Multi-Identity and Multi-Key Leveled FHE from Learning with Errors

Michael Clear and Ciarán McGoldrick

School of Computer Science and Statistics,
Trinity College Dublin

Abstract. Gentry, Sahai and Waters recently presented the first (leveled) identity-based fully homomorphic (IBFHE) encryption scheme (CRYPTO 2013). Their scheme however only works in the single-identity setting; that is, homomorphic evaluation can only be performed on ciphertexts created with the same identity. In this work, we extend their results to the multi-identity setting and obtain a multi-identity IBFHE scheme that is selectively secure in the random oracle model under the hardness of Learning with Errors (LWE). We also obtain a multi-key fully-homomorphic encryption (FHE) scheme that is secure under LWE in the standard model. This is the first multi-key FHE based on a well-established assumption such as standard LWE. The multi-key FHE of López-Alt, Tromer and Vaikuntanathan (STOC 2012) relied on a non-standard assumption, referred to as the Decisional Small Polynomial Ratio assumption.

1 Introduction

Fully homomorphic encryption (FHE) is a cryptographic primitive that facilitates arbitrary computation on encrypted data. Since Gentry's breakthrough realization of FHE in 2009 [1], many improved variants have appeared in the literature [2–6].

A leveled FHE scheme allows an evaluator to evaluate a circuit of limited depth L. The parameter L must be specified in advance when generating the public parameters of the scheme, whose size may depend on L. Furthermore, a leveled homomorphic scheme allows L to be polynomial in the security parameter. A "pure" fully homomorphic encryption scheme allows circuits of unlimited depth to be evaluated. However, for many applications in practice, a leveled scheme is adequate.

Identity-Based Encryption (IBE) is centered around the notion that a user's public key can be efficiently derived from an identity string and system-wide public parameters / master public key. The public parameters are chosen by a trusted authority (TA) along with a secret trapdoor (master secret key), which is used to extract secret keys for user identities. The first secure IBE schemes were presented in 2001 by Boneh and Franklin [7] (based on bilinear pairings), and Cocks [8] (based on the quadratic residuosity problem).

At Crypto 2013, Gentry, Sahai and Waters presented the first (leveled) identity-based fully homomorphic encryption (IBFHE) scheme [6]. Their scheme is secure under the hardness of the Learning with Errors (LWE) problem, a problem introduced by Regev [9] that has received considerable attention in cryptography due to a known worst-case reduction to a hard lattice problem.

Gentry, Sahai and Waters described a compiler [6], which we call the GSW compiler, to transform an LWE-based IBE satisfying certain properties into a leveled IBFHE. They showed that all known LWE-based IBE schemes are compatible with their compiler. However, the GSW compiler only works in the *single-identity* setting. In other words, the resulting IBFHE can only evaluate on ciphertexts created with the same identity. Recently, a multi-identity IBFHE was described in [10], but that construction relies heavily on indistinguishability obfuscation [11], and is therefore highly inefficient at the present time. Furthermore, security cannot be based on a well-established computational problem. Our construction does not require indistinguishability obfuscation and is the first multi-identity IBFHE, to the best of our knowledge, whose security can be based on well-established problem.

Remark 1. Like [6], we omit the qualifier "leveled" for the rest of this paper since we focus only on leveled (IB)FHE in this work.

Note that our multi-identity and multi-key leveled IBFHE are 1-hop homomorphic insofar as after evaluation is complete, no further homomorphic evaluation can be carried out.

1.1 Multi-Identity Setting

Consider the following simplified scenario. Alice and Bob work in an organization C that avails of a semi-trusted cloud server E. Let a and b denote the identity strings of Alice and Bob respectively. Their organization C serves as a trusted authority and issues them secret keys for their respective identity strings. Public users can send confidential data to Alice and Bob by encrypting it with their identity string and the master public key (public parameters) published by C. Suppose this encrypted data is sent by external users to the cloud server E. Furthermore, suppose some entity would like to perform some computation on E using encrypted data intended for Alice and encrypted data intended for Bob. The result should only be decryptable (assuming C is honest) by a collaborative effort made by Alice and Bob; they can run a multi-party computation protocol to collaboratively decrypt the result without leaking their secret keys to each other.

Let c_a and c_b be ciphertexts created with identities a and b respectively. The goal is to allow computation on c_a and c_b together. Assuming this could be achieved, let c' denote the ciphertext that encrypts the result of the computation. Intuitively, we expect the size of c' to depend on the number of distinct identities (2 in our example above i.e. a and b) because information about each identity must be "encoded" in c'. But like the single-identity setting, the size of c' should be independent of the size of the circuit evaluated. Of course we can naturally extend this notion to ciphertexts created under k distinct identities.

In the syntax of multi-identity IBFHE, a parameter $\mathcal{D}$ representing the number of distinct identities tolerated in an evaluation is specified in advance of generating the public parameters. Like the parameter L (the circuit depth supported), the size of the public parameters may depend on $\mathcal{D}$. A multi-identity and multi-attribute IBFHE and ABFHE that rely on indistinguishability obfuscation were described in [12].

Disjunctive Policies There is another way of viewing multi-identity IBFHE, which might be more useful in some settings. It was mentioned in [13]* that access policies consisting of disjunctions can be achieved with IBE. In this case, to issue a secret key for a policy $\hat{f}(X) \triangleq X =$ "MATH" OR $X =$ "CS", the TA issues a secret key for identity string "MATH" and a secret key for identity string "CS". In this case, we view the "identities" as attributes.

Suppose the TA issues a secret key $\mathsf{SK}_{\hat{f}} = \{\mathsf{sk}_{\text{"MATH"}}, \mathsf{sk}_{\text{"CS"}}\}$ for $\hat{f}$ to a professor working in both the Mathematics and Computer Science departments in a university; this secret key comprises an IBE secret key for identity string "MATH" and an IBE secret key for identity string "CS". The professor can decrypt the result of computation performed on ciphertexts with both attributes. This matches our intuition because her policy $\hat{f}$ permits her access to both attributes.

1.2 Our Results

Multi-Identity IBFHE Our central result in this paper is informally summarized in the following theorem statement. The theorem is formally stated and proven later in Appendix A.1.

Theorem 1 (Informal). *There exists a multi-identity IBFHE scheme that is selectively secure under the Learning With Errors problem in the random oracle model.*

Multi-Key FHE Our compiler for multi-identity IBFHE also works in the public-key setting. As a result, we can obtain a multi-key FHE [14] from LWE in the standard model. In fact, multi-identity IBFHE can be seen as an identity-based analog to multi-key FHE. The syntax of multi-key FHE from [14] entails a parameter M, which specifies the maximum number of independent keys tolerated in an evaluation. The size of the parameters and ciphertexts are allowed to depend polynomially on M. Note that M is fixed and specified in advance of generating the scheme's parameters. To the best of our knowledge, our multi-key FHE scheme is the first such scheme (for a non-constant number of keys) that is based on a well-established problem such as LWE; the construction from [14]

*The paper [13] attributes this observation to Brent Waters.

relies on a non-standard computational assumption referred to therein as the Decisional Small Polynomial Ratio (DSPR) assumption. Our scheme positively answers the question raised in [14] as to whether other multi-key FHE schemes exist supporting polynomially-sized M.

1.3 Our Approach: Intuition

We now give an informal sketch of our approach to achieving multi-identity IBFHE. This section is intended to provide an intuition and many of the details are deferred to later in the paper. We remind the reader that a matrix $\mathbf{M}$ is denoted by an uppercase symbol written in boldface, and a vector $\boldsymbol{v}$ is denoted by a lowercase symbol written in boldface. The i-th element of $\boldsymbol{v}$ is denoted by v_i. The inner product of two vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}_q^n$ for some dimension n is written as $\langle a, b \rangle$.

GSW single-identity IBFHE We start by briefly discussing the homomorphic properties of the GSW IBFHE schemes from [6]. This discussion applies to *any* IBFHE constructed with their compiler. A ciphertext in their scheme is an $N \times N$ matrix $\mathbf{C}$ over $\mathbb{Z}_q$ whose entries are "small" with respect to q. Note that N is a parameter that will be discussed later. A secret key for an identity id is an N-dimensional vector $\boldsymbol{v}_{\mathsf{id}} \in \mathbb{Z}_q^N$ with at least one "large" coefficient; let this coefficient (say the i-th one) be $v_{\mathsf{id},i} \in \mathbb{Z}_q$. The scheme can encrypt "small" messages μ; an example to keep in mind is a message in $\{0, 1\}$. We say the matrix $\mathbf{C}$ *encrypts* μ under identity id if $\mathbf{C} \cdot \boldsymbol{v}_{\mathsf{id}} = \mu \cdot \boldsymbol{v}_{\mathsf{id}} + \boldsymbol{e} \in \mathbb{Z}_q^N$ where $\boldsymbol{e}$ is a "small" noise vector (i.e. roughly speaking, each of its coefficients is much less than q). As such, $\boldsymbol{v}_{\mathsf{id}}$ is an *approximate* eigenvector for the matrix $\mathbf{C}$ with eigenvalue μ.

Homomorphic Operations

Suppose $\mathbf{C_1}$ and $\mathbf{C_2}$ encrypt μ_1 and μ_2 respectively; that is, $\mathbf{C_j} \cdot \boldsymbol{v}_{\mathsf{id}} = \mu_j \cdot \boldsymbol{v}_{\mathsf{id}} + \boldsymbol{e_j}$ for $j \in \{1, 2\}$. An additive homomorphism is supported. Let $\mathbf{C}^+ = \mathbf{C_1} + \mathbf{C_2}$. Then we have $\mathbf{C}^+ \cdot \boldsymbol{v}_{\mathsf{id}} = (\mu_1 + \mu_2) \cdot \boldsymbol{v}_{\mathsf{id}} + (\boldsymbol{e_1} + \boldsymbol{e_2})$. The error only grows slightly here, and as long as it remains "small", we can recover the sum $(\mu_1 + \mu_2)$. A multiplicative homomorphism is also supported. Let $\mathbf{C}^\times = \mathbf{C_1} \cdot \mathbf{C_2}$. Then we have

$$\begin{aligned}
\mathbf{C}^\times \cdot \boldsymbol{v}_{\mathsf{id}} &= \mathbf{C_1} \cdot (\mu_2 \cdot \boldsymbol{v}_{\mathsf{id}} + \boldsymbol{e_2}) \\
&= \mu_2 \cdot (\mu_1 \cdot \boldsymbol{v}_{\mathsf{id}} + \boldsymbol{e_1}) + \mathbf{C_1} \cdot \boldsymbol{e_2} \\
&= \mu_1 \cdot \mu_2 \cdot \boldsymbol{v}_{\mathsf{id}} + \mu_2 \cdot \boldsymbol{e_1} + \mathbf{C_1} \cdot \boldsymbol{e_2} \\
&= \mu_1 \cdot \mu_2 \cdot \boldsymbol{v}_{\mathsf{id}} + \text{"small"}.
\end{aligned}$$

Different Identities Now we give a flavor of how our multi-identity scheme operates. Suppose $\mathbf{C_1}$ encrypts μ_1 under identity id_1 and $\mathbf{C_2}$ encrypts μ_2 under identity id_2. Let $\boldsymbol{v_1}$ and $\boldsymbol{v_2}$ be the secret key vectors for id_1 and id_2 respectively.

It holds that $\mathbf{C_1} \cdot \boldsymbol{v_1} = \mu_1 \cdot \boldsymbol{v_1} + \boldsymbol{e_1}$ and $\mathbf{C_2} \cdot \boldsymbol{v_2} = \mu_2 \cdot \boldsymbol{v_2} + \boldsymbol{e_2}$ where $\boldsymbol{e_1}, \boldsymbol{e_2} \in \mathbb{Z}_q^N$ are short vectors.

We would like to be able to perform homomorphic computation on both $\mathbf{C_1}$ and $\mathbf{C_2}$ together; that is, use them both as inputs to the same circuit. Here we denote the circuit by $C \in \mathbb{C}$. Suppose we could produce a resulting $2N \times 2N$ ciphertext matrix $\hat{\mathbf{C}}' \in \mathbb{Z}_q^{2N \times 2N}$ that encrypts $\mu' = C(\mu_1, \mu_2)$. More precisely, suppose that

$$\hat{\mathbf{C}}' \cdot \begin{bmatrix} \boldsymbol{v_1} \\ \boldsymbol{v_2} \end{bmatrix} = \mu' \cdot \begin{bmatrix} \boldsymbol{v_1} \\ \boldsymbol{v_2} \end{bmatrix} + \boldsymbol{e'}$$

where $\boldsymbol{e'}$ is "short". Note that the size of $\hat{\mathbf{C}}'$ just depends (polynomially) on the number of distinct identities (2 in this example).

Let $\boldsymbol{v} \in \mathbb{Z}_q^{2N}$ be the vertical concatenation of the two vectors $\boldsymbol{v_1}$ and $\boldsymbol{v_2}$. We could exploit the homomorphic properties described above to obtain $\hat{\mathbf{C}}'$ if we could somehow transform $\mathbf{C_1}$ and $\mathbf{C_2}$ into $2N \times 2N$ matrices $\hat{\mathbf{C_1}}$ and $\hat{\mathbf{C_2}}$ respectively such that $\hat{\mathbf{C_j}} \cdot \boldsymbol{v} = \mu_j \cdot \boldsymbol{v} +$ "small" for $j \in \{1, 2\}$. Technically this transformation turns out to be difficult; we show how to abstractly accomplish it in Section 3 and concretely in Section 4.

2 Preliminaries

2.1 Notation

A quantity is said to be negligible with respect to some parameter λ, written $\mathsf{negl}(\lambda)$, if it is asymptotically bounded from above by the reciprocal of all polynomials in λ. We use the notation $[k]$ for an integer k to denote the set $\{1, \ldots, k\}$.

