attribute vector $\mathbf{a} \in \{0,1\}^k$ and message $\mu \in \{0,1\}$. It proceeds as follow.

1. Samples a fresh FHE secret key $\mathsf{fhe.sk} \in \mathbb{Z}_q^t$ by running $\mathsf{HE.KeyGen}(1^\lambda, 1^{d'}, 1^k)$.
2. Encrypt the input to obtain

$$\mathsf{fhe.ct} \leftarrow \mathsf{HE.Enc}(\mathsf{fhe.sk}, \mathbf{a}) \in \{0,1\}^\ell$$

3. Compute

$$\mathsf{ct}_{\mathsf{fhe.ct}} \leftarrow \mathsf{PH.Enc}\big(\mathsf{mpk}, (\mathsf{fhe.sk}, \mathsf{fhe.ct}), \mu\big)$$

Note that the $\mathsf{fhe.sk}$ corresponds to the hidden attribute and $\mathsf{fhe.ct}$ corresponds to the public attribute.

4. Output the ciphertext $\mathsf{ct} = (\mathsf{ct}_{\mathsf{fhe.ct}}, \mathsf{fhe.ct})$.

- $\mathsf{Dec}((\mathsf{sk}_C, C), \mathsf{ct})$: The decryption algorithm takes as input the secret key sk_C with corresponding predicate C and the ciphertext ct. If there exists $\gamma = \lfloor q/2 \rfloor - B, \ldots, \lfloor q/2 \rfloor + B$ such that

$$\mathsf{PH.Dec}((\mathsf{sk}_{\widehat{C} \circ \mathsf{IP}_\gamma}, \widehat{C} \circ \mathsf{IP}_\gamma), (\mathsf{ct}_{\mathsf{fhe.ct}}, \mathsf{fhe.ct})) = \mu \neq \perp$$

then output μ. Otherwise, output $\perp$.

4.2 Correctness

Lemma 4.1. *Let $\mathcal{C}$ be a family of predicates bounded by depth d and let $\mathcal{PHPE}$ be the partially-hiding PE and $\mathcal{FHE}$ be a fully-homomorphic encryption as per scheme description. Then, our predicate encryption scheme $\mathcal{PE}$ is correct according to Definition A. Moreover, the size of each secret key is* $\mathrm{poly}(d, \lambda)$ *and the size of each ciphertext is* $\mathrm{poly}(d, \lambda, k)$.

Proof. Fix an arbitrary attribute vector $\mathbf{a}$ and a predicate C.

- If $C(\mathbf{a}) = 1$, we claim that decryption returns μ with all but negligible probability. By the correctness of FHE decryption, we have

$$\langle \mathsf{fhe.sk}, \mathsf{HE.Eval}(\mathsf{fhe.ct}, C) \rangle = \rho \mod q$$

for some scalar ρ in range $[\lfloor q/2 \rfloor - B, \lfloor q/2 \rfloor + B]$. Hence,

$$\mathsf{PH.Dec}((\mathsf{sk}_{\widehat{C} \circ \mathsf{IP}_\gamma}, \widehat{C} \circ \mathsf{IP}_\gamma), (\mathsf{ct}_{\mathsf{fhe.ct}}, \mathsf{fhe.ct})) = \begin{cases} \mu & \text{if } \gamma = \rho \\ \perp & \text{otherwise} \end{cases}$$

by the correctness of partially-hiding scheme.

- If $C(\mathbf{a}) = 0$, then by the correctness of FHE decryption

$$\langle \mathsf{fhe.sk}, \mathsf{HE.Eval}(\mathsf{fhe.ct}, C) \rangle = \rho \mod q$$

for a scalar ρ outside of range $[\lfloor q/2 \rfloor - B, \lfloor q/2 \rfloor + B]$. Hence, for all $\gamma = \lfloor q/2 \rfloor - B, \ldots, \lfloor q/2 \rfloor + B$,

$$\mathsf{PH.Dec}((\mathsf{sk}_{\widehat{C} \circ \mathsf{IP}_\gamma}, \widehat{C} \circ \mathsf{IP}_\gamma), (\mathsf{ct}_{\mathsf{fhe.ct}}, \mathsf{fhe.ct})) = \perp$$

The correctness of the scheme follows. □

4.3 Security

Theorem 4.2. *Let $\mathcal{C}$ be a family of predicates bounded by depth d and let $\mathcal{PHPE}$ be the secure partially-hiding PE and $\mathcal{FHE}$ be the secure fully-homomorphic encryption as per scheme description. Then, our predicate encryption scheme $\mathcal{PE}$ is secure according to Definition A.1*

Proof. We define p.p.t. simulator algorithms $\mathsf{Enc}_{\mathrm{Sim}}$ and argue that its output is indistinguishable from the output of the real experiment. Let $\mathsf{PH.Enc}_{\mathrm{Sim}}$ be the p.p.t. simulator for partially-hiding predicate encryption scheme.

