Bootstrappable Identity-Based Fully Homomorphic Encryption

Michael Clear and Ciarán McGoldrick

School of Computer Science and Statistics,
Trinity College Dublin

Abstract. It has been an open problem for a number of years to construct an identity-based fully homomorphic encryption (IBFHE) scheme (first mentioned by Naccache at CHES/CRYPTO 2010). At CRYPTO 2013, Gentry, Sahai and Waters largely settled the problem by presenting leveled IBFHE constructions based on the Learning With Errors problem. However their constructions are not bootstrappable, and as a result, are not "pure" IBFHE schemes. The major challenge with bootstrapping in the identity-based setting is that it must be possible to non-interactively derive from the public parameters an "encryption" of the secret key for an arbitrary identity. All presently-known leveled IBFHE schemes only allow bootstrapping if such an "encryption" of the secret key is supplied out-of-band. In this work, we present a "pure" IBFHE scheme from indistinguishability obfuscation, and extend the result to the attribute-based setting. Our attribute-based scheme is the first to support homomorphic evaluation on ciphertexts with different attributes. Finally, we characterize presently-known leveled IBFHE schemes with a view to developing a "compiler" from a leveled IBFHE scheme to a bootstrappable IBFHE scheme, and sufficient conditions are identified.

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]. Leveled FHE is a relaxation that supports evaluation of circuits of limited (multiplicative) depth. Such a limit L is specified in advance of generating the parameters of the scheme. The size of the parameters along with the size of keys and ciphertexts are allowed to depend on L. In the public-key setting, a leveled FHE scheme can be transformed into a "pure" FHE scheme (i.e. a scheme supporting evaluation of circuits of unlimited depth) via Gentry's *bootstrapping* theorem [1].

In brief, the process of *bootstrapping* entails using the scheme to homomorphically evaluate its own decryption circuit. More precisely, ciphertexts in existing FHE schemes contain a level of "noise". As long as this "noise" remains below a certain threshold, decryption can be performed correctly. The goal of bootstrapping is to return the noise to a reduced level, so homomorphic operations can continue to be performed. This is achieved by publishing encryptions of the secret key bits, and homomorphically evaluating the scheme's decryption circuit on a "noisy" ciphertext to produce a ciphertext with less noise.

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 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 his talk at CHES/Crypto 2010, Naccache [9] mentioned "identity-based fully homomorphic encryption" as an open problem. At Crypto 2013, Gentry, Sahai and Waters presented the first identity-based (leveled) fully homomorphic encryption scheme [6], largely settling the problem raised by Naccache, which had been further explored in [10, 11].

Achieving fully homomorphic encryption (FHE) in the identity-based setting turned out to be quite a tricky problem, for a variety of reasons. Prior to [6], there were two paradigms for constructing leveled FHE:

1. Gentry's original paradigm based on ideals, which was introduced in [1] (works which built on this include [2, 3]); and
2. Brakersi and Vaikuntanathan's paradigm based on the learning with errors (LWE) problem [4,5] entailing techniques such as relinearization, modulus switching and dimension reduction.

It appeared like there was limited potential for obtaining identity-based FHE from the first paradigm because no secure IBE schemes had been constructed with this structure; that is, roughly speaking no IBE scheme associated an identity with an ideal, and a secret key with a "short" generator for that ideal.

The second paradigm appeared more fruitful. Starting with the work of Gentry, Peikert and Vaikuntanathan (GPV) [12], constructions of IBE from LWE had emerged [13–15]. But it was not straightforward to adapt Brakersi and Vaikuntanathan's (BV) ideas to the identity-based setting. The main reason for this is that BV-type FHE relies on having "encryptions" of some secret key information, termed an *evaluation key*. If a user directly supplies this information to an evaluator out-of-band, then evaluation can be accomplished as in BV. IBE schemes where the evaluation key can be generated by the key holder, but cannot be derived non-interactively, have been termed "weak" [10, 11]. Due to the difficulty of non-interactively deriving an "encryption" of secret key information for a given identity (based on public information alone) meant that the BV paradigm also seemed inhospitable to IBE.

Recently Gentry, Sahai and Waters (GSW) [6] developed a new paradigm from LWE where the secret key is an *approximate* eigenvector of a ciphertext. Their construction is both elegant and asymptotically faster than existing FHE schemes. Furthermore, it does not rely on an evaluation key, which means that it can be adapted to support IBE. In fact, a "compiler" was proposed in [6] to transform an LWE-based IBE satisfying certain properties into an identity-based (leveled) fully homomorphic encryption (IBFHE) scheme, and it was noted that several existing LWE-based IBE schemes satisfy the required properties. The resulting IBFHE constructions are leveled i.e. they can evaluate circuits of bounded multiplicative depth (polynomial in the security parameter, and fixed prior to generation of the public parameters). However unlike their public-key counterparts, these constructions are not bootstrappable, since bootstrapping relies on "encryptions" of secret key information, akin to an evaluation key. As such, to the best of our knowledge, there are no known "pure" IBFHE schemes in the literature, since Gentry's bootstrapping theorem from [1] is the only known way of converting a leveled FHE scheme to a "pure" FHE scheme.

In this paper, we identify sufficient conditions for these leveled IBFHE constructions to be bootstrappable, and we construct the first "pure" IBFHE scheme, which we believe finally resolves the question raised by Naccache [9].

1.1 Contributions

Construction of "Pure" IBFHE We construct the first "pure" IBFHE scheme using the technique of "punctured programming" [16], a powerful tool combining an indistinguishability obfuscator [17] with a puncturable pesudorandom function (PRF) [18–20],

A Compiler from leveled IBFHE to "Pure" IBFHE We exploit indistinguishability obfuscation in constructing a compiler from a leveled IBFHE satisfying certain properties to a bootstrappable, and hence "pure", IBFHE. Our main idea is to include in the public parameters an obfuscation of a program (with the master secret key embedded) so that the evaluator can non-interactively derive an "evaluation key" for any identity. Although our compiler falls short of working with arbitrary leveled IBFHE schemes, we establish sufficient conditions for a leveled IBFHE to satisfy in order for it to be bootstrappable. This leads us to an interesting characterization of compatible schemes, which also encompasses our positive result above.

Attribute-Based Fully Homomorphic Encryption (ABFHE) in the Multi-Attribute Setting Sahai and Waters [21] introduced a generalization of IBE known as Attribute-Based Encryption (ABE). In a (key-policy)* ABE scheme, a user Alice encrypts her message with a descriptive tag or *attribute*. The trusted authority issues secret keys for *access policies* to users depending on their credentials. Hence, if a user Bob is given a secret key for a policy f, he can decrypt messages with attributes that satisfy f. More precisely, let c_a be a ciphertext that encrypts the message m with some attribute a. Then Bob can recover the message m if and only if m satisfies his policy f; that is, $f(a) = 1$ (note that policies can be viewed as predicates).

Gentry, Sahai and Waters [6] constructed the first leveled Attribute-Based Fully Homomorphic Encryption scheme (ABFHE). However, their scheme only works in the single-attribute setting. In other words, homomorphic evaluation is supported only for ciphertexts with the same attribute.

*There are other variants such as ciphertext-policy ABE [22], but we focus on key-policy ABE here.

We present the first ABFHE that supports evaluation on ciphertexts with different attributes. We formalize the notion of multi-attribute ABFHE, which can be viewed as an attribute-based analog to the notion of multi-key FHE [23].