Distributions For a probability distribution $\mathcal{D}$, we denote by $x \overset{\$}{\leftarrow} \mathcal{D}$ the fact that x is sampled according to $\mathcal{D}$. We overload the notation for a set S i.e. $y \overset{\$}{\leftarrow} S$ denotes that y is sampled uniformly from S. Let $\mathcal{D}_0$ and $\mathcal{D}_1$ be distributions. We denote by $\mathcal{D}_0 \underset{C}{\approx} \mathcal{D}_1$ and the $\mathcal{D}_0 \underset{S}{\approx} \mathcal{D}_1$ the facts that $\mathcal{D}_0$ and $\mathcal{D}_1$ are computationally indistinguishable and statistically indistinguishable respectively.

Definition 1 **(B-bounded distributions (Definition 2 [6])).** *A distribution ensemble $\{D_n\}_{n \in \mathbb{N}}$, supported over the integers, is called B-bounded if*

$$\mathsf{Pr}_{e \overset{\$}{\leftarrow} D_n}[|e| > B] = \mathsf{negl}(n).$$

Matrices and Vectors A matrix $\mathbf{M}$ is denoted by an uppercase symbol written in boldface, and a vector $\boldsymbol{v}$ is denoted by a lowercase symbol written in boldface. The i-th element of $\boldsymbol{v}$ is denoted by v_i. The inner product of two vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}_q^n$ for some dimension n is written as $\langle a, b \rangle$.

2.2 Multi-Identity IBFHE

Definition 2. *A Multi-Identity (Leveled) IBFHE scheme is defined with respect to a message space $\mathcal{M}$, an identity space $\mathcal{I}$, a class of circuits $\mathbb{C} \subseteq \mathcal{M}^* \to \mathcal{M}$ and ciphertext space $\mathcal{C}$. A Multi-Identity IBHE scheme is a tuple of PPT algorithms* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Eval})$ *defined as follows:*

- $\mathsf{Setup}(1^\lambda, L, \mathcal{D})$*:*
  On input (in unary) a security parameter λ, a number of levels L (circuit depth to support) and the number of distinct identities $\mathcal{D}$ that can be tolerated in an evaluation, generate public parameters PP *and a master secret key* MSK. *Output* $(\mathsf{PP}, \mathsf{MSK})$.
- $\mathsf{KeyGen}(\mathsf{MSK}, \mathsf{id})$*:*
  On input master secret key MSK *and an identity* id*: derive and output a secret key* $\mathsf{sk}_{\mathsf{id}}$ *for identity* id.
- $\mathsf{Encrypt}(\mathsf{PP}, \mathsf{id}, m)$*:*
  On input public parameters PP*, an identity* id*, and a message $m \in \mathcal{M}$, output a ciphertext $c \in \mathcal{C}$ that encrypts m under identity* id.
- $\mathsf{Decrypt}(\mathsf{sk}_{\mathsf{id}_1}, \ldots, \mathsf{sk}_{\mathsf{id}_d}, c)$*:*
  On input $d \leq \mathcal{D}$ secret keys $\mathsf{sk}_{\mathsf{id}_1}, \ldots, \mathsf{sk}_{\mathsf{id}_d}$ for (resp.) identities $\mathsf{id}_1, \ldots, \mathsf{id}_d$ and a ciphertext $c \in \mathcal{C}$, output $m' \in \mathcal{M}$ if c is a valid encryption under identities $\mathsf{id}_1, \ldots, \mathsf{id}_d$; output a failure symbol $\perp$ otherwise.
- $\mathsf{Eval}(\mathsf{PP}, C, c_1, \ldots, c_\ell)$*: On input public parameters* PP*, a circuit $C \in \mathbb{C}$ and ciphertexts $c_1, \ldots, c_\ell \in \mathcal{C}$, output an* evaluated ciphertext *$c' \in \mathcal{C}$.*

More precisely, the scheme is required to satisfy the following properties:

- *Over all choices of* $(\mathsf{PP}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda)$*, $d \leq \mathcal{D}$, $\mathsf{id}_1, \ldots, \mathsf{id}_d \in \mathcal{I}$, $C : \mathcal{M}^\ell \to \mathcal{M} \in \{C \in \mathbb{C} : \mathsf{depth}(C) \leq L\}$, $j_1, \ldots, j_\ell \in [d]$, $\mu_1, \ldots, \mu_\ell \in \mathcal{M}$, $c_i \leftarrow \mathsf{Encrypt}(\mathsf{PP}, \mathsf{id}_{j_i}, \mu_i)$ for $i \in [\ell]$, and $c' \leftarrow \mathsf{Eval}(\mathsf{PP}, C, c_1, \ldots, c_\ell)$:*
  - ***Correctness***
    $$\mathsf{Decrypt}(\mathsf{sk}_1, \ldots, \mathsf{sk}_d, c') = C(\mu_1, \ldots, \mu_\ell) \tag{2.1}$$
    for any $\mathsf{sk}_i \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, \mathsf{id}_i)$ for $i \in [k]$
  - ***Compactness***
    $$|c'| \leq \mathsf{poly}(\lambda, L, d) \tag{2.2}$$
    where $d \leq \mathcal{D}$ is the number of distinct identities; that is, $d = |\{j_1, \ldots, i_\ell\}|$.

The size of evaluated ciphertexts in our construction grows with $d \leq \mathcal{D}$.

The security definition for multi-identity IBFHE is the same as that for single-identity IBFHE. In this work, we focus on $\mathsf{IND\text{-}sID\text{-}CPA}$ security whose definition remains the same for the multi-identity setting.

2.3 Learning with Errors

The Learning with Errors (LWE) problem was introduced by Regev [9]. The goal of the computational form of the LWE problem is to determine an n-dimensional secret vector $\boldsymbol{s} \in \mathbb{Z}_q^n$ given a polynomial number of samples $(\boldsymbol{a_i}, b_i) \in \mathbb{Z}_q^{n+1}$ where $\boldsymbol{a_i}$ is uniform over $\mathbb{Z}_q^n$ and $b_i \leftarrow \langle \boldsymbol{a_i}, \boldsymbol{s} \rangle + e_i \in \mathbb{Z}_q$ is the inner product of $\boldsymbol{a_i}$ and $\boldsymbol{s_i}$ perturbed by a small *error* $e_i \in \mathbb{Z}$ that is sampled from a distribution χ over $\mathbb{Z}$. We call the distribution χ an error distribution (or noise distribution). The decision variant of the problem is to distinguish such samples $(\boldsymbol{a_i}, b_i) \in \mathbb{Z}_q^{n+1}$ from uniform vectors over $\mathbb{Z}_q^{n+1}$. The decisional variant is more commonly used in cryptography, and is most relevant to our own work. As a result, without further qualification, when we refer to LWE throughout this thesis we are referring to the decisional variant.

Definition 3 ((Decisional) Learning with Errors (LWE) Problem [9]). *Let λ be a security parameter. For parameters $n = n(\lambda)$, $q = q(\lambda) \geq 2$, and a distribution $\chi = \chi(\lambda)$ over $\mathbb{Z}$, the $LWE_{n,q,\chi}$ problem is to distinguish the following distributions:*

- ***Distribution 0**: The i-th sample $(\boldsymbol{a_i}, b_i) \in \mathbb{Z}_q^{n+1}$ is computed by uniformly sampling $\boldsymbol{a_i} \overset{\$}{\leftarrow} \mathbb{Z}_q^n$ and $b_i \overset{\$}{\leftarrow} \mathbb{Z}_q$.*
- ***Distribution 1**: Generate uniform vector $\boldsymbol{s} \overset{\$}{\leftarrow} \mathbb{Z}_q^n$. The i-th sample $(\boldsymbol{a_i}, b_i) \in \mathbb{Z}_q^{n+1}$ is computed by uniformly sampling $\boldsymbol{a_i} \overset{\$}{\leftarrow} \mathbb{Z}_q^n$, sampling an error value $e_i \overset{\$}{\leftarrow} \chi$ and computing $b_i \leftarrow \langle \boldsymbol{a_i}, \boldsymbol{s} \rangle + e_i$.*

Definition 4 (B-bounded distributions (Definition 2 [6])). *A distribution ensemble $\{D_n\}_{n \in \mathbb{N}}$, supported over the integers, is called B-bounded if*

$$\Pr_{e \overset{\$}{\leftarrow} D_n}[|e| > B] = \mathsf{negl}(n).$$

Definition 5 (GapSVP$_\gamma$). *Let n be a lattice dimension, and let d be a real number. Then $GapSVP_\gamma$ is the problem of deciding whether an n-dimensional lattice has a nonzero vector shorter than d (an algorithm should accept in this case) or no nonzero vector shorter than $\gamma(n) \cdot d$ (an algorithm should reject in this case); an algorithm is allowed to error otherwise.*

Theorem 2 (Theorem 1 [6]). *Let $q = q(n) \in \mathbb{N}$ be either a prime power or a product of small ($\mathsf{poly}(n)$) distinct primes, and let $B \geq \omega(\log n) \cdot \sqrt{n}$. Then there exists an efficient sampleable B-bounded distribution χ such that if there is an efficient algorithm that solves the average-case $LWE_{n,q,\chi}$ problem, then:*

- *There is an efficient quantum algorithm that solves $GapSVP_{\tilde{O}(nq/B)}$ on any n-dimensional lattice.*
- *If $q > \tilde{O}(2^{n/2})$, then there is an efficient classical algorithm for $GapSVP_{\tilde{O}(nq/B)}$ on any n-dimensional lattice.*

2.4 GSW Approximate Eigenvector Cryptosystem

Recall our brief overview of the GSW IBFHE construction earlier from Section 1.3. The following exposition describes this construction in more detail. Note that the public-key GSW scheme is similar to the identity-based variant. As such, to simplify the notation, the following discussion deals with the public-key setting, but the ideas apply to both.

Definition 6 (Section 1.3.2 from [6]). ***B-boundedness:*** *Let $B < q$ be an integer. Let $\mathbf{C}$ be a ciphertext matrix that encrypts μ. Let $\boldsymbol{v}$ be a secret key vector such that $\mathbf{C} \cdot \boldsymbol{v} = \mu \cdot \boldsymbol{v} + \boldsymbol{e}$. Then $\mathbf{C}$ is said to be B-bounded (with respect to $\boldsymbol{v}$) if the magnitude of μ is at most B, the magnitude of all the entries of $\mathbf{C}$ is at most B, and $|||\boldsymbol{e}|||_\infty \leq B$.*

Let $\mathbf{C_1}$ and $\mathbf{C_2}$ be two B-bounded ciphertext matrices. Then $\mathbf{C}^+ = \mathbf{C_1} + \mathbf{C_2}$ is $2B$-bounded. Furthermore, $\mathbf{C}^\times = \mathbf{C_1} \cdot \mathbf{C_2}$ is $(N+1)^{B^2}$-bounded. As the authors of [6] point out, the error grows worse than B^{2^L}, where L is the multiplicative depth of a circuit being evaluated. The modulus q can be chosen to exceed this bound, but we must be careful to ensure that the ratio q/B is at most subexponential in N to guarantee security (see Theorem 2). Hence, only circuits of logarithmic multiplicative depth can be evaluated. This gives us a somewhat-homomorphic scheme.

To evaluate deeper circuits, namely those with polynomial multiplicative depth, we must keep the entries of the ciphertext matrices "small". To achieve this, Gentry, Sahai and Waters propose a technique called *flattening*. Consider the following definition.

Definition 7 (Section 1.3.3 from [6]). ***B-strong-boundedness:*** *Let $B < q$ be an integer. Let $\mathbf{C}$ be a ciphertext matrix that encrypts μ. Let $\boldsymbol{v}$ be a secret key vector such that $\mathbf{C} \cdot \boldsymbol{v} = \mu \cdot \boldsymbol{v} + \boldsymbol{e}$. Then $\mathbf{C}$ is said to be B-strongly-bounded (with respect to $\boldsymbol{v}$) if the magnitude of μ is at most 1, the magnitude of all the entries of $\mathbf{C}$ is at most 1, and $|||\boldsymbol{e}|||_\infty \leq B$.*

An example of a B-strongly-bounded ciphertext is a matrix $\mathbf{C}$ with binary entries that encrypts a plaintext bit $\mu \in \{0, 1\}$, provided the coefficients of its corresponding $\boldsymbol{e}$ vector have magnitude at most B. Let $\mathbf{C_1}$ and $\mathbf{C_2}$ be ciphertext matrices that encrypt $\mu_1 \in \{0, 1\}$ and $\mu_2 \in \{0, 1\}$ respectively. A NAND gate can be evaluated on two ciphertexts $\mathbf{C_1}$ and $\mathbf{C_2}$ as follows:

$$\mathbf{C_3} = \mathbf{I_N} - \mathbf{C_1} \cdot \mathbf{C_2},$$

where $\mathbf{I_N}$ is the $N \times N$ identity matrix. The matrix $\mathbf{C_3}$ encrypts μ_1 NAND $\mu_2 \in \{0, 1\}$. Now if $\mathbf{C_1}$ and $\mathbf{C_2}$ are B-strongly-bounded, then the coefficients of $\mathbf{C_3}$'s error vector have magnitude at most $(N+1)B$, which is in contrast to $(N+1)B^2$ above where $\mathbf{C_1}$ and $\mathbf{C_2}$ were just B-bounded. Suppose there were some way to preserve strong-boundedness in $\mathbf{C_3}$ (i.e. to ensure the magnitude of its entries remained at most 1). Then it would be the case that $\mathbf{C_3}$ is $(N+1)B$-strongly-bounded. As a result, the error level would grow to at most $(N+1)^L B$ when evaluating a circuit of NAND gates of depth L. Therefore it would be possible to evaluate circuits of polynomial depth by letting q/B be subexponential. However, how can we preserve strong-boundedness? It is necessary to introduce some basic operations to help describe how strong boundedness is preserved. These operations serve as useful tools for our own constructions later.