- $\mathsf{Enc}_{\mathrm{Sim}}(\mathsf{mpk}, 1^{|\mathbf{a}|}, 1^{|\mu|})$: To compute the encryption, the simulator does the following. It samples FHE secret key $\mathsf{fhe.sk}$ by running $\mathsf{HE.KeyGen}(1^\lambda, 1^{d'}, 1^k)$. It encrypts a zero-string $\mathsf{fhe.ct} \leftarrow \mathsf{HE.Enc}(\mathsf{fhe.sk}, \mathbf{0})$. It runs $\mathsf{PH.Enc}_{\mathrm{Sim}}(\mathsf{mpk}, \mathsf{fhe.ct}, 1^{|\mathsf{fhe.sk}|}, 1^{|\mu|})$ to obtain $\mathsf{ct}_{\mathsf{fhe.ct}}$.

We now argue via a series of hybrids that the output of the ideal experiment.

- **Hybrid 0**: The real experiment.
- **Hybrid 1**: The real encryption algorithm is replaced with Enc^*, where Enc^* is an auxiliary algorithm defined below. On the high level, Enc^* computes the FHE ciphertext honestly by sampling a secret key and using the knowledge of $\mathbf{a}$. It then invokes $\mathsf{PH.Enc}_{\mathrm{Sim}}$ on the honestly generated ciphertext.
- **Hybrid 2**: The simulated experiment.

Auxiliary Algorithms. We define the auxiliary algorithm Enc^* used in Hybrid 1.

- $\mathsf{Enc}^*(\mathbf{a}, 1^{|\mu|})$: The auxiliary encryption algorithm takes as input the attribute vector $\mathbf{a}$ and message length.
  1. Sample a fresh FHE secret key $\mathsf{fhe.sk}$ by running $\mathsf{HE.KeyGen}(1^\lambda, 1^{d'}, 1^k)$.
  2. Encrypt the input attribute vector to obtain a ciphertext
  $$\mathsf{fhe.ct} \leftarrow \mathsf{HE.Enc}(\mathsf{fhe.sk}, \mathbf{a}) \in \{0,1\}^\ell$$
  3. Run $\mathsf{PH.Enc}_{\mathrm{Sim}}$ on input $(\mathsf{mpk}, \mathsf{fhe.ct}, 1^{|\mathsf{fhe.sk}|}, 1^{|\mu|})$ to obtain the ciphertext $\mathsf{ct}_{\mathsf{fhe.ct}}$.

Lemma 4.3. *The output of Hybrid 0 is computationally indistinguishable from the Hybrid 1, assuming security of Partially-Hiding Predicate Encryption.*

Proof. Assume there is an adversary Adv and a distinguisher $\mathcal{D}$ that distinguishes the output $(\mathbf{a}, \mu, \alpha)$ produced in either of the two hybrids. We construct an adversary Adv$'$ and a distinguisher $\mathcal{D}'$ that break the security of the Partially-Hiding Predicate Encryption. The adversary Adv$'$ does the following.

1. Invoke the adversary Adv to obtain an attribute vector $\mathbf{a}$.
2. Sample a fresh FHE secret key $\mathsf{fhe.sk}$ using $\mathsf{HE.KeyGen}(1^\lambda, 1^{d'}, 1^k)$. Encrypt the attribute vector
   $$\mathsf{fhe.ct} \leftarrow \mathsf{HE.Enc}(\mathsf{fhe.sk}, \mathbf{a})$$
   and output the pair $(\mathsf{fhe.sk}, \mathsf{fhe.ct})$ as the "selective" challenge attribute.
3. Upon receiving mpk, it forwards it to Adv.
4. For each oracle query C that Adv makes, Adv$'$ uses its oracle to obtain secret keys $\mathsf{sk}_{\widehat{C} \circ \mathsf{IP}_\gamma}$ for $\gamma = \lfloor q/2 \rfloor - B, \ldots, \lfloor q/2 \rfloor + B$. It outputs $\mathsf{sk}_C = \left(\{\mathsf{sk}_{\widehat{C} \circ \mathsf{IP}_\gamma}\}_{\gamma = \lfloor q/2 \rfloor - B, \ldots, \lfloor q/2 \rfloor + B}\right)$.