Example Scenario
To further illustrate the usefulness of multi-attribute ABFHE, we provide a sketch of an application scenario. Consider a hospital H that avails of the computational facilities of a cloud provider E. Data protection legislation requires the hospital to encrypt all sensitive data stored on third party servers. The hospital deploys attribute-based encryption to manage access to potentially sensitive data. Therefore it manages a "trusted authority" that issues secret keys for access policies to staff in accordance with their roles / credentials. Beyond deploying standard attribute-based encryption, H elects to adopt multi-attribute ABFHE because this allows computation to be performed on encrypted data stored at a third party facility such as E.

Parties such as outside researchers, medical practitioners and internal staff in H are able to encrypt sensitive data with appropriate attributes in order to limit access to authorized staff. For example, a doctor in the maternity unit might encrypt medical data with the attribute "MATERNITY" and a researcher in the cardiology unit might encrypt her data with the attribute "CARDIOLOGY". Suppose both encrypted data sets are sent to the cloud provider E to carry out computational processing on the data (while remaining encrypted). A multi-attribute ABFHE allows E to perform the desired computation homomorphically on both data sets irrespective of the fact that the data sets were encrypted with different attributes.

Suppose a staff member at H has an access policy f defined by

$$f(x) \triangleq x = \text{"MATERNITY" OR } x = \text{"CARDIOLOGY"}.$$

Then this staff member is able to decrypt the result of the computation. This matches our intuition because her policy permits her access to both the data sets used in the computation. However, a member of staff whose access policy permits access to either "MATERNITY" or "CARDIOLOGY" (but no both) should not be able to decrypt the result.

2 Preliminaries

2.1 Notation

A quantity is said to be negligible with respect to some parameter κ, written $\mathsf{negl}(\kappa)$, if it is asymptotically bounded from above by the reciprocal of all polynomials in κ.

For a probability distribution $\mathcal{D}$, we denote by $x \xleftarrow{\$} \mathcal{D}$ the fact that x is sampled according to $\mathcal{D}$. We overload the notation for a set S i.e. $y \xleftarrow{\$} S$ 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.

We use the notation $[k]$ for an integer k to denote the set $\{1, \ldots, k\}$.

2.2 Identity Based Encryption

An Identity Based Encryption (IBE) scheme is a tuple of probabilistic polynomial time (PPT) algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ defined with respect a message space $\mathcal{M}$, an identity space $\mathcal{I}$ and a ciphertext space $\mathcal{C}$ as follows:

- $\mathsf{Setup}(1^\kappa)$:
  On input (in unary) a security parameter κ, 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.

- $\mathsf{Decrypt}(\mathsf{sk}_{\mathsf{id}}, c)$:
  On input a secret key $\mathsf{sk}_{\mathsf{id}}$ for identity id and a ciphertext $c \in \mathcal{C}$, output m' if c is a valid encryption under id; output a failure symbol $\perp$ otherwise.

Indistinguishability under a chosen plaintext attack (IND-CPA) for IBE comes in two flavors - *selective* (denoted by IND-sID-CPA) and *full/adaptive* (denoted by IND-ID-CPA). In the former, the adversary has to choose an identity to attack prior to receiving the public parameters, whereas in the latter, the adversary can make arbitrary secret key queries before choosing a target identity. Formally, the security notions are defined by an adversary $\mathcal{A}$'s success in the following game(s).

- Set $\mathsf{id}^* \leftarrow \perp$.
- **(Selective-security only)**: $\mathcal{A}$ chooses an identity $\mathsf{id}^* \leftarrow \mathcal{I}$ to attack.
- The challenger generates $(\mathsf{PP}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\kappa)$, and gives PP to $\mathcal{A}$.
- **Key Queries (1)**: $\mathcal{A}$ can make queries to an oracle $\mathcal{O}$ defined by

$$\mathcal{O}(\mathsf{id}) = \begin{cases} \mathsf{KeyGen}(\mathsf{MSK}, \mathsf{id}) & \text{if } \mathsf{id} \neq \mathsf{id}^* \\ \perp & \text{otherwise} \end{cases}.$$

- **(Full-security only)**: $\mathcal{A}$ chooses its target identity $\mathsf{id}^* \leftarrow \mathcal{I}$ now.
- **Challenge Phase**: $\mathcal{A}$ chooses two messages $m_0, m_1 \in \mathcal{M}$ and sends them to the challenger.
- The challenger uniformly samples a bit $b \xleftarrow{\$} \{0,1\}$, and returns $c^* \leftarrow \mathsf{Encrypt}(\mathsf{PP}, \mathsf{id}^*, m_b)$.
- **Key Queries (2)**: $\mathcal{A}$ makes additional queries to $\mathcal{O}$.
- **Guess**: $\mathcal{A}$ outputs a guess bit b'.

The adversary is said to win the above game if $b = b'$.

2.3 Identity-Based Fully Homomorphic Encryption (IBFHE)

We first define Leveled IBFHE. This definition is for the single-identity setting, which we consider in this paper. This means that evaluation is supported only for ciphertexts with the same identity.

Definition 1. *A Leveled IBFHE scheme with message space $\mathcal{M}$, identity space $\mathcal{I}$, a class of circuits $\mathbb{C} \subseteq \mathcal{M}^* \to \mathcal{M}$ and ciphertext space $\mathcal{C}$ is a tuple of PPT algorithms* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Eval})$ *defined as follows:*

- $\mathsf{Setup}(1^\kappa, 1^L)$*:*
  On input (in unary) a security parameter κ, and a number of levels L (maximum circuit depth to support) generate public parameters PP *and a master secret key* MSK*. Output* $(\mathsf{PP}, \mathsf{MSK})$*.*
- KeyGen*,* $\mathsf{Encrypt}$ *and* $\mathsf{Decrypt}$ *are defined the same as IBE.*
- $\mathsf{Eval}(\mathsf{PP}, C, c_1, \ldots, c_\ell)$*: 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* $(\mathsf{PP}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\kappa)$*,* $\mathsf{id} \in \mathcal{I}$*,* $C : \mathcal{M}^\ell \to \mathcal{M} \in \{C \in \mathbb{C} : \mathsf{depth}(C) \leq L\}$*,* $\mu_1, \ldots, \mu_\ell \in \mathcal{M}$*,* $c_i \leftarrow \mathsf{Encrypt}(\mathsf{PP}, \mathsf{id}, \mu_i)$ *for* $i \in [\ell]$*, and* $c' \leftarrow \mathsf{Eval}(\mathsf{PP}, C, c_1, \ldots, c_\ell)$*:*
  - ***Correctness***

$$\mathsf{Decrypt}(\mathsf{sk}, c') = C(\mu_1, \ldots, \mu_\ell) \tag{2.1}$$

    for any $\mathsf{sk} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, \mathsf{id})$*.*
  - ***Compactness***

$$|c'| = \mathsf{poly}(\kappa) \tag{2.2}$$

In a leveled fully homomorphic encryption scheme, the size of the public parameters along with the size of keys are allowed to depend on L.

There are different ways to define bootstrapping; the formulation here was chosen to best fit with the results in this paper. We assume without loss of generality that the class of circuits $\mathbb{C}$ supported by the scheme is built from a set of binary operations e.g: $\{\oplus, \odot\}$ i.e. $\oplus : \mathcal{M} \times \mathcal{M} \to \mathcal{M}$ and $\odot : \mathcal{M} \times \mathcal{M} \to \mathcal{M}$.