Basic Operations Let $\ell_q = \lfloor \lg q \rfloor + 1$. Let $\boldsymbol{v} \in \mathbb{Z}_q^{m'}$ be a vector of some dimension m' over $\mathbb{Z}_q$. Let $N = m' \cdot \ell_q$.

- $\mathsf{\mathbf{BitDecomp}}(\boldsymbol{v})$: We define an algorithm $\mathsf{BitDecomp}$ that takes as input a vector $\boldsymbol{v} \in \mathbb{Z}_q^{m'}$ and outputs an N-dimensional vector $(v_{1,0}, \dots, v_{1,\ell_q-1}, \dots, v_{k,0}, \dots, v_{k,\ell_q-1})$ where $v_{i,j}$ is the j-th bit in v_i's binary representation (ordered from least significant to most significant).
- $\mathsf{\mathbf{BitDecomp}}^{\mathbf{-1}}(\boldsymbol{v'})$: We define an "inverse" algorithm $\mathsf{BitDecomp}^{-1}$ that takes an N-dimensional vector $\boldsymbol{v'} = (v'_{1,0}, \dots, v'_{1,\ell_q-1}, \dots, v'_{k,0}, \dots, v'_{k,\ell_q-1})$, and outputs a m'-dimensional vector $(\sum_{j=0}^{\ell_q-1} 2^j \cdot v'_{1,j}, \dots, \sum_{j=0}^{\ell_q-1} 2^j \cdot v'_{k,j})$. Note that the input vector $\boldsymbol{v'}$ need not be binary, the algorithm is well-defined for any input vector in $\mathbb{Z}_q^N$.
- $\mathsf{\mathbf{Flatten}}(\boldsymbol{v'})$: The algorithm $\mathsf{Flatten}$ takes as input an N-dimensional vector $\boldsymbol{v'} \in \mathbb{Z}_q^N$ and outputs an N-dimensional binary vector $\mathsf{BitDecomp}(\mathsf{BitDecomp}^{-1}(\boldsymbol{v'}) \in \{0,1\}^N$.
- $\mathsf{\mathbf{Powersof2}}(\boldsymbol{v})$: The algorithm $\mathsf{Powersof2}$ takes a m'-dimensional vector $\boldsymbol{v} \in \mathbb{Z}_q^{m'}$ and outputs an N-dimensional vector $(v_1, 2v_1, \dots, 2^{\ell_q-1}v_1, \dots, v_k, 2v_k, \dots, 2^{\ell_q-1}v_k)$.

We also define $\mathsf{BitDecomp}$, $\mathsf{BitDecomp}^{-1}$ and $\mathsf{Flatten}$ for matrix inputs; in this case, the respective algorithm is applied to each row independently.

We restate the following straightforward facts from [6] (Section 1.3.3): Let $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}_q^{m'}$ be m'-dimensional vectors, and let $\boldsymbol{a'} \in \mathbb{Z}_q^N$ be an N-dimensional vector:

- $\langle \mathsf{BitDecomp}(\boldsymbol{a}), \mathsf{Powersof2}(\boldsymbol{b}) \rangle = \langle \boldsymbol{a}, \boldsymbol{b} \rangle$.
- $\langle \boldsymbol{a'}, \mathsf{Powersof2}(\boldsymbol{b}) \rangle = \langle \mathsf{BitDecomp}^{-1}(\boldsymbol{a'}), \boldsymbol{b} \rangle = \langle \mathsf{Flatten}(\boldsymbol{a'}), \mathsf{Powersof2}(\boldsymbol{b}) \rangle$.

Flattening With the help of $\mathsf{BitDecomp}$, $\mathsf{BitDecomp}^{-1}$, $\mathsf{Powersof2}$ and $\mathsf{Flatten}$, we can tackle the problem of preserving strong boundedness after a NAND operation. In order to make the coefficients of $\mathbf{C_3}$ above have magnitude at most 1, Gentry, Sahai and Waters propose to apply $\mathsf{Flatten}$ to the matrix $\mathbf{C_3}$. Thus, we compute $\mathbf{C}^{\mathsf{NAND}} \leftarrow \mathsf{Flatten}(\mathbf{C_3})$ to produce the output ciphertext of the NAND gate. Now for this to work, the vector $\boldsymbol{v}$ must have a special form. More precisely, $\boldsymbol{v}$ is computed as $\mathsf{Powersof2}(\boldsymbol{s}) \in \mathbb{Z}_q^N$ for some secret key vector $\boldsymbol{s} \in \mathbb{Z}_q^{m'}$ for some m'. Furthermore, the parameter N is defined as $N = m' \cdot \ell_q$, where $\ell_q = \lfloor \lg q \rfloor + 1$. With this form of secret key vector $\boldsymbol{v}$, it holds that $\mathsf{Flatten}(\mathbf{C}) \cdot \boldsymbol{v} = \mathbf{C} \cdot \boldsymbol{v}$ for any $N \times N$ matrix $\mathbf{C}$. So $\mathbf{C}^{\mathsf{NAND}}$ will have entries in $\{0,1\}$ and thus be strongly-bounded.

2.5 GSW Compiler for IBE in the Single-Identity Setting

The Gentry, Sahai and Waters (GSW) compiler from Crypto 2013 [6] (Section 4) allows transformation of an IBE scheme based on the Learning with Errors (LWE) problem into a related IBFHE scheme, provided the IBE scheme satisfies the following properties:

1. **Property 1 (Ciphertext and secret key vectors):** The secret key for identity id and a ciphertext created under id are vectors $\boldsymbol{s}_{\mathsf{id}}, \boldsymbol{c}_{\mathsf{id}} \in \mathbb{Z}_q^{m'}$ for some m'. The first coefficient of $\boldsymbol{s}_{\mathsf{id}}$ is 1.
2. **Property 2 (Small Dot Product):** If $\boldsymbol{c}_{\mathsf{id}}$ encrypts 0, then $\langle \boldsymbol{c}_{\mathsf{id}}, \boldsymbol{s}_{\mathsf{id}} \rangle$ is "small".
3. **Property 3 (Security):** Encryptions of 0 are indistinguishable from uniform vectors over $\mathbb{Z}_q$ under the hardness of LWE.

As noted in [6] all known LWE-based IBE schemes satisfy the above properties e.g: [15–18].

Let $\mathcal{E}$ be an IBE satisfying the Properties 1-3 above. Then $\mathcal{E}$ can be transformed into a single-identity IBFHE scheme $\mathcal{E}'$.

The public parameters PP generated by $\mathcal{E}.\mathsf{Setup}$ includes a modulus q and an integer m' representing the length of both secret key and ciphertext vectors in $\mathcal{E}$. Let $\ell_q = \lfloor \lg q \rfloor + 1$ and $N = m' \times \ell_q$.

To encrypt a message $\mu \in \{0, 1\}$ under identity $\mathsf{id} \in \mathcal{I}$, the encryptor generates N encryptions of 0 using $\mathcal{E}$. More precisely, she computes $\boldsymbol{e_i} \leftarrow \mathcal{E}.\mathsf{Encrypt}(\mathsf{PP}, \mathsf{id}, 0) \in \mathbb{Z}_q^{m'}$ for every $i \in [N]$. The set of N vectors $\boldsymbol{e_1}, \ldots, \boldsymbol{e_N}$ form the rows of an $N \times m'$ matrix $E \in \mathbb{Z}_q^{N \times m'}$. Finally the encryptor computes the $N \times N$ ciphertext matrix $\mathbf{C} \in \{0, 1\}^{N \times N}$ as follows

$$\mathbf{C} \leftarrow \mathsf{Flatten}(\mu \cdot \mathbf{I_N} + \mathsf{BitDecomp}(\mathbf{E}))$$

where $\mathbf{I_N}$ denotes the $N \times N$ identity matrix.

A secret key in $\mathcal{E}'$ for identity id is an N-dimensional vector $\boldsymbol{v}_{\mathsf{id}}$ derived from a secret key $\boldsymbol{s}_{\mathsf{id}}$ for identity id in $\mathcal{E}$. This is computed as $\boldsymbol{v}_{\mathsf{id}} \leftarrow \mathsf{Powersof2}(\boldsymbol{s}_{\mathsf{id}})$. Decryption of a ciphertext $\mathbf{C}$ with $\boldsymbol{v}_{\mathsf{id}}$ is as follows. By construction of $\boldsymbol{v}_{\mathsf{id}}$, it has at least one "large" coefficient; denote this by $v_{\mathsf{id},i}$, To perform decryption, we take the i-th row $\boldsymbol{c_i}$ of matrix $\mathbf{C}$, compute the inner product $x \leftarrow \langle \boldsymbol{c_i}, \boldsymbol{v}_{\mathsf{id}} \rangle = \mu \cdot v_{\mathsf{id},i} + e_i$ and output the plaintext $\mu \leftarrow \lfloor x / v_{\mathsf{id},i} \rceil$. This is correct because

$$\mathbf{C} \cdot \boldsymbol{v}_{\mathsf{id}} = \mu \cdot \boldsymbol{v}_{\mathsf{id}} + \mathbf{E} \cdot \boldsymbol{s}_{\mathsf{id}} = \mu \cdot \boldsymbol{v}_{\mathsf{id}} + \text{“small”}$$

where $\mathbf{E} \cdot \boldsymbol{s}_{\mathsf{id}}$ is "small" as a consequence of Property 2. It is also easy to see that semantic security for $\mathcal{E}'$ follows immediately from the fact that $\mathcal{E}$ satisfies Property 3.

3 A Compiler for Multi-Identity Leveled IBFHE

In this section, we present a new compiler that can transform an LWE-based IBE into a *multi-identity* IBFHE. As we will see, achieving multi-identity IBFHE is far more difficult than single-identity IBFHE.

3.1 Intuition

Suppose $\mathcal{E}$ is an LWE-based IBE that satisfies properties 1 - 3 above. We can apply the GSW compiler to yield an IBFHE scheme $\mathcal{E}'$ in the single-identity setting. Our goal is to construct a compiler for the multi-identity setting. Consider two ciphertexts $\mathbf{C_1}$ and $\mathbf{C_2}$ that encrypt μ_1 and μ_2 under identities id_1 and id_2 respectively. Let $\boldsymbol{s_1}$ and $\boldsymbol{s_2}$ be secret keys in the scheme $\mathcal{E}$ for identities id_1 and id_2 respectively. Accordingly, a decryptor can compute $\boldsymbol{v_1} \leftarrow \mathsf{Powersof2}(\boldsymbol{s_1})$ and $\boldsymbol{v_2} \leftarrow \mathsf{Powersof2}(\boldsymbol{s_2})$. It holds that $\mathbf{C_1} \cdot \boldsymbol{v_1} = \mu_1 \cdot \boldsymbol{v_1} + \boldsymbol{e_1}$ and $\mathbf{C_2} \cdot \boldsymbol{v_2} = \mu_2 \cdot \boldsymbol{v_2} + \boldsymbol{e_2}$ where $\boldsymbol{e_1}, \boldsymbol{e_2} \in \mathbb{Z}_q^N$ are short vectors.

We would like to be able to perform homomorphic computation on both $\mathbf{C_1}$ and $\mathbf{C_2}$ together; that is, use them both as inputs in the same circuit. Here we denote the circuit by $C \in \mathbb{C}$. We expect the size of the resulting ciphertext to grow if $\mathsf{id}_1 \neq \mathsf{id}_2$. This is intuitive because the resulting ciphertext must *encode* information about *both* identities. Assume that $\mathsf{id}_1 \neq \mathsf{id}_2$. The compactness condition of multi-identity IBFHE allows the size of the resulting ciphertext to depend polynomially on the number of *distinct* identities d (in this case $d = 2$). Suppose we could produce a resulting $2N \times 2N$ ciphertext matrix $\mathbf{C}' \in \mathbb{Z}_q^{2N \times 2N}$ that encrypts $\mu' = C(\mu_1, \mu_2)$. More precisely, suppose that

$$\mathbf{C}' \cdot \begin{bmatrix} \boldsymbol{v_1} \\ \boldsymbol{v_2} \end{bmatrix} = \mu' \cdot \begin{bmatrix} \boldsymbol{v_1} \\ \boldsymbol{v_2} \end{bmatrix} + \boldsymbol{e'}$$

where $\boldsymbol{e'}$ is "short". The size of the ciphertext matrix is quadratic in the number of distinct identities, and thus satisfies the compactness condition. How can such a matrix $\mathbf{C}'$ be computed?

The main idea behind our approach is to transform each input ciphertext matrix (i.e. $\mathbf{C_1}$ and $\mathbf{C_2}$ in this example) into a corresponding $dN \times dN$ "expanded matrix" where d is the number of distinct identities (i.e. $d = 2$ in our example).