5. It outputs message μ that Adv produces, obtains a ciphertext $\mathsf{ct}_{\mathsf{fhe.ct}}$ and sends $\mathsf{ct} = (\mathsf{ct}_{\mathsf{fhe.ct}}, \mathsf{fhe.ct})$ back to Adv to obtain α.

We note that given Adv that is admissible, Adv$'$ is also admissible. That is, for all queries $\widehat{C} \circ \mathsf{IP}_\gamma$ that Adv$'$ makes satisfies $(\widehat{C} \circ \mathsf{IP}_\gamma)(\mathsf{fhe.sk}, \mathsf{fhe.ct}) = 0$ since $\langle \mathsf{fhe.sk}, \widehat{C}(\mathsf{fhe.ct}) \rangle \neq \gamma$ for $\gamma = \lfloor q/2 \rfloor - B, \ldots, \lfloor q/2 \rfloor + B$ by the correctness of FHE 2.3. Finally, the distinguisher $\mathcal{D}'$ on input $(\mathsf{fhe.sk}, \mathsf{fhe.ct}, \mu, \alpha)$ invokes $\mathcal{D}$ and outputs whatever it outputs. Now, in Hybrid 0 the algorithms used as $\mathsf{PH.Setup}, \mathsf{PH.Keygen}, \mathsf{PH.Enc}$ which corresponds exactly to the real security game of PHPE. However, in Hybrid 1 the algorithms correspond exactly to the simulated security game. Hence, we can distinguish between the real and simulated experiments contradicting the security of PHPE scheme. □

Lemma 4.4. *The output of Hybrid 1 and Hybrid 2 are computationally indistinguishable, assuming semantic security of Fully-Homomorphic Encryption Scheme.*

Proof. The only difference in Hybrids 1 and 2 is how the FHE ciphertext is produced. In one experiment, it is computed honestly by encrypting the attribute vector $\mathbf{a}$, while in the other experiment it is always an encryption of $\mathbf{0}$. Hence, we can readily construct an FHE adversary that distinguishes encryption of $\mathbf{a}$ from encryption of $\mathbf{0}$. The FHE adversary invokes the admissible PE adversary to obtain an attribute vector $\mathbf{a}$. It then runs the honest $\mathsf{PH.Setup}$ and $\mathsf{PH.Keygen}$ algorithms replying to every query C such that $C(\mathbf{a}) = 0$ with a corresponding secret key sk_C. To simulate the ciphertext, it first forwards a pair $(\mathbf{a}, \mathbf{0})$ to the FHE challenger to obtain a ciphertext $\mathsf{fhe.ct}$. It then runs $\mathsf{PH.Enc}_{\mathrm{Sim}}(\mathsf{mpk}, \mathsf{fhe.ct}, 1^{|\mathsf{fhe.sk}|}, 1^\mu)$ to obtain a ciphertext $\mathsf{ct}_{\mathsf{fhe.ct}}$ which it forwards to PE adversary. Finally, it runs the PE distinguisher on input $(\mathbf{a}, \mu, \alpha)$ and outputs its guess. □

This completes the proof of the security of the scheme. □

4.4 Parameters Selection

We summarize the lattice parameters selection for our construction. First, we set the LWE dimension $n = \mathrm{poly}(\lambda)$ and the error distribution $\chi = \chi(n) = D_{\mathbb{Z},\sqrt{n}}$. Now, we set FHE secret key size $t = \mathrm{poly}(\lambda)$ and modulo $q = \tilde{O}(tnd)^{O(d)}$. To encrypt k-bit attribute vector and support FHE evaluation of arbitrary depth-d circuits, we set $\ell = \mathrm{poly}(k, d, \lambda, \log q)$ and $d' = \mathrm{poly}(d, \lambda, \log q)$. That is, the master public key, ciphertext and secret keys all have size $\mathrm{poly}(\lambda, k, d)$ and we achieve security under $\mathsf{LWE}_{n,q,\chi}$ where $q = 2^{\mathrm{poly}(d,n,\log k)}$ and the modulus-to-noise ratio is $2^{\mathrm{poly}(d,n,\log k)}$.