Definition 2. *A leveled IBFHE is said to be bootstrappable if there exists a pair of PPT algorithms* $(\mathsf{GenBootstrapKey}, \mathsf{Bootstrap})$ *defined as follows:*

- $\mathsf{GenBootstrapKey}(\mathsf{PP}, \mathsf{id})$: *takes as input public parameters* PP *and an identity* id*, and outputs a* bootstrapping key $\mathsf{bk}_{\mathsf{id}}$.
- $\mathsf{Bootstrap}(\mathsf{PP}, \mathsf{bk}_{\mathsf{id}}, c)$ *takes as input public parameters* PP*, a* bootstrapping key $\mathsf{bk}_{\mathsf{id}}$ *for identity* id*, and a ciphertext* $c \in \mathcal{C}$*, and outputs a ciphertext* $c' \in \mathcal{C}$.

Over all $(\mathsf{PP}, \mathsf{MSK})$*: for every pair of ciphertexts* $c_1, c_2 \in \mathcal{C}$*, all identities* id *and all secret keys* $\mathsf{sk}_{\mathsf{id}}$ *and for all* $\circ \in \{\oplus, \odot\}$*:*

$$\mathsf{Decrypt}(\mathsf{sk}_{\mathsf{id}}, \mathsf{Eval}(\circ, \mathsf{Bootstrap}(\mathsf{PP}, \mathsf{id}, c_1), \mathsf{Bootstrap}(\mathsf{PP}, \mathsf{id}, c_2)) = \mathsf{Decrypt}(\mathsf{sk}_{\mathsf{id}}, c_1) \circ \mathsf{Decrypt}(\mathsf{sk}_{\mathsf{id}}, c_2).$$

Informally, what the above definition says is that at least one additional homomorphic operation (either $\oplus$ or $\odot$) can be applied to a pair of "refreshed" (i.e. bootstrapped) ciphertexts before bootstrapping is needed again. For a more thorough discussion on bootstrapping, we refer the reader to [1].

2.4 Indistinguishability Obfuscation

Garg et al. [17] recently introduced a candidate construction of an indistinguishability obfuscator based on multi-linear maps. Many of our constructions in this work depend on the notion of indistinguishability obfuscation. Here we give a brief overview of its syntax and security definition.

Definition 3 (Indistinguishability Obfuscation (Based on Definition 7 from [24])). *A uniform PPT machine* $i\mathcal{O}$ *is called an indistinguishability obfuscator for every circuit class* $\{\mathcal{C}_\kappa\}$ *if the following two conditions are met:*

- ***Correctness:*** *For every* $\kappa \in \mathbb{N}$*, for every* $C \in \mathcal{C}_\kappa$*, for every* x *in the domain of* C*, we have that*

$$\mathsf{Pr}[C'(x) = C(x) : C' \leftarrow i\mathcal{O}(C)] = 1.$$

- ***Indistinguishability:*** *For every* $\kappa \in \mathbb{N}$*, for all pairs of circuits* $C_0, C_1 \in \mathcal{C}_\kappa$*, if* $C_0(x) = C_1(x)$ *for all inputs* x*, then for all PPT adversaries* $\mathcal{A}$*, we have:*

$$|\mathsf{Pr}[\mathcal{A}(i\mathcal{O}(C_0)) = 1]| - |\mathsf{Pr}[\mathcal{A}(i\mathcal{O}(C_1)) = 1]| \leq \mathsf{negl}(\kappa).$$

2.5 Puncturable Pseudorandom Functions

A puncturable pseudorandom function (PRF) is a constrained PRF $(\mathsf{Key}, \mathsf{Eval})$ with an additional PPT algorithm $\mathsf{Puncture}$. Let $n(\cdot)$ and $m(\cdot)$ be polynomials. Our definition here is based on [24] (Definition 3.2). A PRF key K is generated with the PPT algorithm Key which takes as input a security parameter κ. The Eval algorithm is deterministic, and on input a key K and an input string $x \in \{0,1\}^{n(\kappa)}$, outputs a string $y \in \{0,1\}^{m(\kappa)}$.

A puncturable PRF allows one to obtain a "punctured" key $K' \leftarrow \mathsf{Puncture}(K, S)$ with respect to a subset of input strings $S \subset \{0,1\}^{n(\kappa)}$ with $|S| = \mathsf{poly}(\kappa)$. It is required that $\mathsf{Eval}(K, x) = \mathsf{Eval}(K', x) \quad \forall x \in \{0,1\}^{n(\kappa)} \setminus S$, and for any poly-bounded adversary $(\mathcal{A}_1, \mathcal{A}_2)$ with $S \leftarrow \mathcal{A}_1(1^\kappa) \subset \{0,1\}^{n(\kappa)}$ and $|S| = \mathsf{poly}(\kappa)$, any key $K \leftarrow \mathsf{Key}(1^\kappa)$, any $K' \leftarrow \mathsf{Puncture}(K, S)$, and any $x \in S$, it holds that

$$\mathsf{Pr}[\mathcal{A}_2(K', x, \mathsf{Eval}(K, x)) = 1] - \mathsf{Pr}[\mathcal{A}_2(K', x, u) = 1] \leq \mathsf{negl}(\kappa)$$

where $u \xleftarrow{\$} \{0,1\}^{m(\kappa)}$.

3 Construction of "Pure" IBFHE

We now construct a "pure" IBFHE from indistinguishability obfuscation. The main idea is to use the technique of punctured programming, which involves using indistinguishability obfuscation together with a puncturable PRF. In our case, we use the puncturable PRF for the derivation of a user's public key from her identity. Moreover, a unique key pair for a public-key encryption (PKE) scheme can be associated with every identity. If the PKE scheme is also "pure" fully-homomorphic, then we obtain a "pure" IBFHE scheme. Let $\mathcal{E}_{\mathsf{FHE}} := (\mathsf{Gen}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Eval})$ be a public-key FHE. We denote by $\mathcal{PK}_{\mathsf{FHE}}$ and $\mathcal{SK}_{\mathsf{FHE}}$ its public-key and private-key space respectively. Consider the following function $F_{\mathsf{MapPK}} : \mathcal{I} \to \mathcal{PK}_{\mathsf{FHE}}$ that maps an identity $\mathsf{id} \in \mathcal{I}$ to a public key for $\mathcal{E}_{\mathsf{FHE}}$:

Program $F_{\mathsf{MapPK}}(\mathsf{id})$:

1. Compute $r_{\mathsf{id}} \leftarrow \mathsf{PRF.Eval}(K, \mathsf{id})$.
2. Compute $(\mathsf{pk}_{\mathsf{id}}, \mathsf{sk}_{\mathsf{id}}) \leftarrow \mathcal{E}_{\mathsf{FHE}}.\mathsf{Gen}(1^\kappa; r_{\mathsf{id}})$.
3. **Output** $\mathsf{pk}_{\mathsf{id}}$

A formal description of a scheme $\hat{\mathcal{E}}_*$ that uses an obfuscation of F_{MapPK} is as follows.