Consider any input ciphertext matrix $\mathbf{C} \in \mathbb{Z}_q^{N \times N}$ that encrypts a plaintext μ under identity id_1. We denote by $\hat{\mathbf{C}} \in \mathbb{Z}_q^{2N \times 2N}$ its corresponding "expanded matrix". We require this expanded matrix to satisfy

$$\hat{\mathbf{C}} \cdot \begin{bmatrix} \boldsymbol{v_1} \\ \boldsymbol{v_2} \end{bmatrix} = \mu \cdot \begin{bmatrix} \boldsymbol{v_1} \\ \boldsymbol{v_2} \end{bmatrix} + \text{“small”}.$$

Now $\hat{\mathbf{C}}$ can be viewed as consisting of 2×2 submatrices in $\mathbb{Z}_q^{N \times N}$. We denote the submatrix on row i and column j as $\hat{\mathbf{C}}_{i,j} \in \mathbb{Z}_q^{N \times N}$. To satisfy the "top" part of the above equation, it is sufficient to set $\hat{\mathbf{C}}_{1,1} \leftarrow \mathbf{C}$ and $\hat{\mathbf{C}}_{1,2} \leftarrow \mathbf{0}$. To satisfy the "bottom" part of the equation, we need to find matrices $\mathbf{X}, \mathbf{Y} \in \{0, 1\}^{N \times N}$ such that

$$\mathbf{X} \cdot \boldsymbol{v_1} + \mathbf{Y} \cdot \boldsymbol{v_2} = \mu \cdot \boldsymbol{v_2} + \text{“small”}.$$

We refer to a pair of solution matrices $(\mathbf{X}, \mathbf{Y})$ as a "mask" because of the fact that they hide the plaintext μ from a party that does not have a secret key for the recipient identity. In this section, we will abstract over the process of finding solution matrices $\mathbf{X}$ and $\mathbf{Y}$ with respect to arbitrary identities. Towards this goal, we introduce an abstraction called a *masking system*. In short, a masking system allows an encryptor to produce information $U \in \{0,1\}^*$ that allows an evaluator to derive matrices $\mathbf{X}$ and $\mathbf{Y}$ that solve the above equation with respect to any arbitrary identity. Informally, an adversary without a secret key for the *recipient identity* (id_1 in the above example) learns nothing about μ given U, but can still efficiently derive solution matrices $\mathbf{X}$ and $\mathbf{Y}$ with respect to any chosen identity. This notion is formalized in the next section, where we present our compiler. A concrete construction of a masking system is presented in Section 4.2.

3.2 Abstract Compiler

We start by describing an abstract framework for multi-identity IBFHE from Learning with Errors (LWE). Our compiler uses the aforementioned abstraction which we call a *masking system*. An additional prerequisite for an IBE scheme $\mathcal{E}$ (beyond Properties 1-3) to work with our compiler is that there exists a masking system $\mathsf{MS}_{\mathcal{E}}$ for $\mathcal{E}$. First we provide a formal definition of a masking system.

Definition 8. *Let $\mathcal{E}$ be an IBE scheme satisfying Properties 1-3. A* masking system *for $\mathcal{E}$ is a pair of PPT algorithms* $(\mathsf{GenUnivMask}, \mathsf{DeriveMask})$ *defined as follows:*

- $\mathsf{GenUnivMask}(\mathsf{PP}, \mathsf{id}, \mu)$ *takes as input public parameters* PP *for $\mathcal{E}$, an identity* $\mathsf{id} \in \mathcal{I}$ *and a message $\mu \in \{0,1\}$, and outputs $U \in \{0,1\}^*$ (referred to as a* universal mask*).*
- $\mathsf{DeriveMask}(\mathsf{PP}, U, \mathsf{id}')$ *takes as input public parameters* PP *for $\mathcal{E}$, a* universal mask *$U \in \{0,1\}^*$ and an identity* $\mathsf{id}' \in \mathcal{I}$*, and outputs a pair of matrices* $(\mathbf{X}, \mathbf{Y}) \in (\mathbb{Z}_q^{N \times N})^2$.

A masking system $(\mathsf{GenUnivMask}, \mathsf{DeriveMask})$ *must satisfy the following properties:*

- ***Correctness:*** *Let $w(\cdot)$ be a polynomial associated with the masking system. Let $w = w(\lambda)$. We refer to w as the error expansion factor. For correctness, it is required that for any* $(\mathsf{PP}, \mathsf{MSK}) \leftarrow \mathcal{E}.\mathsf{Setup}(1^\lambda)$*, any identities* $\mathsf{id}, \mathsf{id}' \in \mathcal{I}$*, any secret keys* $\boldsymbol{v}_{\mathsf{id}} \leftarrow \mathsf{Powersof2}(\mathcal{E}.\mathsf{KeyGen}(\mathsf{MSK}, \mathsf{id})) \in \mathbb{Z}_q^N$ *and* $\boldsymbol{v}_{\mathsf{id}'} \leftarrow \mathsf{Powersof2}(\mathcal{E}.\mathsf{KeyGen}(\mathsf{MSK}, \mathsf{id}')) \in \mathbb{Z}_q^N$*, and any $\mu \in \{0,1\}$, and over all*
  - $U \leftarrow \mathsf{GenUnivMask}(\mathsf{PP}, \mathsf{id}, \mu)$,
  - $(\mathbf{X}, \mathbf{Y}) \leftarrow \mathsf{DeriveMask}(\mathsf{PP}, U, \mathsf{id}')$

  it holds that
  $$\mathbf{X}\boldsymbol{v}_{\mathsf{id}} + \mathbf{Y}\boldsymbol{v}_{\mathsf{id}'} = \mu \cdot \boldsymbol{v}_{\mathsf{id}'} + \boldsymbol{e} \tag{3.1}$$
  where $\||\boldsymbol{e}\||_\infty \leq w \cdot B$.
- ***Security:*** *The masking system is said to be secure if all PPT adversaries have a negligible advantage in the following modified* $\mathsf{IND}\text{-}X\text{-}\mathsf{CPA}$ *game for $\mathcal{E}$ where $X \in \{\mathsf{sID}, \mathsf{ID}\}$. The only change in the security game is that the adversary is given* $U^* \leftarrow \mathsf{GenUnivMask}(\mathsf{PP}, \mathsf{id}^*, \mu_b)$ *in place of the challenge*

ciphertext in the original game, where $b \xleftarrow{\$} \{0,1\}$ *is the challenger's random bit,* id^* *is the adversary's target identity, and* μ_0 *and* μ_1 *are the challenge messages chosen by the adversary.*

Our compiler can compile an IBE scheme $\mathcal{E}$ into a IBFHE scheme $\mathcal{E}'$ if the following conditions are met (for completeness, we restate Properties 1-3 above):

CP.1: (Ciphertext and secret key vectors): The secret key for identity id and a ciphertext created under id are vectors $\boldsymbol{s}_{\mathsf{id}}, \boldsymbol{c}_{\mathsf{id}} \in \mathbb{Z}_q^{m'}$ for some m'. The first coefficient of $\boldsymbol{s}_{\mathsf{id}}$ is 1.
CP.2: (Small Dot Product): . If $\boldsymbol{c}_{\mathsf{id}}$ encrypts 0 under identity id, then $\boldsymbol{e} = \langle \boldsymbol{c}_{\mathsf{id}}, \boldsymbol{s}_{\mathsf{id}} \rangle$ is "small" where $\boldsymbol{s}_{\mathsf{id}}$ is generated as in CP.1. Formally, $\boldsymbol{e}$ is B-bounded; that is, $\| |\boldsymbol{e}| \|_\infty \leq B$.
CP.3: (Security): Encryptions of 0 are indistinguishable from uniform vectors over $\mathbb{Z}_q$ under the hardness of LWE.
CP.4: (Masking System): There exists a masking system $(\mathsf{GenUnivMask}, \mathsf{DeriveMask})$ for $\mathcal{E}$ meeting the correctness and security conditions of Definition 8.

Let $\mathsf{MS}_{\mathcal{E}} = (\mathsf{MS}_{\mathcal{E}}\mathsf{GenUnivMask}, \mathsf{MS}_{\mathcal{E}}\mathsf{DeriveMask})$ be a *masking system* for $\mathcal{E}$ that satisfies CP.4. A formal description is now given of a generic scheme, which we call mIBFHE, that uses $\mathcal{E}$ and $\mathsf{MS}_{\mathcal{E}}$. We have $\mathsf{mIBFHE.Setup} = \mathcal{E}.\mathsf{Setup}$ and $\mathsf{mIBFHE.KeyGen} = \mathcal{E}.\mathsf{KeyGen}$. The remaining algorithms are described as follows.

Encryption To encrypt a message μ under identity $\mathsf{id} \in \mathcal{I}$, an encryptor performs the following steps. The encryptor computes the universal mask

$$U \leftarrow \mathsf{MS}_{\mathcal{E}}.\mathsf{GenUnivMask}(\mathsf{PP}, \mathsf{id}, \mu)$$

and outputs the ciphertext $\mathsf{CT} := (\mathsf{id}, \mathsf{type} := 0, \mathsf{enc} := U)$. Setting the type component of CT to 0 indicates a "fresh" ciphertext.

Evaluation The evaluator is given as input a circuit $C \in \mathbb{C}$ and a collection of ℓ ciphertexts $\mathsf{CT}_1 := (\mathsf{id}_1, \mathsf{type} := 0, \mathsf{enc} := U_1), \ldots, \mathsf{CT}_\ell := (\mathsf{id}_\ell, \mathsf{type} := 0, \mathsf{enc} := U_\ell)$.

Consider the set of *distinct* identities $I = \{\mathsf{id}_1, \ldots, \mathsf{id}_\ell\}$. Suppose that $|I| = d \leq \ell$ is the number of distinct identities. If $d > \mathcal{D}$ (i.e. the maximum supported number of distinct identities is exceeded), the evaluator aborts the evaluation. For simplicity we re-label the distinct identities as $\mathsf{id}_1, \ldots, \mathsf{id}_d$. Thus, each distinct identity in the collection is associated with a unique index in $[d]$. Before evaluation can be performed, each ciphertext must be "transformed" into a $dN \times dN$ matrix, which we call an *expanded matrix*. This is achieved as follows.

Let $(\mathsf{id}_r, \mathsf{type} := 0, \mathsf{enc} := U)$ be a ciphertext whose associated identity has been assigned the index $r \in [d]$. A matrix $\hat{\mathbf{C}} \in \mathbb{Z}_q^{dN \times dN}$ is formed as follows. Start by setting $\hat{\mathbf{C}}$ to the zero matrix. Now $\hat{\mathbf{C}}$ can be viewed as consisting of $d \times d$ submatrices in $\mathbb{Z}_q^{N \times N}$. We denote the submatrix on row i and column j as $\hat{\mathbf{C}}_{\mathbf{i},\mathbf{j}} \in \mathbb{Z}_q^{N \times N}$.

For $i \in [d]$:

1. Run $(\mathbf{X_i}, \mathbf{Y_i}) \leftarrow \mathsf{MS}_{\mathcal{E}}.\mathsf{DeriveMask}(\mathsf{PP}, U, \mathsf{id}_i)$.
2. Set $\hat{\mathbf{C}}_{\mathbf{i,i}} \leftarrow \mathbf{Y_i}$.
3. Set $\hat{\mathbf{C}}_{\mathbf{i,r}} \leftarrow \mathsf{Flatten}(\hat{\mathbf{C}}_{\mathbf{i,r}} + \mathbf{X_i})$. (The reason for addition here is to handle the special case of $i = r$).

This completes the process for computing the expanded matrix $\hat{\mathbf{C}}$. Consider an example where $r = 1$ and $\mathscr{d} > 2$. The expanded matrix looks like the following:

$$\hat{\mathbf{C}} = \begin{pmatrix} (\mathsf{Flatten}(\mathbf{X_1} + \mathbf{Y_1}) & & & \\ \mathbf{X_2} & \mathbf{Y_2} & & \\ \vdots & & \ddots & \\ \mathbf{X}_{\mathscr{d}} & & & \mathbf{Y}_{\mathscr{d}} \end{pmatrix}$$

Perform the steps above to produce the expanded matrix $\hat{\mathbf{C}}^{(\mathbf{i})}$ for every input ciphertext CT_i. Then the circuit $C \in \mathbb{C}$ is evaluated gate-by-gate (NAND gates) on the expanded matrices to yield a $\mathscr{d}N \times \mathscr{d}N$ matrix $\hat{\mathbf{C}}'$. Suppose each $\hat{\mathbf{C}}^{(\mathbf{i})}$ encrypts $\mu_i \in \{0, 1\}$. Then $\hat{\mathbf{C}}'$ encrypts $C(\mu_1, \ldots, \mu_\ell)$. Finally, the evaluation algorithm outputs the tuple $\mathsf{CT}' := (\mathsf{id}_1, \ldots, \mathsf{id}_{\mathscr{d}}, \mathsf{type} := 1, \mathsf{enc} := \hat{\mathbf{C}}')$. Setting the type component to 1 indicates an evaluated ciphertext. Note that the scheme is 1-hop homomorphic.

Decryption On input a ciphertext $\mathsf{CT} := (\mathsf{id}_1, \ldots, \mathsf{id}_{\mathscr{d}}, \mathsf{type}, \mathsf{enc})$ and a sequence of secret keys $\boldsymbol{v}_{\mathsf{id}_1}, \ldots, \boldsymbol{v}_{\mathsf{id}_{\mathscr{d}}} \in \mathbb{Z}_q^N$ where $\boldsymbol{v}_{\mathsf{id}_i}$ is a secret key for id_i for $i \in [\mathscr{d}]$, the decryptor performs the following steps. Form the column vector $\boldsymbol{v}$ as the vertical concatenation of the column vectors $\boldsymbol{v}_{\mathsf{id}_1}, \ldots, \boldsymbol{v}_{\mathsf{id}_{\mathscr{d}}}$. If $\mathsf{type} = 0$, parse enc as the universal mask U, compute $(\mathbf{X}, \mathbf{Y}) \leftarrow \mathsf{MS}_{\mathcal{E}}.\mathsf{DeriveMask}(\mathsf{PP}, U, \mathsf{id}_1)$ and set $\mathbf{C} \leftarrow \mathbf{X} + \mathbf{Y}$. Else if $\mathsf{type} = 1$, parse enc as $\hat{\mathbf{C}}$ and set $\mathbf{C} \leftarrow \hat{\mathbf{C}}$.

Let i be an index such that $v_i = 2^i \in (q/4, q/2]$. Compute $d_i \leftarrow \langle \boldsymbol{c_i}, \boldsymbol{v} \rangle$ where $\boldsymbol{c_i}$ is the i-th row of $\mathbf{C}$ and output $\mu' \leftarrow \lfloor d_i / v_i \rceil \in \{0, 1\}$. This works to recover the message because as a result of Equation 3.1 (in Definition 8), we have

$$\mathbf{C}\boldsymbol{v} = \mu \cdot \boldsymbol{v} + \boldsymbol{e}$$

with $|||\boldsymbol{e}|||_\infty \leq w \cdot B$, where w is the error expansion factor associated with the masking system $\mathsf{MS}_{\mathcal{E}}$.