References

[ABB10] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In *EUROCRYPT*, pages 553–572, 2010.

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

[AGVW13] Shweta Agrawal, Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption: New perspectives and lower bounds. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 500–518. Springer, 2013.

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

[AP14] Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. In *CRYPTO (I)*, pages 297–314, 2014.

[BB04] Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity based encryption without random oracles. In *Advances in Cryptology—EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238. Berlin: Springer-Verlag, 2004.

[BGG+14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In *EUROCRYPT*, pages 533–556, 2014.

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

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

[BV11a] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In *CRYPTO*, pages 505–524, 2011.

[BV11b] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*, pages 97–106, 2011.

[BV14] Zvika Brakerski and Vinod Vaikuntanathan. Lattice-based FHE as secure as PKE. In *ITCS*, pages 1–12, 2014.

[BW07] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In *TCC*, pages 535–554, 2007.

[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.

[CHL+14] Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehlé. Cryptanalysis of the multilinear map over the integers. Cryptology ePrint Archive, Report 2014/906, 2014. http://eprint.iacr.org/.

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

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

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

[GGH+13b] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, pages 40–49, 2013. Also, Cryptology ePrint Archive, Report 2013/451.

[GGH15] Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. In *TCC*, 2015. Also, Cryptology ePrint Archive, Report 2014/645.

[GGHZ14] Sanjam Garg, Craig Gentry, Shai Halevi, and Mark Zhandry. Fully secure functional encryption without obfuscation. Cryptology ePrint Archive, Report 2014/666, 2014.

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

[GMW15] Romain Gay, Pierrick Méaux, and Hoeteck Wee. Predicate encryption for multi-dimensional range queries from lattices. In *Public Key Cryptography*, 2015. Also, Cryptology ePrint Archive, Report 2014/965.

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

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

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO (1)*, pages 75–92, 2013.

[GV14] Sergey Gorbunov and Dhinakaran Vinayagamurthy. Riding on asymmetry: Efficient ABE for branching programs. Cryptology ePrint Archive, Report 2014/819, 2014. http://eprint.iacr.org/.

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

[GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In *STOC*, pages 545–554, 2013. Also, Cryptology ePrint Archive, Report 2013/337.

[IW14] Yuval Ishai and Hoeteck Wee. Partial garbling schemes and their applications. In *ICALP (1)*, pages 650–662, 2014.

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

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

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

[OT12] Tatsuaki Okamoto and Katsuyuki Takashima. Adaptively attribute-hiding (hierarchical) inner product encryption. In *EUROCRYPT*, pages 591–608, 2012. Also, Cryptology ePrint Archive, Report 2011/543.

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

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

[SBC+07] Elaine Shi, John Bethencourt, Hubert T.-H. Chan, Dawn Xiaodong Song, and Adrian Perrig. Multi-Dimensional Range Query over Encrypted Data. In *IEEE Symposium on Security and Privacy*, pages 350–364, 2007.

[SS10] Amit Sahai and Hakan Seyalioglu. Worry-free encryption: functional encryption with public keys. In *ACM Conference on Computer and Communications Security*, pages 463–472, 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.

A Predicate Encryption

We present the definitions of predicate encryption for general circuits [BW07, KSW08, AFV11]. A predicate encryption scheme $\mathcal{PE}$ with respect to an attribute universe $\mathcal{A}$, predicate universe $\mathcal{C}$ and a message universe $\mathcal{M}$ consists of four algorithms $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$:

$\mathsf{Setup}(1^\lambda, \mathcal{A}, \mathcal{C}, \mathcal{M}) \to (\mathsf{mpk}, \mathsf{msk})$. The setup algorithm gets as input the security parameter λ and outputs the public parameter mpk, and the master key msk.

$\mathsf{Enc}(\mathsf{mpk}, a, \mu) \to \mathsf{ct}$. The encryption algorithm gets as input mpk, an attribute $a \in \mathcal{A}$ and a message $\mu \in \mathcal{M}$. It outputs a ciphertext ct.

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

$\mathsf{Dec}((\mathsf{sk}_C, C), \mathsf{ct}) \to \mu$. The decryption algorithm gets as input sk_C and a ciphertext ct. It outputs a message μ.