- $\hat{\mathcal{E}}_*.\mathsf{Setup}(1^\kappa)$: Compute $K \leftarrow \mathsf{PRF.Key}(1^\kappa)$, compute obfuscation $H \leftarrow i\mathcal{O}(F_{\mathsf{MapPK}})$ of F_{MapPK} with K embedded. Output (H, K) (note that H constitutes the public parameters and K constitutes the master secret key).
- $\hat{\mathcal{E}}_*.\mathsf{KeyGen}(K, \mathsf{id})$: Compute $r_{\mathsf{id}} \leftarrow \mathsf{PRF.Eval}(K, \mathsf{id})$, compute $(\mathsf{pk}_{\mathsf{id}}, \mathsf{sk}_{\mathsf{id}}) \leftarrow \mathcal{E}_{\mathsf{FHE}}.\mathsf{Gen}(1^\kappa; r_{\mathsf{id}})$, and output $\mathsf{sk}_{\mathsf{id}}$.
- $\hat{\mathcal{E}}_*.\mathsf{Encrypt}(H, \mathsf{id}, m)$: Compute $\mathsf{pk}_{\mathsf{id}} \leftarrow H(\mathsf{id})$ and output $\mathcal{E}_{\mathsf{FHE}}.\mathsf{Encrypt}(\mathsf{pk}_{\mathsf{id}}, m)$.
- $\hat{\mathcal{E}}_*.\mathsf{Decrypt}(\mathsf{sk}_{\mathsf{id}}, c)$: Output $\mathcal{E}_{\mathsf{FHE}}.\mathsf{Decrypt}(\mathsf{sk}_{\mathsf{id}}, c)$.
- $\hat{\mathcal{E}}_*.\mathsf{Eval}(H, C, c_1, \ldots, c_\ell)$: Compute $\mathsf{pk}_{\mathsf{id}} \leftarrow H(\mathsf{id})$ and output $\mathcal{E}_{\mathsf{FHE}}.\mathsf{Eval}(\mathsf{pk}_{\mathsf{id}}, C, c_1, \ldots, c_\ell)$.

Lemma 1. *Assuming indistinguishability obfuscation, a secure puncturable PRF and an* IND-CPA*-secure public-key FHE scheme $\mathcal{E}_{\mathsf{FHE}}$, the scheme $\hat{\mathcal{E}}_*$ is* IND-sID-CPA *secure.*

Proof. We prove the lemma via a hybrid argument.
Game 0: This is the real system.
Game 1: This is the same as Game 0 except for the following changes. Suppose the adversary chooses id^* as the identity to attack. We compute $K' \leftarrow \mathsf{PRF.Puncture}(K, \mathsf{id}^*)$ and answer secret key requests using K' instead of K.

The adversary cannot detect any difference between the games since for all $\mathsf{id} \neq \mathsf{id}^*$, it holds that $\mathsf{PRF.Eval}(K, \mathsf{id}) = \mathsf{PRF.Eval}(k', \mathsf{id})$.
Game 2 This is the same as Game 1 except that we make the following changes to F_{MapPK}:

- Add before step 1: if $\mathsf{id} = \mathsf{id}^*$, then output $\mathsf{pk}_{\mathsf{id}^*}$ (defined below). Else run steps 1 - 3.
- Replace K with K'.

where $(\mathsf{pk}_{\mathsf{id}^*}, \mathsf{sk}_{\mathsf{id}}) \leftarrow \mathcal{E}_{\mathsf{FHE}}.\mathsf{Gen}(1^\kappa; r_{\mathsf{id}^*})$ and $r_{\mathsf{id}^*} \leftarrow \mathsf{PRF.Eval}(K, \mathsf{id}^*)$.

Observe that the modified function is identical to F_{MapPK}, and due to the security of indistinguishability obfuscation, their respective obfuscations are thus computationally indistinguishable.
Game 3: This is the same as Game 2 except that we change how $\mathsf{pk}_{\mathsf{id}^*}$ is computed. We do this indirectly by changing how r_{id^*} is computed instead. More precisely, we choose a uniformly random string $r_{\mathsf{id}^*} \xleftarrow{\$} \{0,1\}^m$ where m is the length of the pseudorandom outputs of $\mathsf{PRF.Eval}$ i.e. $m = |\mathsf{PRF.Eval}(K, \mathsf{id}^*)|$.

By the security of the puncturable PRF, we have that

$$\{(K', \mathsf{id}^*, \mathsf{PRF.Eval}(K, \mathsf{id}^*)\} \underset{C}{\approx} \{(K', \mathsf{id}^*, r) : r \xleftarrow{\$} \{0,1\}^m)\}.$$

It follows that Game 2 and Game 3 are computationally indistinguishable.
Game 4: This is the same as Game 3 except that we replace the challenge ciphertext with an encryption of a random message. The adversary has a zero advantage in this game.

If a PPT adversary $\mathcal{A}$ can distinguish between Game 3 and Game 4, then there exists a PPT adversary $\mathcal{B}$ that can use $\mathcal{A}$ to attack the $\mathsf{IND\text{-}sID\text{-}CPA}$ security of $\mathcal{E}_{\mathsf{FHE}}$. When $\mathcal{B}$ receives the challenger's public key pk, it sets $\mathsf{pk}_{\mathsf{id}^*} \leftarrow \mathsf{pk}$ where id^* is the target identity chosen by $\mathcal{A}$. Note that $\mathsf{pk}_{\mathsf{id}^*}$ has the same distribution as that from Game 3. Suppose m_0 and m_1 are the messages chosen by $\mathcal{A}$. $\mathcal{B}$ samples a random bit b, and also samples a random message $m' \xleftarrow{\$} \mathcal{M}$, and sends (m_b, m') to the $\mathsf{IND\text{-}CPA}$ challenger, who responds with a challenge ciphertext c^*. Then $\mathcal{B}$ relays c^* to $\mathcal{A}$ as the challenge ciphertext. Let b' denote the random bit chosen by the challenger. If $b' = 0$, then the game is distributed identically to Game 3; otherwise if $b' = 1$ it is distributed identically to Game 4. It follows that any $\mathcal{A}$ with a non-negligible advantage distinguishing between the games contradicts the hypothesized $\mathsf{IND\text{-}CPA}$ security of $\mathcal{E}_{\mathsf{FHE}}$. □

Theorem 1. *Assuming indistinguishability obfuscation, one-way functions and fully homomorphic encryption, there exists an* $\mathsf{IND\text{-}sID\text{-}CPA}$*-secure "pure" IBFHE scheme i.e. an identity-based scheme that can homomorphically evaluate all circuits.*

Proof. The construction $\hat{\mathcal{E}}_*$ is fully homomorphic if the underlying PKE scheme $\mathcal{E}_{\mathsf{FHE}}$ is fully homomorphic. Lemma 1 shows that $\hat{\mathcal{E}}_*$ is $\mathsf{IND\text{-}sID\text{-}CPA}$ secure assuming indistinguishability obfuscation, one-way functions and the $\mathsf{IND\text{-}CPA}$ security of $\mathcal{E}_{\mathsf{FHE}}$. The result follows. □

3.1 Extension to Attribute Based Encryption

The scheme $\hat{\mathcal{E}}_*$ can be extended to an Attribute Based Encryption (ABE) scheme. Recall that in a (key-policy) ABE scheme, an encryptor associates an attribute $a \in \mathbb{A}$ with her message, whereas a decryptor can only successfully decrypt a ciphertext with attribute $a \in \mathbb{A}$ if he holds a secret key for a policy (i.e. a predicate) $f\colon \mathbb{A} \to \{0,1\}$ with $f(a) = 1$. We denote by $\mathbb{F}$ the class of supported policies. Therefore, in an ABE scheme, the trusted authority issues secret keys for policies instead of identities as in IBE. The fundamental difference is that there is no longer a one-to-one correspondence between attributes and policies (which is the case in IBE).