Lemma 1. *Let B be a bound such that all freshly encrypted ciphertexts are B-strongly-bounded. Let $\mathscr{D}$ and L be positive integers. If $q > 8 \cdot w \cdot B(\mathscr{D}N + 1)^{L}$[**], then the scheme* mIBFHE *is correct and can evaluate NAND-based Boolean circuits of depth L with any number of distinct identities $\mathscr{d} \leq \mathscr{D}$.*

See Appendix B for the proof of Lemma 1.

[**]Note that N (which depends on n) is itself dependent on $\lg q$. For security, it is required that $q/B = 2^{n^\epsilon}$ for some $\epsilon \in (0, 1)$. A discussion on parameters is provided in Section C.

Theorem 3. *Let $\mathcal{E}$ be an IBE scheme satisfying CP.1 - CP.4. Then $\mathcal{E}$ can be transformed into a multi-identity IBFHE scheme $\mathcal{E}'$.*

Proof. The proof of the theorem is constructive. By CP.4, there exists a masking system $\mathsf{MS}_{\mathcal{E}}$ for $\mathcal{E}$. The multi-identity IBFHE scheme $\mathcal{E}'$ that we obtain is mIBFHE instantiated with $\mathcal{E}$ and $\mathsf{MS}_{\mathcal{E}}$. By Lemma 1, the scheme is correct. CP.4 implies that $\mathcal{E}'$ is IND-X-CPA secure for some $X \in \{\mathsf{sID}, \mathsf{ID}\}$.

4 Concrete Construction of Multi-Identity Leveled IBFHE

To exploit our compiler from the last section to obtain a multi-identity IBFHE, we need to find an LWE-based IBE scheme $\mathcal{E}$ that satisfies CP.1 - CP.4. The major obstacle is finding a scheme for which a secure masking system can be constructed. A natural starting point is the IBE of Cash, Hofheinz, Kiltz and Peikert (CHKP) [18], which is IND-ID-CPA secure in the standard model. This IBE was adapted by Gentry, Sahai and Waters ([6] Appendix A.1) to work with their compiler. There are difficulties however in developing a secure masking system for this IBE. Instead, we consider the IBE of Gentry, Peikert and Vaikuntanathan (GPV) [15]. Unfortunately this scheme is only secure under LWE in the random oracle model. On the plus side, we show that it enjoys the distinction of admitting a secure masking system, and as a consequence of Theorem 3 can be compiled into a multi-identity IBFHE scheme.

4.1 The Gentry, Peikert and Vaikuntanthan (GPV) IBE

In the GPV scheme, the TA needs to use a lookup table [$\star\star\star$] to store secret keys that are issued to users in order to ensure that only a single unique secret key is ever issued for a given identity. This is required for the security proof in the random oracle model.

A hash function $H : \{0,1\}^* \to \mathbb{Z}_q^n$ (modeled as a random oracle in the security proof) is used to map an identity string $\mathsf{id} \in \{0,1\}^*$ to a vector $\boldsymbol{z}_{\mathsf{id}} \in \mathbb{Z}_q^n$. Due to space constraints a formal description of the GPV scheme is deferred to Appendix A. It is easy to see that GPV fulfills CP.1 and CP.2. Furthermore, GPV can be shown to be IND-sID-CPA secure in the random oracle model [15] under LWE, and CP.3 follows from the security proof. It remains to construct a masking system for GPV.

4.2 A masking system for GPV

Relaxation: support for a single identity As a warm up, we consider a relaxation of a masking system. In this relaxation, it is sufficient to find $\mathbf{X}$ and

[$\star\star\star$]Alternatively with the additional assumption of a PRF, a lookup table could be avoided by deterministically deriving secret keys (i.e. obtaining random coins from the PRF).

$\mathbf{Y}$ for only *one* identity id', specified by the encryptor. More precisely, let id be the recipient's identity and let $\mathsf{id}' \neq \mathsf{id}$ be another identity known to the encryptor. Furthermore, let $\boldsymbol{v}$ be a secret key for id and let $\boldsymbol{v'}$ be a secret key for id'. Then the goal is to allow the evaluator to find matrices $\mathbf{X}$ and $\mathbf{Y}$ satisfying

$$\mathbf{X} \cdot \boldsymbol{v} + \mathbf{Y} \cdot \boldsymbol{v'} = \mu \cdot \boldsymbol{v'} + \text{“small”},$$

where μ is the plaintext. For every $i \in N$, we need to find row vectors $\boldsymbol{x_i}$ and $\boldsymbol{y_i}$ with $\langle \boldsymbol{x_i}, \boldsymbol{v} \rangle + \langle \boldsymbol{y_i}, \boldsymbol{v'} \rangle = \mu \cdot \boldsymbol{v'} + \text{“small”}$.

A trivial way to do this is for the encryptor to set $\boldsymbol{x_i} \leftarrow \mathbf{0}$ and $\boldsymbol{y_i} \leftarrow \mathsf{Flatten}((\underbrace{0}_{1,\ldots,i-1}, \mu, \underbrace{0}_{i+1,\ldots,N}) + \mathsf{BitDecomp}(\mathcal{E}.\mathsf{Encrypt}(\mathsf{PP}, \mathsf{id}', 0)) \in \{0,1\}^N$ where the latter is a GSW row encryption of μ under identity id'. Observe that such an $\boldsymbol{x_i}$ and $\boldsymbol{y_i}$ serve as a solution to the above equation. However, it is easy to see that such a trivial solution violates semantic security, since a decryptor with a secret key $\boldsymbol{v'}$ for id' (and no secret key for id) can still recover the plaintext μ.

One strategy for remedying the above approach is to prevent a key holder for identity id' from recovering μ from $\boldsymbol{y_i}$ by appropriately hiding some components of $\boldsymbol{y_i}$. Let us take a look at the structure of $\boldsymbol{y_i}$ when $\mathcal{E}$ is GPV. It is of the form

$$\mathsf{Flatten}((\underbrace{0}_{1,\ldots,i-1}, \mu, \underbrace{0}_{i+1,\ldots,N}) + \mathsf{BitDecomp}((\langle \boldsymbol{z}_{\mathsf{id}'}, \boldsymbol{r} \rangle + e, \boldsymbol{r} \cdot \mathbf{A} + \boldsymbol{f}) \in \mathbb{Z}_q^{m'})$$

where $e \xleftarrow{\$} \chi$, $\boldsymbol{f} \xleftarrow{\$} \chi^m$, $\boldsymbol{r} \xleftarrow{\$} \mathbb{Z}_q^n$ and $\boldsymbol{z}_{\mathsf{id}'} = H(\mathsf{id}') \in \mathbb{Z}_q^n$. Suppose we instead generate $\boldsymbol{y_i}$ as

$$\boldsymbol{y_i} \leftarrow \mathsf{Flatten}((\underbrace{0}_{1,\ldots,i-1}, \mu, \underbrace{0}_{i+1,\ldots,N}) + \mathsf{BitDecomp}((0, \boldsymbol{r} \cdot \mathbf{A} + \boldsymbol{f}).$$

Now what we have done here is effectively set the first ℓ_q components of $\boldsymbol{y_i}$ to 0 with the exception of the special case $i \in [\ell_q]$ which we will handle separately later. As a result of this modification, we will have $\langle \boldsymbol{y_i}, \boldsymbol{v'} \rangle \approx -\langle \boldsymbol{z}_{\mathsf{id}'}, \boldsymbol{r} \rangle + \mu \cdot 2^{i \mod \ell_q}$ (the symbol $\approx$ denotes equality up to “small” differences). Therefore, to cancel out the term $-\langle \boldsymbol{z}_{\mathsf{id}'}, \rangle, weneedtoensurethatweset\boldsymbol{x_i}$ such that $\langle \boldsymbol{x_i}, \boldsymbol{v} \rangle \approx \langle \boldsymbol{z}_{\mathsf{id}'}, \boldsymbol{r} \rangle$.

The approach we take to achieve this is to *blind* the element $\langle \boldsymbol{z}_{\mathsf{id}'}, \boldsymbol{r} \rangle$ with a a GPV encryption of zero under identity id such that it can only be *unblinded* with a secret key for identity id (note that the value cannot be recovered outright; instead a noisy approximation is obtained). For simplicity we define the algorithm Blind which takes an identity id and a value $v \in \mathbb{Z}_q$ and outputs a vector $\mathsf{Flatten}((c_1 + v, c_2, \ldots, c_{m'}))$ where $\boldsymbol{c} \leftarrow \mathcal{E}.\mathsf{Encrypt}(\mathsf{PP}, \mathsf{id}, 0)$. So to provide an $\boldsymbol{x_i}$ counterpart to the vector $\boldsymbol{y_i}$ we generated above, we set $\boldsymbol{x_i} \leftarrow \mathsf{Blind}(\mathsf{id}, (\langle \boldsymbol{z}_{\mathsf{id}'}, \boldsymbol{r} \rangle)$ where $\boldsymbol{r}$ is the vector used in the generation of $\boldsymbol{y_i}$ above. It follows that $\langle \boldsymbol{x_i}, \boldsymbol{v} \rangle + \langle \boldsymbol{y_i}, \boldsymbol{v'} \rangle = \mu \cdot \boldsymbol{v'} + \text{“small”}$.

There are subtleties that we have overlooked. For security reasons, we need to change how we generate $\boldsymbol{x_i}$ and $\boldsymbol{y_i}$ for $i \in [\ell_q]$. This is because for the first ℓ_q components of $\boldsymbol{y_i}$ as generated above, the plaintext μ is not hidden; it is effectively sent in the clear. However we can resolve this issue by setting $\boldsymbol{x_i} \leftarrow$ $\mathsf{Blind}(\mathsf{id}, \mu \cdot 2^{i-1}$ $\boldsymbol{y_i} \leftarrow \mathbf{0}$ and simply setting $\boldsymbol{y_i} \leftarrow \mathbf{0}$.

However there is still a major weakness in this approach. Suppose a decryptor has access to two decryption vectors $\boldsymbol{u'}, \boldsymbol{v'} \in \mathbb{Z}_q^N$ that decrypt ciphertexts with identity id'. For example, the TA might have generated distinct secret key vectors when issuing keys to different parties, and the parties may have shared that information.

It is easy to see that

$$\mathbf{Y} \cdot \boldsymbol{u'} - \mathbf{Y} \cdot \boldsymbol{v'} = \mu \cdot (\boldsymbol{u'} - \boldsymbol{v'}) + \text{“small”},$$

which allows the decryptor to easily determine $\mu \in \{0, 1\}$. Hence a necessary condition for the approach to work is that there be a unique secret key vector for every identity. In fact, this is the primary reason our techniques do not work for ABE. Technically, this restriction means that the system can only support simple classes of access policies, namely classes of predicates with disjoint support sets, which includes the special case of IBE. Fortunately, in the GPV scheme, only a single secret key is ever issued for a given identity.

Support for all identities The algorithm above allows an encryptor to create a secure “mask” for a specific identity that he knows. But how can we create a succinct “universal mask” from which “masks” for arbitrary identities can be derived? To achieve this, we need to take a look at the structure of vector $\boldsymbol{x_i}$ in our masking system, which is constructed as $\boldsymbol{x_i} \leftarrow \mathsf{Blind}(\mathsf{id}, \langle \boldsymbol{z_{\mathsf{id}'}}, \boldsymbol{r} \rangle)$ where id' is known to the encryptor. But what if id' is an arbitrary identity (i.e. not simply one that is known beforehand by the encryptor but one that is chosen by the evaluator at evaluation time)? In this case, we need to obtain an $\boldsymbol{x_i}$ that blinds $\langle \boldsymbol{z_{\mathsf{id}'}}, \boldsymbol{r} \rangle$. Our goal is to include information in the universal mask that we derive so that for any identity id' one can derive an $\boldsymbol{x_i}$ that blinds $\langle \boldsymbol{z_{\mathsf{id}'}}, \boldsymbol{r} \rangle$ where $\boldsymbol{z_{\mathsf{id}'}} = H(\mathsf{id}')$.

Recall the following property of $\mathsf{BitDecomp}$ from Section 2.4:

$$\langle \boldsymbol{z_{\mathsf{id}'}}, \boldsymbol{r} \rangle = \langle \mathsf{BitDecomp}(\boldsymbol{z_{\mathsf{id}'}}), \mathsf{Powersof2}(\boldsymbol{r}) \rangle.$$

Our approach is to *blind* each coefficient of $\mathsf{Powersof2}(\boldsymbol{r})$, whose length is $\ell_q \cdot n$. We produce a matrix $\mathbf{B}^{(i)} \in \mathbb{Z}_q^{(\ell_q \cdot n) \times m'}$ by letting $\boldsymbol{b_j^{(i)}} \leftarrow \mathsf{BitDecomp}^{-1}(\mathsf{Blind}(\mathsf{id}, p_j))$ where p_j be the j-th coefficient of $\mathsf{Powerof2}(\boldsymbol{r})$. Then to generate $\boldsymbol{x_i}$, one computes $\boldsymbol{x_i} \leftarrow \mathsf{Flatten}(\mathsf{BitDecomp}(\boldsymbol{z_{\mathsf{id}'}}) \cdot \mathbf{B}^{(i)})$. Note that $\boldsymbol{y_i}$ is generated as before.

More precisely what we have is shown is how to generate $\mathbf{B}^{(i)}$ and $\boldsymbol{y}_i$ for $i \in [\ell_q]$. Recall that in our previous masking system we generated $\boldsymbol{x_i}$ and $\boldsymbol{y_i}$ differently for $i \in [\ell_q]$. This will also apply here. Instead of computing $\mathbf{B}^{(i)}$ for $i \in [\ell_q]$, we instead merely compute $\boldsymbol{x_i} \leftarrow \mathsf{Blind}(\mathsf{id}, \mu \cdot 2^{i-1})$ and $\boldsymbol{y_i} \leftarrow \mathbf{0}$. This completes the description of our masking system.