Correctness. We require that for all $(a, C) \in \mathcal{A} \times \mathcal{C}$ such that $C(a) = 1$ and all $\mu \in \mathcal{M}$,

$$\Pr\left[\mathsf{sk}_C \leftarrow \mathsf{KeyGen}(\mathsf{msk}, C), \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, a, \mu); \mathsf{Dec}((\mathsf{sk}_C, C), \mathsf{ct}) = \mu\right] \geq 1 - \mathrm{negl}(\lambda)$$

And for all $(a, C) \in \mathcal{A} \times \mathcal{C}$ such that $C(a) = 0$ and all $\mu \in \mathcal{M}$,

$$\Pr\left[\mathsf{sk}_C \leftarrow \mathsf{KeyGen}(\mathsf{msk}, C), \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, a, \mu); \mathsf{Dec}((\mathsf{sk}_C, C), \mathsf{ct}) = \perp\right] \geq 1 - \mathrm{negl}(\lambda)$$

where the probability is taken over $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and the coins of $\mathsf{Enc}, \mathsf{KeyGen}$.

A.1 Security Model

Definition A.1 (SIM-AH)**.** *For every stateful p.p.t. adversary* Adv*, and a p.p.t. simulator* Sim*, consider the following two experiments:*

$\underline{\exp^{\mathsf{real}}_{\mathcal{PE},\mathrm{Adv}}(1^\lambda)}$**:**

1: $a \leftarrow \mathrm{Adv}(1^\lambda)$
2: $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$
3: $\mu \leftarrow \mathrm{Adv}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{mpk})$
4: $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, a, \mu)$
5: $\alpha \leftarrow \mathrm{Adv}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{ct})$
6: *Output* (a, μ, α)

$\underline{\exp^{\mathsf{ideal}}_{\mathcal{PE},\mathrm{Sim}}(1^\lambda)}$**:**

1: $a \leftarrow \mathrm{Adv}(1^\lambda)$
2: $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$
3: $\mu \leftarrow \mathrm{Adv}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{mpk})$
4: $\mathsf{ct} \leftarrow \mathrm{Sim}(\mathsf{mpk}, 1^{|a|}, 1^{|\mu|})$
5: $\alpha \leftarrow \mathrm{Adv}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{ct})$
6: *Output* (a, μ, α)

We say an adversary Adv *is admissible if for all oracle queries that it makes $C \in \mathcal{C}$, $C(a) = 0$. The Predicate Encryption scheme $\mathcal{PE}$ is then said to be simulation-based* attribute-hiding (SIM-AH) *if there is a p.p.t. simulator* Sim *such that for every stateful p.p.t. adversary* Adv*, the following two distributions are computationally indistinguishable:*

$$\left\{\exp^{\mathsf{real}}_{\mathcal{PE},\mathrm{Adv}}(1^\lambda)\right\}_{\lambda\in\mathbb{N}} \overset{c}{\approx} \left\{\exp^{\mathsf{ideal}}_{\mathcal{PE},\mathrm{Sim}}(1^\lambda)\right\}_{\lambda\in\mathbb{N}}$$

A.2 Relations to Other Security Models

Single Message Implies Many Message Security. We point out a simple composition result for our definition. The definition for many message security remains virtually identical, except the adversary declares a list of tuples (a_i, μ_i) for which it sees either real or simulated ciphertexts. Given a one-message simulator Sim_1, we can construct a many message simulator Sim_m which just invokes the one-message simulator to simulate each ciphertext independently. The security follows via the standard hybrid argument and crucially relies on the fact that the adversary is restricted to queries that *do not allow* to decrypt. Similar result follows for partially hiding predicate encryption.

Impossibility of Strong-Simulation Security We point out the many-messages strong-simulation security is impossible to realize. In the strong-simulation security, the adversary is *allowed* to query for secret keys that allow to decrypt. The simulator is given the result oracle access to functionality that returns the outputs of the predicates.