Beyond notationally replacing the set of identities $\mathcal{I}$ with a set of attributes $\mathbb{A}$ in $\hat{\mathcal{E}}_*$, nothing changes for setup, encryption and evaluation. The primary change takes place with respect to key generation. In KeyGen, given a punctured PRF key K' and a policy $f \in \mathbb{F}$, we return as the secret key for f an obfuscation $d_f \leftarrow i\mathcal{O}(F_{\mathsf{Dec}_f})$, where F_{Dec_f} is defined as follows with respect to f:

Program $F_{\mathsf{Dec}_f}(\mathsf{a}, c)$:

1. If $f(a) = 0$, **Output** $\perp$.
2. Compute $r_a \leftarrow \mathsf{PRF.Eval}(K, a)$.
3. Compute $(\mathsf{pk}_a, \mathsf{sk}_a) \leftarrow \mathcal{E}_{\mathsf{FHE}}.\mathsf{Gen}(1^\kappa; r_a)$.
4. **Output** $\mathcal{E}_{\mathsf{FHE}}.\mathsf{Decrypt}(\mathsf{sk}_a, c)$.

Decryption is straightforward: given a secret key for f, namely the obfuscation d_f, a decryptor simply computes $d_f(a, c)$ where a is the attribute associated with ciphertext c. Hence, we obtain an ABFHE for general-purpose policies f.

3.2 Multi-Attribute ABFHE

One of the limitations of our ABFHE construction is that homomorphic evaluation is restricted to the single-attribute setting. In other words, homomorphic evaluation is only supported for ciphertexts with the same attribute. In fact, this is the case for the only known leveled ABFHE in the literature [6].

A related notion to multi-attribute ABFHE was formalized in [25], although we will use a simpler definition here. Recall our illustrated example from the introduction, where a computation was performed on data encrypted under the attribute "MATERNITY" along with data encrypted under the attribute "CARDIOLOGY". In this case, the number of *distinct attributes* was 2.

Let M be an upper bound on the number of distinct attributes supported when homomorphically evaluating a circuit. In multi-attribute ABFHE, the main syntactic change is that the size of an evaluated ciphertext

is allowed to depend on M. Also, M is a parameter that is specified in advance of generating the public parameters.

To be more precise, consider ciphertexts $c_1, \ldots, c_\ell$ passed to the Eval algorithm. Each of the ℓ ciphertexts may have a different attribute. Thus there is at most $k \leq \ell$ distinct attributes in this set. As long as $k \leq M$, the scheme can handle the evaluation of a circuit. Let $c^* \leftarrow \mathsf{Eval}(\mathsf{PP}, C, c_1, \ldots, c_\ell)$ be an evaluated ciphertext, where PP is the public parameters and C is a circuit. It is required that $|c^*| = \mathsf{poly}(\kappa, M)$.

The main idea in [25] is to use multi-key FHE, as introduced by López-Alt, Tromer and Vaikuntanathan [23], to construct a scheme with similar properties to a multi-attribute ABFHE, but with a few limitations. One of these limitations is that only a bounded number of ciphertexts N can be evaluated (where N is fixed a priori), regardless of whether there are less than N distinct attributes. So basically, the scheme from [25] places a limit on the number of independent senders. In contrast, multi-attribute ABFHE permits an unbounded number of independent senders provided the total number of distinct attributes is at most M.

Multi-Attribute ABFHE can be viewed as an attribute-based analog to multi-key FHE from [23]. In multi-key FHE, the size of evaluated ciphertexts depends on an a priori fixed parameter M, which represents the number of independent keys tolerated by the scheme. Hence data encrypted under at most M distinct public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_M$ can be used together in an evaluation.

We exploit multi-key FHE to construct a multi-attribute ABFHE. Our scheme is very similar to our (single-attribute) ABFHE scheme described above in Section 3.1. The main change is that $\mathcal{E}_{\mathsf{FHE}}$ is replaced with a multi-key FHE scheme $\mathcal{E}_{\mathsf{MKFHE}}$ (such as the NTRU-based scheme from [23]). The latter is instantiated with parameter M supplied when generating the public parameters. Suppose a collection of input ciphertexts $c_1, \ldots, c_\ell$ are associated with a set of $k \leq M$ distinct attributes $a_1, \ldots, a_k \in \mathbb{A}$. Hence, an evaluated ciphertext c^* is associated with a *set* $A = \{a_1, \ldots, a_k\}$.

Decryption depends on the intended semantics. One may wish that the decryption process is collaborative i.e. there may not be a single f that satisfies all k attributes, but users may share secret keys for a set of policies $\{f\}$ that "covers all" k attributes. Alternatively, and this is the approach taken in [25], it may be desired that a user can only decrypt c^* if she has a secret key for a policy f that satisfies *all* k attributes. For simplicity, the approach taken here offers security for only the former approach, but it is not difficult to adapt this to achieve security for the latter. Security relies on whether there is an efficient test to check whether a ciphertext has been "fully decrypted". More precisely, we require that $\mathcal{E}_{\mathsf{MKFHE}}.\mathsf{Decrypt}(\{\mathsf{sk}_{a_1}, \ldots, \mathsf{sk}_{a_k}\}, c)$ outputs a message $m \in \mathcal{M}$ if and only if the supplied secret keys $\mathsf{sk}_{a_1}, \ldots, \mathsf{sk}_{a_k}$ are sufficient to *fully* decrypt to a plaintext message, and outputs $\perp$ otherwise. This is the case for the scheme in [23] with all but negligible probability.

Program $F_{\mathsf{Dec}_f}(A, c)$:

1. Set $k \leftarrow |A|$
2. Parse A as $\{a_1, \ldots, a_k\}$.
3. If $k = 0$ or $\exists i \in [k] \quad f(a_i) = 0$, **Output** $\perp$.
4. For $i \in [k]$:
   (a) Compute $r_{a_i} \leftarrow \mathsf{PRF.Eval}(K, a_i)$.
   (b) Compute $(\mathsf{pk}_{a_i}, \mathsf{sk}_{a_i}) \leftarrow \mathcal{E}_{\mathsf{MKFHE}}.\mathsf{Gen}(1^\kappa; r_{a_i})$.
5. **Output** $\mathcal{E}_{\mathsf{MKFHE}}.\mathsf{Decrypt}(\{\mathsf{sk}_{a_1}, \ldots, \mathsf{sk}_{a_k}\}, c)$.

4 A Compiler to Transform a Leveled IBFHE into a "Pure" IBFHE

So far we have obtained "pure" IBFHE, ABFHE and multi-attribute ABFHE schemes. Although these constructions are impractical, they serve as possibility results for these primitives. Next we turn our attention to obtaining a "compiler" to transforming an arbitrary leveled IBFHE into a bootstrappable IBFHE, and as a consequence, a "pure" IBFHE. One of the primary reasons for this is efficiency. One of the reasons our previous constructions are impractical is that they rely on indistinguishability obfuscation for the frequently used process of deriving a public-key for a user's identity. With appropriate parameters, bootstrapping is a process that might be carried out infrequently - or needed only in especially rare occasions. Therefore,

preserving the performance of existing leveled IBFHEs for encryption, decryption and evaluation of "not-too-deep" circuits is desirable. But having the capability to bootstrap, even if expensive, is useful in those cases where evaluation of a deep circuit is needed. This is particularly true in the identity-based setting because keys cannot be generated on a once-off basis as they might be in many applications** of public-key FHE, nor can they be changed as frequently, since all users of the identity-based infrastructure are affected.