We now formally present our masking system for GPV. (which we call $\mathsf{MS}_{\mathsf{GPV}}$). Let $\eta = \ell_q \cdot n$.

$\mathbf{MS_{GPV}.GenUnivMask}(\mathsf{PP}, \mathsf{id}, \mu)$:

```
1. For $i \in [\ell_q]$:
   (a) Set $\boldsymbol{x_i} \leftarrow \mathsf{Blind}(\mathsf{id}, \mu \cdot 2^{i-1})$
   (b) Set $\boldsymbol{y_i} \leftarrow \mathbf{0}$
2. For $\ell_q < i \leq N$:
   (a) Generate $\boldsymbol{r} \xleftarrow{\$} \mathbb{Z}_q^n$ and sample a short error vector $\boldsymbol{e} \xleftarrow{\$} \chi^{m'}$.
   (b) For $j \in [\eta]$ :
       i. Set $\boldsymbol{b_j^{(i)}} \leftarrow \mathsf{BitDecomp}^{-1}(\mathsf{Blind}(\mathsf{id}, p_j)) \in \mathbb{Z}_q^{m'}$ where $p_j$ be the $j$-th
          coefficient of $\mathsf{Powerof2}(\boldsymbol{r})$
   (c) Form matrix $\mathbf{B^{(i)}}$ from rows $\boldsymbol{b_1^{(i)}}, \ldots, \boldsymbol{b_\eta^{(i)}}$.
   (d) Set $\boldsymbol{y_i} \leftarrow \mathsf{Flatten}((\underbrace{0}_{1,\ldots,i-1}, \mu, \underbrace{0}_{i+1,\ldots,N}) + \mathsf{BitDecomp}((0, \boldsymbol{r} \cdot \mathbf{A} + \boldsymbol{f})))$
3. Form matrix $\mathbf{Y}$ from rows $\boldsymbol{y_1}, \ldots, \boldsymbol{y_N}$.
4. Output $U := (\boldsymbol{x_1}, \ldots, \boldsymbol{x_{\ell_q}}, \mathbf{Y}, \mathbf{B^{(\ell_q+1)}}, \ldots, \mathbf{B^{(N)}})$.
```

$\mathsf{MS_{GPV}}$.**DeriveMask**$(\mathsf{PP}, U, \mathsf{id}')$:

```
1. Parse $U$ as $(\boldsymbol{x_1}, \ldots, \boldsymbol{x_{\ell_q}}, \mathbf{Y}, \mathbf{B^{(\ell_q+1)}}, \ldots, \mathbf{B^{(N)}})$.
2. Compute $\boldsymbol{z_{\mathsf{id}'}} \leftarrow H(\mathsf{id}')$.
3. For $\ell_q < i \leq N$:
   (a) Set $\boldsymbol{x_i} \leftarrow \mathsf{Flatten}(\mathsf{BitDecomp}(\boldsymbol{z_{\mathsf{id}'}}) \cdot \mathbf{B}^{(i)})$
4. Form $\mathbf{X} \in \{0,1\}^{N \times N}$ from $\boldsymbol{x_1}, \ldots, \boldsymbol{x_N}$.
5. Output $(\mathbf{X}, \mathbf{Y})$.
```

It is easy to see from the definition of $\mathsf{MS_{GPV}}$.$\mathsf{DeriveMask}$ that the error expansion factor is $w = \eta + 1$. This is because each row in an expanded matrix is formed from a row of $\mathbf{X}$ and a row of $\mathbf{Y}$. But the former decomposes into a sum of η ciphertexts (and hence error terms).

Theorem 4. *[Informal] The masking system* $\mathsf{MS_{GPV}}$ *is selectively secure in the random oracle model (i.e.* $\mathsf{MS_{GPV}}$ *meets the security condition of Definition 8).*

A formal statement of Theorem 4 along with the proof is given in Appendix E. See Appendix A.1 on how to apply the compiler.

5 Multi-Key FHE

If we replace the GPV IBE with the Dual-Regev public-key encryption scheme from [15], then we can obtain a multi-key FHE. The only change in the masking system is that identity vectors (i.e. $\boldsymbol{z_{\mathsf{id}}} = H(\mathsf{id}) \in \mathbb{Z}_q^n$) are replaced with public-key vectors in $\mathbb{Z}_q^n$. As a result, the random oracle H is no longer needed, and security holds in the standard model. Our multi-key scheme is the first to the best of our knowledge that is based on well-established problem such as LWE in the standard model (recall that the scheme from [14] requires the non-standard Decisional Small Polynomial Ratio (DSPR) problem). See the full version [19] of this work for a description of an adaptation of our masking systm to the RLWE setting.

Acknowledgments

We would like to thank the anonymous reviewers of for their helpful comments. The authors would like to thank Fuqun Wang for pointing out errors in an earlier version of this paper.

References

1. Gentry, C.: Fully homomorphic encryption using ideal lattices. Proceedings of the 41st annual ACM Symposium on Theory of Computing STOC 09 (2009) 169
2. Smart, N., Vercauteren, F.: Fully homomorphic encryption with relatively small key and ciphertext sizes. In Nguyen, P., Pointcheval, D., eds.: Public Key Cryptography – PKC 2010. Volume 6056 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2010) 420–443
3. van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully homomorphic encryption over the integers. In Gilbert, H., ed.: Advances in Cryptology – EUROCRYPT 2010. Volume 6110 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2010) 24–43
4. Brakerski, Z., Vaikuntanathan, V.: Fully Homomorphic Encryption from Ring-LWE and Security for Key Dependent Messages, Advances in Cryptology – CRYPTO 2011. Volume 6841 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg, Berlin, Heidelberg (2011) 505–524
5. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) lwe. In Ostrovsky, R., ed.: FOCS, IEEE (2011) 97–106
6. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Canetti, R., Garay, J.A., eds.: CRYPTO (2013). Volume 8042 of Lecture Notes in Computer Science., Springer (2013) 75–92
7. Boneh, D., Franklin, M.K.: Identity-based encryption from the weil pairing. In: CRYPTO '01: Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology, London, UK, Springer-Verlag (2001) 213–229
8. Cocks, C.: An identity based encryption scheme based on quadratic residues. In: Proceedings of the 8th IMA International Conference on Cryptography and Coding, London, UK, Springer-Verlag (2001) 360–363
9. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: STOC '05: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing, New York, NY, USA, ACM (2005) 84–93
10. Clear, M., McGoldrick, C.: Bootstrappable identity-based fully homomorphic encryption. Cryptology ePrint Archive, Report 2014/491 (2014) `http://eprint.iacr.org/`.
11. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: FOCS, IEEE Computer Society (2013) 40–49
12. Clear, M., McGoldrick, C.: Bootstrappable identity-based fully homomorphic encryption. In: Cryptology and Network Security - 13th International Conference, CANS 2014, Heraklion, Crete, Greece, October 22-24, 2014. Proceedings. (2014) 1–19

13. Agrawal, S., Freeman, D.M., Vaikuntanathan, V.: Functional encryption for inner product predicates from learning with errors. In Lee, D.H., Wang, X., eds.: ASIACRYPT. Volume 7073 of Lecture Notes in Computer Science., Springer (2011) 21–40
14. López-Alt, A., Tromer, E., Vaikuntanathan, V.: On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In: Proceedings of the 44th symposium on Theory of Computing. STOC '12, New York, NY, USA, ACM (2012) 1219–1234
15. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: STOC '08: Proceedings of the 40th annual ACM symposium on Theory of computing, New York, NY, USA, ACM (2008) 197–206
16. Agrawal, S., Boneh, D., Boyen, X.: Efficient lattice (H)IBE in the standard model. In: Proc. of Eurocrypt'10. Volume 6110 of LNCS. (2010) 553–572
17. Agrawal, S., Boneh, D., Boyen, X.: Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical ibe. In: CRYPTO. (2010) 98–115
18. Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai trees, or how to delegate a lattice basis. [20] 523–552
19. Clear, M., McGoldrick, C.: Multi-identity and multi-key leveled FHE from learning with errors. IACR Cryptology ePrint Archive **2014** (2014) 798 `http://eprint.iacr.org/2014/798`.
20. Gilbert, H., ed.: Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30 - June 3, 2010. Proceedings. In Gilbert, H., ed.: EUROCRYPT. Volume 6110 of Lecture Notes in Computer Science., Springer (2010)
21. Ajtai, M.: Generating hard instances of the short basis problem. In: ICAL '99: Proceedings of the 26th International Colloquium on Automata, Languages and Programming, London, UK, Springer-Verlag (1999) 1–9
22. Alwen, J., Peikert, C.: Generating shorter bases for hard random lattices. Cryptology ePrint Archive, Report 2008/521 (2008) `http://eprint.iacr.org/2008/521`.
23. Micciancio, D., Peikert, C.: Trapdoors for lattices: Simpler, tighter, faster, smaller. In: EUROCRYPT. (2012) 700–718

A The Gentry, Peikert and Vaikuntanthan (GPV) IBE

Note that this variant has been adapted in the same manner as CHKP in [6] for compatibility with the GSW compiler.

Let $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ be a matrix. We define the lattice $\Lambda^{\perp}(\mathbf{A}) = \{\boldsymbol{x} \in \mathbb{Z}^m : \mathbf{A} \cdot \boldsymbol{x} = \mathbf{0} \mod q\}$ as the space of vectors orthogonal to the rows of $\mathbf{A}$ modulo q. GPV depends on two efficient probabilistic algorithms, which are informally presented as follows:

- **TrapGen**(n, m, q): [21, 22] Generate a statistically uniform matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ together with a short basis $\mathbf{S} \in \mathbb{Z}^{m \times m}$ for $\Lambda^{\perp}(\mathbf{A})$. Output $(\mathbf{A}, \mathbf{S})$.
- **SamplePre**$(\mathbf{S}, \mathbf{A}, \boldsymbol{u})$: [15] Generate a "short" solution $\boldsymbol{x} \in \mathbb{Z}_q^m$ to the equation $\mathbf{A} \cdot \boldsymbol{x} = \boldsymbol{u} \in \mathbb{Z}_q^n$.

See Appendix C.1 for more background on these algorithms. Furthermore, see Appendix C for a discussion on suitable parameter settings.

GPV.Setup(1^λ): Choose parameters $n = n(\lambda)$, $m = m(\lambda)$, $q = q(\lambda)$, a noise distribution $\chi : \mathbb{Z}$. Let $m' = m + 1$. These parameters are implicit in the public parameters PP below. Generate statistically uniform $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ together with a short basis $\mathbf{S} \in \mathbb{Z}^{m \times m}$ of $\Lambda^{\perp}(\mathbf{A})$ by running $(\mathbf{A}, \mathbf{S}) \leftarrow \mathsf{TrapGen}(n, m, q)$. Choose a collision-resistant hash function $H \colon \{0,1\}^t \to \mathbb{Z}_q^n$. Output $\mathsf{PP} := (\mathbf{A}, H)$ and $\mathsf{MSK} := \mathbf{S}$.

$\mathbf{GPV.KeyGen}(\mathsf{MSK}, \mathsf{id} \in \{0,1\}^*)$: If $(\mathsf{id}, \boldsymbol{s}_{\mathsf{id}}) \in \mathsf{store}$, output $\boldsymbol{s}_{\mathsf{id}}$ and abort.

Compute $\boldsymbol{z}_{\mathsf{id}} \leftarrow H(\mathsf{id}) \in \mathbb{Z}_q^n$. Compute $\boldsymbol{w}_{\mathsf{id}} \leftarrow \mathsf{SamplePre}(\mathbf{S}, \mathbf{A}, \boldsymbol{z}_{\mathsf{id}}) \in \mathbb{Z}_q^m$. Set $\boldsymbol{s}_{\mathsf{id}} \leftarrow (1, -\boldsymbol{w}_{\mathsf{id}}) \in \mathbb{Z}_q^{m'}$. Add $(\mathsf{id}, \boldsymbol{s}_{\mathsf{id}})$ to store. Output $\boldsymbol{s}_{\mathsf{id}}$.

Let $\mathbf{A}'_{\mathsf{id}} = \boldsymbol{z}_{\mathsf{id}} \parallel \mathbf{A} \in \mathbb{Z}_q^{m'}$. Observe that $\mathbf{A}'_{\mathsf{id}} \cdot \boldsymbol{s}_{\mathsf{id}} = \mathbf{0} \in \mathbb{Z}_q^n$.

$\mathbf{GPV.Encrypt}(\mathsf{PP}, \mathsf{id} \in \{0,1\}^*, \mu \in \{0,1\})$: Compute $\boldsymbol{z}_{\mathsf{id}} \leftarrow H(\mathsf{id}) \in \mathbb{Z}_q^n$. Let $\mathbf{A}'_{\mathsf{id}} = \boldsymbol{z}_{\mathsf{id}} \parallel \mathbf{A} \in \mathbb{Z}_q^{m'}$. Let $\boldsymbol{\mu} \in \mathbb{Z}_q^{m'}$ be the vector of 0's except with $\mu \cdot \lfloor q/2 \rfloor$ in the first coefficient. Choose random $\boldsymbol{r} \xleftarrow{\$} \mathbb{Z}_q^n$ and small error vector $\boldsymbol{e} \xleftarrow{\$} \chi^{m'}$. Output $\boldsymbol{c}_{\mathsf{id}} \leftarrow \boldsymbol{r} \cdot \mathbf{A}'_{\mathsf{id}} + \boldsymbol{e} + \boldsymbol{\mu} \in \mathbb{Z}_q^{m'}$.