Definition A.2 (Many-Messages SIM-SHA)**.** *For every stateful p.p.t. adversary* Adv*, and a p.p.t. simulator* Sim*, consider the following two experiments:*

$\underline{\exp^{\mathsf{real}}_{\mathcal{PE},\mathrm{Adv}}(1^\lambda)}$**:**	$\underline{\exp^{\mathsf{ideal}}_{\mathcal{PE},\mathrm{Sim}}(1^\lambda)}$**:**
1: $\overrightarrow{a} \leftarrow \mathrm{Adv}(1^\lambda)$	1: $\overrightarrow{a} \leftarrow \mathrm{Adv}(1^\lambda)$
2: $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$	2: $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$
3: $\overrightarrow{\mu} \leftarrow \mathrm{Adv}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{mpk})$	3: $\overrightarrow{\mu} \leftarrow \mathrm{Adv}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{mpk})$
4: $\overrightarrow{\mathsf{ct}} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \overrightarrow{a}, \overrightarrow{\mu})$	4: $\overrightarrow{\mathsf{ct}} \leftarrow \mathrm{Sim}^{O(\cdot)}(\mathsf{mpk}, 1^{\lvert a_i \rvert}, 1^{\lvert \mu_i \rvert})$
5: $\alpha \leftarrow \mathrm{Adv}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\overrightarrow{\mathsf{ct}})$	5: $\alpha \leftarrow \mathrm{Adv}^{O'(\mathsf{msk},\cdot)}(\overrightarrow{\mathsf{ct}})$
6: *Output* $(\overrightarrow{a}, \overrightarrow{\mu}, \alpha)$	6: *Output* $(\overrightarrow{a}, \overrightarrow{\mu}, \alpha)$

Where the oracles are defines as:

- $O(C)$*: returns a list* $(C(a_i), b_i)$ *(for all* $i \in |\overrightarrow{a}|$*) where* $b_i = \mu_i$ *if* $C(a_i) = 1$ *and* $b_i = \perp$ *otherwise. In words, it returns the results of the decryption queries that the adversary should learn by the correctness of the scheme.*
- $O'(\mathsf{msk}, C)$*: is the second stage of the simulator, namely* $\mathrm{Sim}^{O(\cdot)}(\mathsf{msk})$*. It is given access to the same oracle that returns results of the decryption queries that must be satisfied by the correctness.*

We say the simulator is admissible if it queries its oracle $O(\cdot)$ *on the identical ordered set of queries issued by* Adv*. The Predicate Encryption scheme* $\mathcal{PE}$ *is then said to be simulation-based* strong attribute-hiding *for many-messages if there is an admissible stateful p.p.t. simulator* Sim *such that for every admissible stateful p.p.t. adversary* Adv*, the following two distributions are computationally indistinguishable:*

$$\left\{ \exp^{\mathsf{real}}_{\mathcal{PE},\mathrm{Adv}}(1^\lambda) \right\}_{\lambda \in \mathbb{N}} \overset{c}{\approx} \left\{ \exp^{\mathsf{ideal}}_{\mathcal{PE},\mathrm{Sim}}(1^\lambda) \right\}_{\lambda \in \mathbb{N}}$$

The impossibility of this notion follows from the compression arguments of [BSW11, AGVW13]. In particular, consider an adversary that outputs a random list of messages $\overrightarrow{\mu}$ and arbitrary identities $\overrightarrow{\mathbf{a}}$. It sets the lengths of these lists to be much greater than the length of secret keys of the scheme. Then, upon receiving the challenge ciphertext, it asks for a query that allows to decrypt all messages. The simulator, on the other side, must first commit to a long string $\overrightarrow{\mathsf{ct}}$ and then later fake a short secret key that must decrypt the entire ciphertext correctly, which is impossible by the standard information theoretic argument.

A.3 Indistinguishability Security of PE

For comparison, we include here the indistinguishability-based formulation of selective, weakly attribute-hiding predicate encryption.

Definition A.3 (IND-AH)**.** *For a stateful adversary $\mathcal{A}$, we define the advantage function*

$$\mathsf{Adv}^{\mathcal{PE}}_{\mathcal{A}}(\lambda) := \Pr\left[\beta = \beta' : \begin{array}{l} (x_0, x_1) \leftarrow \mathcal{A}(1^\lambda); \\ \beta \xleftarrow{\$} \{0,1\}; \\ (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^n, \mathcal{X}, \mathcal{Y}, \mathcal{M}); \\ (\mu_0, \mu_1) \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{mpk}); \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, x_\beta, \mu_\beta); \\ \beta' \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{ct}) \end{array}\right] - \frac{1}{2}$$

with the restriction that all queries C that $\mathcal{A}$ makes to $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ satisfies $C(x_0) = C(x_1) = 0$ (that is, sk_C does not decrypt ct). A predicate encryption scheme is selectively secure *if for all PPT adversaries $\mathcal{A}$, the advantage $\mathsf{Adv}^{\mathcal{PE}}_{\mathcal{A}}(\lambda)$ is a negligible function in λ.*