Intuitively, the central idea to make a leveled IBFHE scheme bootstrappable is as follows. Firstly, we include an obfuscation of a program in the public parameters. This program "hides" the master secret key (trapdoor) of the scheme. Such a program can use the trapdoor to generate a secret key for an identity, and then use that secret key to output a bootstrapping key that is derived from the secret key. Hence, an evaluator can run the obfuscated program to non-interactively accomplish bootstrapping.

However in order to prove selective security of such a scheme, we need to remove all secret key information for the adversary's target identity. The reason for this is that our obfuscator is not a virtual black-box obfuscator i.e. we cannot argue that the obfuscated program leaks no information about the trapdoor to the adversary. Therefore, certain properties are needed of a leveled IBFHE scheme $\mathcal{E}$ before it is admissible for our "compiler".

4.1 Weakly-bootstrappable IBFHE

Our starting point is leveled IBFHE schemes, such as those constructed via the GSW compiler from [6], that support bootstrapping when given "encryptions" of secret key bits. We refer to such "encryptions" of secret key bits as a *bootstrapping key*. As mentioned in the introduction, there is no known way (in current schemes) to non-interactively derive a bootstrapping key for a given identity from the public parameters alone. The only way bootstrapping can be achieved in such schemes is when a bootstrapping key is passed to the evaluator out-of-band, which breaks an attractive property of IBE, namely that all keys are derivable from the public parameters and a user's identity alone.

We now give a formal definition for a leveled IBFHE that supports bootstrapping when supplied with a bootstrapping key, and we say such a scheme is *weakly bootstrappable*. The main difference between *weakly bootstrappable* and *bootstrappable* (see Definition 2) is that the former requires a secret key for an identity in order to generate a bootstrapping key, whereas the latter only needs an identity. Note that the leveled IBFHEs from [6] are weakly bootstrappable.

Definition 4. *A leveled IBFHE scheme $\mathcal{E}$ is said to be weakly bootstrappable if there exists a pair of PPT algorithms* $(\mathsf{WGenBootstrapKey}, \mathsf{Bootstrap})$ *where* $\mathsf{Bootstrap}$ *is defined as in Definition 2 and* $\mathsf{WGenBootstrapKey}$ *is defined as follows:*

- $\mathsf{WGenBootstrapKey}(\mathsf{PP}, \mathsf{sk}_{\mathsf{id}})$: *takes as input public parameters* PP *and a secret key* $\mathsf{sk}_{\mathsf{id}}$ *for identity* id*, and outputs a* bootstrapping key $\mathsf{bk}_{\mathsf{id}}$.

Like a bootstrappable leveled IBFHE, a weakly-bootstrappable leveled IBFHE requires a circular security assumption to be made to prove $\mathsf{IND\text{-}sID\text{-}CPA}$ security. This is because an adversary is given $\mathsf{bk}_{\mathsf{id}^*}$ for her target identity id^*, which consists of encryptions of secret key bits.

4.2 Single-Point Trapdoor Puncturability

The next requirement we place on a leveled IBFHE to work with our compiler is called single-point trapdoor puncturability. Intuitively, this means that there is a way to "puncture" the master secret key (aka trapdoor) T to yield a *proper subset* $T' \subset T$ that is missing information needed to derive a secret key for a given identity id^*. Furthermore, for all other identities $\mathsf{id} \neq \mathsf{id}^*$, the punctured trapdoor contains enough information to efficiently derive the *same* secret key for id as one would derive with the original trapdoor T, assuming we are given the same randomness. A formal definition will help to elucidate this notion.

**For many applications of public-key FHE, leveled FHE is usually adequate because a new key pair can be generated on a once-off basis for a particular circuit, whose depth is known, and a leveled FHE can be parameterized accordingly.

Definition 5. *An IBE scheme $\mathcal{E}$ is single-point trapdoor-puncturable if there exists PPT algorithms* $\mathsf{TrapPuncture}$ *and* $\mathsf{SimKeyGen}$ *with*

- $\mathsf{TrapPuncture}(T, \mathsf{id}^*)$*: On input trapdoor T and identity id^*, output a "punctured trapdoor" $T' \subset T$ with respect to id^*.*
- $\mathsf{SimKeyGen}(T', \mathsf{id})$*: On input a "punctured trapdoor" T' with respect to some identity id^*, and an identity id, output a secret key for id if $\mathsf{id} \neq \mathsf{id}^*$, and $\perp$ otherwise.*

and these algorithms satisfy the following conditions for any $(\mathsf{PP}, T) \leftarrow \mathcal{E}.\mathsf{Setup}(1^\kappa)$, $\mathsf{id}^* \in \mathcal{I}$ *and* $T' \leftarrow \mathsf{TrapPuncture}(T, \mathsf{id}^*) \subset T$*:*

$$\mathcal{E}.\mathsf{KeyGen}(T, \mathsf{id}) = \mathsf{SimKeyGen}(T', \mathsf{id}) \quad \forall \mathsf{id} \in \mathcal{I} \setminus \{\mathsf{id}^*\}. \tag{4.1}$$

4.3 Our Compiler

Let $\mathcal{E}$ be a *leveled* IBFHE scheme. The required properties that $\mathcal{E}$ must satisfy for compatibility with our compiler are:

Property 1: (**Weakly-Bootstrappable**) $\mathcal{E}$ is weakly-bootstrappable i.e. there exists a pair of PPT algorithms $(\mathsf{WGenBootstrapKey}, \mathsf{Bootstrap})$ satisfying Definition 4.

Property 2: (**Single-Point Trapdoor-Puncturable**) $\mathcal{E}$ is single-point trapdoor-puncturable i.e. there exists a pair of PPT algorithms $(\mathsf{TrapPuncture}, \mathsf{SimKeyGen})$ satisfying Definition 5.

Property 3: (**Indistinguishability given punctured trapdoor**) For all $\mathsf{id} \in \mathcal{I}$ and $m \in \mathcal{M}$: for every $\mathsf{sk}_{\mathsf{id}^*} \leftarrow \mathcal{E}.\mathsf{KeyGen}(T, \mathsf{id}^*)$, and $\mathsf{bk}_{\mathsf{id}^*} \leftarrow \mathsf{WGenBootstrapKey}(\mathsf{PP}, \mathsf{sk}_{\mathsf{id}^*})$, the distributions

$$\{(\mathsf{PP}, T', \mathsf{bk}_{\mathsf{id}^*}, \mathcal{E}.\mathcal{E}.\mathsf{Encrypt}(\mathsf{PP}, \mathsf{id}^*, m)\} \underset{C}{\approx} \{(\mathsf{PP}, T', \mathsf{bk}_{\mathsf{id}^*}, \mathcal{E}.\mathcal{E}.\mathsf{Encrypt}(\mathsf{PP}, \mathsf{id}^*, m')) : m' \xleftarrow{\$} \mathcal{M}\}$$

are computationally indistinguishable.

There are concrete schemes that *almost* meet all three properties. One such example is the leveled IBFHE from Appendix A of [6]. This scheme admits algorithms $(\mathsf{TrapPuncture}, \mathsf{SimKeyGen})$ that satisfy a relaxation of Equation 4.1 in Definition 5, namely the requirement of equality is relaxed to statistical indistinguishability; more precisely it holds that

$$\mathcal{E}.\mathsf{KeyGen}(T, \mathsf{id}) \underset{S}{\approx} \mathsf{SimKeyGen}(T', \mathsf{id}) \quad \forall \mathsf{id} \in \mathcal{I} \setminus \{\mathsf{id}^*\}$$

for any $\mathsf{id} \in \mathcal{I}$. However, we have been unable to find a leveled IBFHE scheme (from the GSW compiler) that meets the stronger condition of Equation 4.1.