$\mathbf{GPV.Decrypt}(\boldsymbol{s}_{\mathsf{id}}, \boldsymbol{c}_{\mathsf{id}})$: Set $\delta \leftarrow \langle \boldsymbol{c}_{\mathsf{id}}, \boldsymbol{s}_{\mathsf{id}} \rangle \in \mathbb{Z}_q$. If δ is small, output 0; if $\delta - q/2 \mod q$ is small, output 1; otherwise, output $\perp$.

A.1 Proof of Theorem 1

It is now possible to put all the pieces together. In more detail, we can now apply our compiler to the IBE scheme GPV with the masking system $\mathsf{MS}_{\mathsf{GPV}}$ to yield an $\mathsf{IND\text{-}sID\text{-}CPA}$ secure multi-identity IBFHE in the random oracle model.

Theorem 1. *There exists a multi-identity leveled IBFHE scheme that is* $\mathsf{IND\text{-}sID\text{-}CPA}$ *secure in the random oracle model under the hardness of LWE.*

Proof. Let $\mathcal{D}$ be a maximum degree of composition to support, and let L be a desired number of levels. Let λ be the security parameter. We show there exists a leveled IBFHE scheme with maximum degree of composition $\mathcal{D}$, maximum circuit depth L and security parameter λ.

Choose dimension parameter $n = n(\lambda, L)$ and bound $B = B(n)$. Lemma 1 requires

$$q > 8 \cdot w \cdot B(\mathcal{D}N + 1)^L \tag{A.1}$$

to ensure correctness. Note that w is the expansion factor of the masking system. Now the error expansion factor of $\mathsf{MS}_{\mathsf{GPV}}$ is $w = \eta + 1$. But this can be simplified to $N^{\dagger}$. Theorem 4 requires $m \geq 2n \lg q$, and we have $N = (m+1)\lg q$. We need to set q first before setting these parameters (m and N) because of their dependence on q. To do so, q must be expressed without dependence on N. It can be straightforwardly derived from the inequality A.1 that a suitable q is given by

$$q = B \cdot 2^{O(L \lg n\mathcal{D})}$$

with additional care taken to ensure q/B is subexponential in n.

Our parameter settings ensure that the GPV scheme meets CP.1, CP.2 and CP.3, three of the prerequisites for our compiler in Section 3. Furthermore, the masking system $\mathsf{MS}_{\mathsf{GPV}}$ is secure (via Theorem 4). As a result, CP.4 is additionally satisfied. Therefore, Theorem 3 ensures there exists a secure leveled IBFHE scheme, which by virtue of our parameter settings above (which meet Lemma 1), can correctly evaluate L-depth circuits over ciphertexts with at most $\mathcal{D}$ distinct identities .

$^{\dagger} w = \eta + 1 = \ell_q \cdot n + 1 \leq \ell_q \cdot m < N.$

B Proof of Lemma 1

Lemma 1. *Let B be a bound such that all freshly encrypted ciphertexts are B-strongly-bounded. Let $\mathscr{D}$ and L be positive integers. If $q > 8 \cdot w \cdot B(\mathscr{D}N+1)^{L}$*[‡]*, then the scheme* mIBFHE *is correct and can evaluate NAND-based Boolean circuits of depth L with any number of distinct identities $\mathscr{d} \leq \mathscr{D}$.*

Proof. Let the $\mathscr{d} \leq \mathscr{D}$ distinct identities involved in an evaluation be $\mathsf{id}_1, \ldots, \mathsf{id}_{\mathscr{d}}$. Consider an expanded matrix derived from a "fresh" ciphertext $\mathsf{CT} = (\mathsf{id}_i, \mathsf{type} := 0, \mathsf{enc} := U)$ associated with identity id_i for some $i \in [\mathscr{d}]$. Let $\boldsymbol{v}_j$ be a secret key that decrypts ciphertexts with identity id_j for $j \in [\mathscr{d}]$. Let $\hat{\boldsymbol{v}}$ be the column vector consisting of the concatenation of $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{\mathscr{d}}$. Let $\hat{\mathbf{C}}$ be the expanded matrix for CT computed with respect to identities $\mathsf{id}_1, \ldots, \mathsf{id}_{\mathscr{d}}$ and $(\mathbf{X_j}, \mathbf{Y_j}) \leftarrow \mathsf{MS}_{\mathcal{E}}.\mathsf{DeriveMask}(\mathsf{PP}, U, \mathsf{id}_j)$ for $j \in [\mathscr{d}]$. Now by construction, $\hat{\mathbf{C}}$ consists of $\mathscr{d} \times \mathscr{d}$ submatrices in $\mathbb{Z}_q^{N \times N}$. There are 2 non-zero submatrices on $N-1$ rows when $\hat{\mathbf{C}}$ is viewed as $\mathscr{d} \times \mathscr{d}$ matrix over $\mathbb{Z}_q^{N \times N}$, and one non-zero submatrix on the i-th row. The correctness condition for the masking system $\mathsf{MS}_{\mathcal{E}}$ gives us

$$\begin{pmatrix} \mathbf{Y_1} & & \mathbf{X_1} & & \\ & \ddots & \vdots & & \\ & & \mathsf{Flatten}(\mathbf{X_i}+\mathbf{Y_i}) & & \\ & & \vdots & \ddots & \\ & & \mathbf{X}_{\mathscr{d}} & & \mathbf{Y}_{\mathscr{d}} \end{pmatrix} \cdot \begin{bmatrix} \boldsymbol{v_1} \\ \vdots \\ \boldsymbol{v_i} \\ \vdots \\ \boldsymbol{v}_{\mathscr{d}} \end{bmatrix} = \begin{bmatrix} \mathbf{X_1}\boldsymbol{v_1} + \mathbf{Y_1}\boldsymbol{v_1} \\ \vdots \\ \mathbf{X_i}\boldsymbol{v_i} + \mathbf{Y_i}\boldsymbol{v_i} \\ \vdots \\ \mathbf{X}_{\mathscr{d}}\boldsymbol{v}_{\mathscr{d}} + \mathbf{Y}_{\mathscr{d}}\boldsymbol{v}_{\mathscr{d}} \end{bmatrix} = \mu \cdot \begin{bmatrix} \boldsymbol{v_1} \\ \vdots \\ \boldsymbol{v_i} \\ \vdots \\ \boldsymbol{v}_{\mathscr{d}} \end{bmatrix} + \text{'small'} .$$

Since each of these submatrices is B-strongly-bounded, it follows that $\hat{\mathbf{C}} \cdot \hat{\boldsymbol{v}} = \mu \cdot \hat{\boldsymbol{v}} + \hat{\boldsymbol{e}}$ where the coefficients of the error vector $\hat{\boldsymbol{e}}$ are bounded by $w \cdot B$.Therefore, $\hat{\boldsymbol{C}}$ is $w \cdot B$-strongly-bounded. Multiplying two $\mathscr{d}N \times \mathscr{d}N$ expanded matrices in a NAND operation produces a matrix that is $w \cdot B(\mathscr{d}N+1)$-strongly-bounded. After L successive levels, the bound on the error is $w \cdot B(\mathscr{d}N+1)^L$. For correctness of decryption we need $w \cdot B(\mathscr{d}N+1)^L < q/8$. Since we have $\mathscr{d} \leq \mathscr{D}$, it follows that

$$w \cdot B(\mathscr{d}N+1)^L \leq w \cdot B(\mathscr{D}N+1)^L \leq \frac{8 \cdot w \cdot B(\mathscr{D}N+1)^L}{8} < \frac{q}{8}.$$

□

C Parameters for our Scheme

Before discussing how parameters are chosen for our scheme, more background is needed on preimage sampling.

C.1 Background on Preimage Sampling

Let $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ be a matrix. We define the lattice $\Lambda^{\perp}(\mathbf{A}) = \{\boldsymbol{x} \in \mathbb{Z}^m : \mathbf{A} \cdot \boldsymbol{x} = \mathbf{0} \bmod q\}$ as the space of vectors orthogonal to the rows of $\mathbf{A}$ modulo q. There exist

[‡]Note that N (which depends on n) is itself dependent on $\lg q$. For security, it is required that $q/B = 2^{n^{\epsilon}}$ for some $\epsilon \in (0,1)$. A discussion on parameters is provided in Section C.

efficient algorithms to generate a statistically uniform matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ together with a short basis $\mathbf{S} \in \mathbb{Z}^{m \times m}$ for $\Lambda^\perp(\mathbf{A})$ [21, 22]. Such an algorithm will be simply called TrapGen here; that is, we will write $(\mathbf{A}, \mathbf{S}) \leftarrow \mathsf{TrapGen}(n, m, q)$. We denote by $\tilde{\mathbf{S}}$ the Gram-Schmidt orthonormalization of a basis $\mathbf{S}$. Let $\mathfrak{L} = \|\tilde{\mathbf{S}}\|$ be the norm of $\mathbf{S}$. There are instances of TrapGen that achieve $\mathfrak{L} = m^{1+\epsilon}$ for any $\epsilon > 0$ [15], although this has been improved upon in other works [23]. Hence, our setting of $\mathfrak{L}$ later will be a conservative choice.

Let d and t be positive integers with $d \leq t$. Let $\mathbf{B} \in \mathbb{R}^{d \times t}$ be a basis for a d-dimensional lattice $\Lambda(\mathbf{B}) \subset \mathbb{R}^t$. Then the discrete Gaussian distribution on $\Lambda(\mathbf{B})$ with center $\boldsymbol{c} \in \mathbb{R}^t$ and standard deviation $\sigma \in \mathbb{R}$ is denoted by $D_{\Lambda(\mathbf{B}),s,\boldsymbol{c}}$. When $\boldsymbol{c}$ is understood to be zero, the center parameter is omitted.

Gentry, Peikert and Vaikuntanthan [15] describe an algorithm to sample from a discrete Gaussian distribution on an arbitrary lattice. They describe an efficient probabilistic algorithm $\mathsf{SampleD}(\mathbf{B}, \sigma, \boldsymbol{c})$ that samples from a distribution that is statistically close to $D_{\Lambda(\mathbf{B}),\sigma,\boldsymbol{c}}$, provided $\sigma \geq \|\tilde{\mathbf{B}}\| \cdot \omega(\sqrt{\log d})$.

Consider the function $f_A : \mathbb{Z}_q^m \to \mathbb{Z}_q^n$ defined by $f(\boldsymbol{x}) = \mathbf{A} \cdot \boldsymbol{x} \in \mathbb{Z}_q^n$. Given any vector $\boldsymbol{u} \in \mathbb{Z}_q^n$, a *preimage* of $\boldsymbol{u}$ under f_A is any $\boldsymbol{x} \in \mathbb{Z}_q^m$ with $f_A(\boldsymbol{x}) = \boldsymbol{u}$.

It turns out SampleD can be used to efficiently to find *short preimages* $\boldsymbol{x} \in \mathbb{Z}_q^m$ such that $\mathbf{A} \cdot \boldsymbol{x} = \boldsymbol{u} \in \mathbb{Z}_q^n$ for an arbitrary vector $\boldsymbol{u} \in \mathbb{Z}_q^n$. Consider the following algorithm SamplePre from [15]. Note that s is a parameter for which possible settings are given in the next section.

- **SamplePre**$(\mathbf{S}, \mathbf{A}, \boldsymbol{u})$: Find an arbitrary solution $\boldsymbol{t} \in \mathbb{Z}_q^m$ (via linear algebra) such that $\mathbf{A} \cdot \boldsymbol{t} = \boldsymbol{u} \mod q$. Sample a vector $\boldsymbol{e} \xleftarrow{\$} D_{\Lambda^\perp(\mathbf{A}),s,-\boldsymbol{t}}$ by running $\boldsymbol{e} \leftarrow \mathsf{SampleD}(\mathbf{S}, s, -\boldsymbol{t})$, and output the vector $\boldsymbol{x} \leftarrow \boldsymbol{e} + \boldsymbol{t}$.

We remind the reader that there are improved variants of SamplePre in the literature [23].

C.2 Preimage Distribution

We need $s \geq \mathfrak{L} \cdot \omega(\sqrt{\log m})$ to satisfy Theorem 5.9 of [15]. Let $B_{\mathsf{preimage}} \geq \sqrt{n} \cdot s$. Then the probability of the magnitude of any coefficient of a preimage vector exceeding B_{preimage} is exponentially small in n via a standard tail inequality for a normal distribution [§]. One possible setting is $s = \mathfrak{L} \cdot \log m$, and $B_{\mathsf{preimage}} = \sqrt{n} \cdot s$.

C.3 Noise Distribution

To satisfy Theorem 2, we need the noise distribution χ to be B_χ-bounded for some B_χ (to satisfy Theorem 2, we require q/B_χ to be at most subexponential). Setting $\chi \leftarrow D_{\mathbb{Z},r}$ with $r = \log m$ and $B_\chi \geq \sqrt{n} \cdot r$ ensures that χ is B_χ-bounded, since by the aforementioned tail inequality, we have that $\mathsf{Pr}[x \xleftarrow{\$} D_{\mathbb{Z},r}, |x| > B_\chi]$ is exponential in n.

[§]A normal variable with standard deviation σ is within $t \cdot \sigma$ standard deviations of its mean, except with probability at most $\frac{1}{t} \cdot \frac{1}{e^{t^2/2}}$ [15].

C.4 Parameter B (B-strong-boundedness)

"Fresh" ciphertexts in our scheme are B-strongly-bounded. The parameter B is derived from the product of B_{preimage} and B_χ, since when the ciphertext matrix is multiplied by a secret key vector, the resulting error vector is formed from the inner product of the noise vector in the ciphertext (drawn from χ) and the secret key (a sampled preimage). Concretely, with the suggested parameter setting, we have $B = \mathfrak{L} \cdot n \cdot \log^2 m$. It is necessary that q/B_1 is at most subexponential in N. However, our analysis simplifies this by taking q/B to be subexponential; however, since B_{preimage} is polynomial in N, it also holds that q/B_χ is subexponential.