Note that it is only necessary that $\mathsf{SimKeyGen}$ run in polynomial time - the essential challenge is to derive some "canonical" secret key for an identity given *less* trapdoor information (but the same randomness). In Appendix A, we discuss the failure of single-point trapdoor-puncturability in current LWE-based leveled IBFHE schemes. It appears new ideas are needed to achieve it.

Formal Description We now proceed with a formal description of a bootstrappable scheme $\hat{\mathcal{E}}_1$ that is constructed using a scheme $\mathcal{E}$ satisfying the above properties. Let $(\mathsf{WGenBootstrapKey}, \mathsf{Bootstrap})$ be a pair of PPT algorithms meeting Property 1.

Consider the following program F_{GenBK} to generate a bootstrapping key:

Program $F_{\mathsf{GenBK}}(\mathsf{id})$:

1. Compute $r_1 \parallel r_2 \leftarrow \mathsf{PRF.Eval}(K, \mathsf{id})$.
2. Compute $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(T, \mathsf{id}; r_1)$.
3. **Output** $\mathsf{WGenBootstrapKey}(\mathsf{PP}_{\mathcal{E}}, \mathsf{sk}_{\mathsf{id}}; r_2)$.

The scheme $\hat{\mathcal{E}}_1$ includes an obfuscation of this program (with key K and trapdoor T) for the purpose of bootstrapping:

- $\hat{\mathcal{E}}_1.\mathsf{Setup}(1^\kappa)$:
  1. Set $(\mathsf{PP}_\mathcal{E}, T) \leftarrow \mathcal{E}.\mathsf{Setup}(1^\kappa)$.
  2. Compute $K \leftarrow \mathsf{PRF.Key}(1^\kappa)$
  3. Compute $\beta \leftarrow i\mathcal{O}(F_{\mathsf{GenBK}})$
  4. Output $(\mathsf{PP} := (\mathsf{PP}_\mathcal{E}, \beta), \mathsf{MSK} := T)$
- $\hat{\mathcal{E}}_1.\mathsf{KeyGen} = \mathcal{E}.\mathsf{KeyGen}$
- $\hat{\mathcal{E}}_1.\mathsf{Encrypt} = \mathcal{E}.\mathsf{Encrypt}$
- $\hat{\mathcal{E}}_1.\mathsf{Decrypt} = \mathcal{E}.\mathsf{Decrypt}$
- **Homomorphic Evaluation:** Evaluation of a gate such as NAND is as in $\mathcal{E}$. Bootstrapping is performed as follows:
  - $\hat{\mathcal{E}}_1.\mathsf{Bootstrap}(\mathsf{PP}, \mathsf{id}, c)$:
    1. Parse PP as $(\mathsf{PP}_\mathcal{E}, \beta)$.
    2. Set $\mathsf{bk}_{\mathsf{id}} \leftarrow \beta(\mathsf{id})$.
    3. Output $\mathsf{Bootstrap}(\mathsf{PP}_\mathcal{E}, \mathsf{bk}_{\mathsf{id}}, c)$.

The main idea is that $\hat{\mathcal{E}}_1$ includes an obfuscation $\beta \leftarrow i\mathcal{O}(F_{\mathsf{GenBK}})$ in its public parameters so an evaluator can derive a bootstrapping key $\mathsf{bk}_{\mathsf{id}}$ for a given identity id and then invoke $\mathsf{Bootstrap}$.

Theorem 2. *Assuming indistinguishability obfuscation, one-way functions, $\hat{\mathcal{E}}_1$ is* IND-sID-CPA *secure if $\mathcal{E}$ satisfies Property 1 - Property 3.*

The theorem is proved in Appendix B.

The construction ($\hat{\mathcal{E}}_*$) from Section 3 satisfies Property 1 - Property 3 .
The construction $\hat{\mathcal{E}}_*$ trivially satisfies Property 1. The reason for this is that it uses an underlying public-key FHE scheme. All public-key FHEs are bootstrappable (in the sense of Definition 2), and being bootstrappable implies being weakly-bootstrappable.

Furthermore $\hat{\mathcal{E}}_*$ is also single-point trapdoor-puncturable (Property 2). In fact, this notion is closely tied with puncturable PRFs. It can be easily seen that by letting $T = K$ and $T' = K'$ with $K \leftarrow \mathsf{PRF.Key}(1^\kappa)$ and $K' \leftarrow \mathsf{PRF.Puncture}(K, \mathsf{id}^*)$, an IBE scheme such as $\hat{\mathcal{E}}_*$ using an obfuscation of a function similar to F_{MapPK} is naturally single-point trapdoor puncturable. Moreover, in this case, we have that $\mathsf{KeyGen} = \mathsf{SimKeyGen} = \mathsf{PRF.Eval}$.

Finally, Property 3 follows from the security of the public-key FHE, since $T' = K'$ gives no information about the secret key for the public-key FHE scheme due to the security of the PRF.

Alternative Approach There is an alternative approach to constructing our compiler which relies on a different requirement to single-point puncturability. However, the bootstrappable schemes that are produced are less efficient. An overview of this approach is given in Appendix C.

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. Cryptology ePrint Archive, Report 2011/344 (2011) `http://eprint.iacr.org/`.

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. Naccache, D.: Is theoretical cryptography any good in practice? (2010) Talk given at CHES 2010 and Crypto 2010.
10. Brakerski, Z., Vaikuntanathan, V.: Efficient Fully Homomorphic Encryption from (Standard) LWE. Cryptology ePrint Archive, Report 2011/344 Version: 20110627:080002 (2011) `http://eprint.iacr.org/`.
11. Clear, M., Hughes, A., Tewari, H.: Homomorphic encryption with access policies: Characterization and new constructions. In Youssef, A., Nitaj, A., Hassanien, A., eds.: Progress in Cryptology AFRICACRYPT 2013. Volume 7918 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2013) 61–87
12. 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
13. 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
14. Agrawal, S., Boneh, D., Boyen, X.: Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical ibe. In: CRYPTO. (2010) 98–115
15. Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai trees, or how to delegate a lattice basis. In: EUROCRYPT. (2010) 523–552
16. Sahai, A., Waters, B.: How to use indistinguishability obfuscation: Deniable encryption, and more. IACR Cryptology ePrint Archive **2013** (2013) 454
17. 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
18. Boneh, D., Waters, B.: Constrained pseudorandom functions and their applications. In Sako, K., Sarkar, P., eds.: ASIACRYPT (2). Volume 8270 of Lecture Notes in Computer Science., Springer (2013) 280–300
19. Boyle, E., Goldwasser, S., Ivan, I.: Functional signatures and pseudorandom functions. In Krawczyk, H., ed.: Public Key Cryptography. Volume 8383 of Lecture Notes in Computer Science., Springer (2014) 501–519
20. Kiayias, A., Papadopoulos, S., Triandopoulos, N., Zacharias, T.: Delegatable pseudorandom functions and applications. In Sadeghi, A.R., Gligor, V.D., Yung, M., eds.: ACM Conference on Computer and Communications Security, ACM (2013) 669–684
21.
22. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: Proceedings of the 2007 IEEE Symposium on Security and Privacy. SP '07, Washington, DC, USA, IEEE Computer Society (2007) 321–334
23. 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
24. Goldwasser, S., Goyal, V., Jain, A., Sahai, A.: Multi-input functional encryption. Cryptology ePrint Archive, Report 2013/727 (2013) `http://eprint.iacr.org/`.
25. Clear, M., McGoldrick, C.: Policy-Based Non-interactive Outsourcing of Computation using multikey FHE and CP-ABE. Proceedings of the 10th Internation Conference on Security and Cryptography, SECRYPT 2013 (2013)
26. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Attribute-based encryption for circuits. In: Proceedings of the 45th annual ACM symposium on Symposium on theory of computing. STOC '13, New York, NY, USA, ACM (2013) 545–554
27. 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
28. Micciancio, D., Peikert, C.: Trapdoors for lattices: Simpler, tighter, faster, smaller. In: EUROCRYPT. (2012) 700–718