C.5 Sample Parameters and Ciphertext Size

Gentry, Sahai and Waters simplify their analysis by taking n to be a fixed parameter. This is a simplification because q/B must be subexponential in n, and q depends on L; therefore in actuality n depends on L.

Let L be the desired number of levels and let $\mathcal{D}$ be the desired maximum number of distinct identities to support in an evaluation. According to Lemma 1, correctness requires that

$$q > 8 \cdot w \cdot B(\mathcal{D}N + 1)^L. \tag{C.1}$$

In Section C.1, it was mentioned that $\mathfrak{L} \approx m$. Putting this together with the derivation of B above in Section C.4 gives $B = mn \cdot \log^2 m$, where $m \geq 2n \lg q$ from Theorem 4. Choosing B in this way means that it is not too large and allows us to derive $\lg q$ from the inequality C.1 above as follows: $\lg q = O(L(\lg \mathcal{D} + \lg n))$.

Consider the following concrete parameters. Suppose we require a circuit depth of $L = 40$ and a number of distinct identities up to $\mathcal{D} = 100$. We can satisfy the correctness constraint given by C.1 by setting $\lg q = \lceil c \cdot L(\lg \mathcal{D} + \lg L) = 4 \cdot 40(\lg 100 + \lg 40) \rceil = 1915$ (the constant $c = 4$ was chosen to meet the condition) and choosing the dimension to be $n = 2000$. However the size of freshly encrypted ciphertexts in our leveled IBFHE scheme with these parameters is greater than one exabyte (i.e. $> 2^{30}$ gigabytes) per bit of plaintext, which is extremely impractical. This illustrates the impracticality of our scheme, but it also highlights the impracticality at the present time of the GSW leveled IBFHE and ABFHE schemes.

D Size of Evaluated Ciphertexts

As mentioned in the previous section, n is not a fixed parameter that depends solely on the security level λ. Instead n grows with both L and $\mathcal{D}$ because q/B must be subexponential in n to guarantee security. There is an optimization that applies to both our construction and the GSW constructions in terms of the size of evaluated ciphertexts. Decryption only requires a single row of a ciphertext matrix (see Section 3.2), so an evaluated ciphertext can have size $d \cdot N$ where d is the number of distinct identities in the evaluation. Let this vector be denoted by $\hat{\boldsymbol{c}} \in \{0,1\}^{d \cdot N}$. Applying $\mathsf{BitDecomp}^{-1}$, the vector $\boldsymbol{c} \leftarrow \mathsf{BitDecomp}^{-1}(\hat{\boldsymbol{c}}) \in \mathbb{Z}_q^{m'}$ is obtained. As explained in [6], if we include additional information in the public parameters, the technique of modulus reduction [5] can be employed to each coefficient in $\boldsymbol{c}$ so that the size of each coefficient can be made independent of $\mathcal{D}$ and L; their size must still depend on d to ensure correctness, but this is allowed for by the compactness condition. However, while every coefficient can be reduced, the dimension cannot be reduced. This is because the technique of dimension reduction [5] appears to be only compatible with the public key setting since it relies on publishing encryptions of the secret key. We defer the details to [5]. So the length of the ciphertext vector is the length of $\boldsymbol{c}$, namely m', which in turn depends on both L and $\mathcal{D}$. Therefore, technically speaking, our multi-identity IBFHE in addition to both the IBFHE and ABFHE constructions of Gentry, Sahai and Waters are not *leveled* in the strict sense of the size of an evaluated ciphertext being independent of L.

E Proof of Theorem 4

Corollary 1 (Corollary 5.4 [15]). *Let n be a positive integer, and let q be a prime. Let $m \geq 2n \lg q$. Then for all but a $2q^{-n}$ fraction of all $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and for any $s \geq \omega(\sqrt{\log m})$, the distribution of the syndrome $\boldsymbol{u} = \mathbf{A}\boldsymbol{e} \mod q$ is statistically close to uniform over $\mathbb{Z}_q^n$, where $\boldsymbol{e} \sim D_{\mathbb{Z}^m,s}$.*

Theorem 4. *Let n, m, q be chosen to meet Corollary 1. Let χ be a B_χ-bounded distribution where B_χ satisfies Theorem 2. Let* TrapGen *be an algorithm that generates a statistically uniform matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ together with a basis $\mathbf{S} \in \mathbb{Z}^{m\times m}$ such that $\|\tilde{\mathbf{S}}\| \leq \mathfrak{L}$ except with negligible probability. Let $s \geq \mathfrak{L} \cdot \omega(\sqrt{\log m})$. Let the scheme* GPV *be instantiated with* TrapGen *and the* SamplePre *algorithm (with parameter s) described in Section C.1.*

Then the masking system $\mathsf{MS}_{\mathsf{GPV}}$ is selectively secure in the random oracle model (i.e. $\mathsf{MS}_{\mathsf{GPV}}$ meets the security condition of Definition 8) under the hardness of $LWE_{n,q,\chi}$.

Proof. We prove the theorem by means of a hybrid argument.
Game 0: This is the standard selective security game described in Definition 8.

Game 1: The following changes are made in this game. Let $\mathsf{id}^* \in \mathcal{I}$ be the adversary's target identity.

1. The matrix $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$ is generated as uniformly random.
2. The vector $\boldsymbol{z}_{\mathsf{id}^*} \xleftarrow{\$} \mathbb{Z}_q^n$ is generated as uniformly random.
3. The random oracle H is simulated as follows: if the adversary $\mathcal{A}$ queries H on identity $\mathsf{id} \in \mathcal{I}$, run:
   (a) If $\mathsf{id} = \mathsf{id}^*$, then return $\boldsymbol{z}_{\mathsf{id}^*}$.
   (b) Else if $(\mathsf{id}, \boldsymbol{s}_{\mathsf{id}}, \boldsymbol{z}_{\mathsf{id}}) \in \mathsf{store}$, return $\boldsymbol{z}_{\mathsf{id}}$.
   (c) Else sample $\boldsymbol{t}_{\mathsf{id}} \xleftarrow{\$} D_{\mathbb{Z}^{m'-1},s}$, compute $\boldsymbol{z}_{\mathsf{id}} \leftarrow \mathbf{A}\cdot\boldsymbol{t}_{\mathsf{id}} \mod q$, set $\boldsymbol{s}_{\mathsf{id}} \leftarrow (1, -\boldsymbol{t}_{\mathsf{id}}) \in \mathbb{Z}_q^{m'}$, add $(\mathsf{id}, \boldsymbol{s}_{\mathsf{id}}, \boldsymbol{z}_{\mathsf{id}})$ to store and return $\boldsymbol{z}_{\mathsf{id}}$.
   (d) Secret key queries are answered as follows. Suppose $\mathcal{A}$ queries a secret key for identity $\mathsf{id} \neq \mathsf{id}^*$. We assume w.l.o.g. that $\mathcal{A}$ has first queried H on id. In response to the query, $\boldsymbol{s}_{\mathsf{id}}$ is returned where $(\mathsf{id}, \boldsymbol{s}_{\mathsf{id}}, \boldsymbol{z}_{\mathsf{id}}) \in \mathsf{store}$.

We claim that $\mathcal{A}$'s view in Game 0 is statistically close to $\mathcal{A}'s$ view in Game 1. The first two changes above follow immediately from the definition of GPV (in particular, the trapdoor basis generation algorithm employed guarantees that a near uniform $\mathbf{A}$ can be generated). In regard to the simulation of H, Corollary 1 implies that the vector $H(\mathsf{id})$ when $\mathsf{id} \neq \mathsf{id}^*$ is statistically close to uniform. Finally, with regard to the distribution of secret keys, Lemma 5.2 from [15] states that a preimage $\boldsymbol{t}_{\mathsf{id}}$ sampled with SamplePre (with parameter s) in GPV.KeyGen is identically distributed

to $\boldsymbol{t}_{\mathsf{id}} \sim D_{\mathbb{Z}^{m'-1},s}$ conditioned on $\mathbf{A}_{\mathsf{id}} \cdot \boldsymbol{t}_{\mathsf{id}} = \boldsymbol{z}_{\mathsf{id}} \mod q$. It follows that the secret keys $\boldsymbol{s}_{\mathsf{id}}$ in Game 1 have the same distribution as Game 0.
For $i \in [\ell_q]$:
Game $\mathbf{i}+1$: This game is the same as the previous game except that Step 1a of $\mathsf{MS}_{\mathsf{GPV}}.\mathsf{GenUnivMask}$ for iteration i (*only*) is replaced with

$$\boldsymbol{x_i} \leftarrow \mathsf{BitDecomp}(\boldsymbol{t}).$$

where $\boldsymbol{t} \xleftarrow{\$} \mathbb{Z}_q^{m'}$.

Given an LWE instance $\boldsymbol{x^*} \in \mathbb{Z}_q^{m'}$, one can easily generate $\boldsymbol{x_i}$ according to Game i or Game $i+1$. Suppose a distinguisher $\mathcal{D}$ has a non-negligible advantage distinguishing between Game i and Game $i+1$. We can use $\mathcal{D}$ to construct an algorithm $\mathcal{B}$ that can solve an LWE instance. Given an appropriate number of samples from either the distribution $D_0 := \{\{(\boldsymbol{u}_j, \langle \boldsymbol{u}_j, \boldsymbol{s}\rangle + e_j) : \boldsymbol{u_j} \xleftarrow{\$} \mathbb{Z}_q^n, e)_j \xleftarrow{\$} \chi\} : \boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_q^n\}$ or the distribution $D_1 := \{\{(\boldsymbol{u}_j, \boldsymbol{v}_j) : \boldsymbol{u}_j, \boldsymbol{v_j} \xleftarrow{\$} \mathbb{Z}_q^n\}\}$, the $\boldsymbol{u}_j$ are used to construct $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and $\boldsymbol{z}_{\mathsf{id}^*} \in \mathbb{Z}_q^n$. The algorithm $\mathcal{B}$ simulates the random oracle H as explained above, and answers secret key queries in the manner described above. Note that the distribution of $\mathbf{A}$ and $\boldsymbol{z}_{\mathsf{id}^*}$ remain unchanged.

The algorithm $\mathcal{B}$ runs the same variant of $\mathsf{MS}_{\mathsf{GPV}}.\mathsf{GenUnivMask}$ as the previous game. The only difference is that on the i-th iteration, it replaces Step 1a with

$$\boldsymbol{x_i} \leftarrow \mathsf{BitDecomp}(\boldsymbol{x^*} + (\mu \cdot 2^i, 0, \ldots, 0))$$

where $\boldsymbol{x^*} \in \mathbb{Z}_q^{m'}$ is an LWE challenge vector that is either $\boldsymbol{s} \cdot \boldsymbol{z}_{\mathsf{id}^*} \parallel \mathbf{A} + \boldsymbol{e} \in \mathbb{Z}_q^{m'}$ or a uniformly random $\boldsymbol{t^*} \in \mathbb{Z}_q^{m'}$. In the former case, the view is statistically close to Game i whereas the view in the latter case is statistically close to Game $i+1$. It follows that $\mathcal{B}$ can output $\mathcal{D}$'s guess to solve an LWE instance. The games are thus indistinguishable by the hypothesized hardness of LWE.

As a shorthand for Game $(\ell_q+1)+(i-\ell_q-1)\cdot(\eta+1)+j$, we use the notation Game (i,j) for $\ell_q < i \le N$ and $j \in [\eta+1]$.

For $\ell_q < i \le N$:

For $j \in [\eta]$:

- **Game $(\mathbf{i},\mathbf{j})$**: This game is the same as the previous game except that we change the way that the j-th row of $\mathbf{B}^{(\mathrm{i})}$ is generated in $\mathsf{MS}_{\mathsf{GPV}}.\mathsf{GenUnivMask}$. More precisely, Step 2(b)i of algorithm $\mathsf{MS}_{\mathsf{GPV}}.\mathsf{GenUnivMask}$ is replaced with

  $$\boldsymbol{b_j^{(i)}} \leftarrow \mathsf{BitDecomp}(\boldsymbol{t})$$

  with $\boldsymbol{t} \xleftarrow{\$} \mathbb{Z}_q^{m'}$. for the *specific case* of the i-th iteration of the outer loop and the j-th iteration of the inner loop.

  An analogous argument to the argument made above concerning the indistinguishability of Game i and $\mathsf{i}+1$ for $\mathsf{i} \in [\ell_q]$ can be made here to show that a non-negligible advantage distinguishing between the games implies a non-negligible advantage against LWE.

Remark 2. At this stage, note that $\mathbf{B}^{(\mathrm{i})}$ from $\mathsf{MS}_{\mathsf{GPV}}.\mathsf{GenUnivMask}$ is uniform over $\mathbb{Z}_q^{\eta\times m'}$; in particular it does not rely on any $\boldsymbol{r}$ associated with a $\boldsymbol{y_i}$ nor does it rely on μ.

Game $(\mathbf{i}, \eta+\mathbf{1})$: The modification in this game is as follows. Step 2d of $\mathsf{MS}_{\mathsf{GPV}}.\mathsf{GenUnivMask}$ for the i-th iteration is replaced with

$$\boldsymbol{y_i} \leftarrow \mathsf{Flatten}((\mathsf{BitDecomp}((0, \boldsymbol{t})).$$

with $\boldsymbol{t} \xleftarrow{\$} \mathbb{Z}_q^{m'}$

Once again an analogous LWE-based argument to that above shows that one can embed an LWE challenge when generating $\boldsymbol{y_i}$ such that indistinguishability between the games implies a non-negligible advantage against LWE.

We conclude the proof by observing that in Game $(N, \eta+1)$, the plaintext bit μ has been eliminated entirely from the generation of the universal mask U. It follows that an adversary has a zero advantage guessing the challenger's bit b, since no information about b is incorporated in the universal mask U given to the adversary. $\square$