A Failure of single-point trapdoor-puncturability in LWE-based IBFHE schemes

All concrete trapdoor-puncturable weakly-bootstrappable IBFHE constructions are based on LWE***. More precisely, they are transformations via the GSW compiler [6] of LWE-based IBEs built from the preimage sampleable

***With the exception of the scheme from Section 3 based on punctured programming.

functions from [12] and the basis extension technique introduced in [15]. These *underlying* IBEs include the Binary Tree Encoding HIBE from [15] and the IBE from [26], and exclude the schemes from [12–14] since these schemes are not trapdoor-puncturable in the sense captured by Definition 5.

We can classify all LWE-based schemes we are aware of that satisfy Definition 5 in the following way, Consider an identity space $\mathcal{I} = \{0,1\}^\ell$ for some fixed integer ℓ. The public parameters in these schemes include matrices $\boldsymbol{A_{1,0}}, \boldsymbol{A_{1,1}}, \ldots, \boldsymbol{A_{\ell,0}}, \boldsymbol{A_{\ell,1}}$. An encryption of a message under an identity $\mathsf{id} = \mathsf{id}_1 \ldots \mathsf{id}_\ell \in \{0,1\}^\ell$ is performed as a dual-Regev [12, 27] encryption with the matrix $\boldsymbol{A_{\mathsf{id}}} = \boldsymbol{A_{1,\mathsf{id}_1}} \parallel \cdots \parallel \boldsymbol{A_{1,\mathsf{id}_\ell}}$. Consider the function $f_{\boldsymbol{A_{\mathsf{id}}}}(\boldsymbol{x}) = \boldsymbol{A_{\mathsf{id}}}\boldsymbol{x}$ mod q, and let $\boldsymbol{u}$ be a public vector. It is a hard problem to find a "short" preimage of $\boldsymbol{u}$ under $f_{\boldsymbol{A_{\mathsf{id}}}}$. Such a preimage $\boldsymbol{e}$ is a secret key for identity $\mathsf{id} \in \{0,1\}^\ell$. In the real system, the matrices $\boldsymbol{A_{i,b}}$ for $i \in [\ell], b \in \{0,1\}$ are generated together with trapdoors $\boldsymbol{T_{i,b}}$ using a trapdoor generation algorithm such as that from [28]. So we have $T = \{\boldsymbol{T_{i,b}}\}_{i\in[\ell],b\in\{0,1\}}$. Owing to basis extension techniques, a "short" preimage in $f^{-1}_{\boldsymbol{A_{\mathsf{id}}}}(\boldsymbol{u})$ can be sampled given only a single $\boldsymbol{T_{j,\mathsf{id}_j}}$ for some $j \in [\ell]$. It follows that for a target identity id^*, the punctured trapdoor is $T' = \{\boldsymbol{T_{i,\mathbf{1}-\mathsf{id}^*_i}}\}_{i\in[\ell]} \subset T$. However, although any trapdoor can be used to sample statistically close "short" preimages, we are not aware of any method for these trapdoors to find the same preimage in polynomial time, even when the same randomness is used. As such, there is no known efficient simulator $\mathsf{SimKeyGen}$ that can satisfy Equation 4.1.

B Proof of Theorem 2

Theorem 2. *Assuming indistinguishability obfuscation, one-way functions, $\hat{\mathcal{E}}_1$ is* IND-sID-CPA *secure if $\mathcal{E}$ satisfies Property 1 - Property 3.*

Proof. We prove the theorem via a hybrid argument.
Game 0: This is the real system.
Game 1: This is the same as Game 0 except for the following changes. Suppose the adversary chooses id^* as the identity to attack. Compute $r_1 \parallel r_2 \leftarrow \mathsf{PRF.Eval}(K, \mathsf{id}^*)$ and compute $\mathsf{bk}_{\mathsf{id}^*} \leftarrow \mathsf{WGenBootstrapKey}(\mathsf{PP}_{\mathcal{E}}, \mathsf{sk}_{\mathsf{id}^*}; r_2)$ where $\mathsf{sk}_{\mathsf{id}^*} \leftarrow \mathsf{KeyGen}(T, \mathsf{id}^*; r_1)$. Make the following changes to F_{GenBK}, which we call F'_{GenBK}, and set $\beta \leftarrow i\mathcal{O}(F'_{\mathsf{GenBK}})$

1. if $\mathsf{id} = \mathsf{id}^*$, then output $\mathsf{bk}_{\mathsf{id}^*}$.
2. Else: Run Step 1 - 3 of F_{GenBK}.

Observe that F_{GenBK} is identical to F'_{GenBK} since $\mathsf{bk}_{\mathsf{id}^*}$ is computed above in the same manner as F_{GenBK}. The games are indistinguishable due to the security of indistinguishability obfuscation.
Game 2 This is the same as Game 1 except with the following changes. Compute a punctured PRF key $K' \leftarrow \mathsf{PRF.Puncture}(K, \mathsf{id}^*)$ that is defined for all strings except the input string id^*, where id^* is the "target" identity chosen by the adversary. Replace all occurrences of K in F'_{GenBK} with K'. We call the modified function F''_{GenBK}.

Observe that $F'_{\mathsf{GenBK}} = F''_{\mathsf{GenBK}}$ because $\mathsf{PRF.Eval}(K, \mathsf{id}) = \mathsf{PRF.Eval}(k', \mathsf{id})$ for all $\mathsf{id} \neq \mathsf{id}^*$. Therefore, the games are indistinguishable due to the security of indistinguishability obfuscation.
Game 3: This is the same as Game 2 except that we change how $\mathsf{bk}_{\mathsf{id}^*}$ is computed. We do this indirectly by changing how $r_1 \parallel r_2 \leftarrow \mathsf{PRF.Eval}(K, \mathsf{id}^*)$ is computed instead. More precisely, we choose a uniformly random string $r'_1 \parallel r'_2 \xleftarrow{\$} \{0,1\}^m$ where m is the length of the pseudorandom outputs of $\mathsf{PRF.Eval}$ i.e. $m = |\mathsf{PRF.Eval}(K, \mathsf{id}^*)|$.

By the security of the puncturable PRF, we have that

$$\{(K', \mathsf{id}^*, \mathsf{PRF.Eval}(K, \mathsf{id}^*)\} \overset{}{\approx}_C \{(K', \mathsf{id}^*, r) : r \xleftarrow{\$} \{0,1\}^m)\}.$$

It follows that Game 2 and Game 3 are computationally indistinguishable.
Game 4: This is the same as Game 3 except that we make the following changes. We compute a punctured trapdoor $T' \subset T$ using the $\mathsf{TrapPuncture}$ algorithm (which exists by Property 2) i.e. $T' \leftarrow \mathsf{TrapPuncture}(T, \mathsf{id}^*)$. We answer secret key queries with $\mathsf{SimKeyGen}(T', \cdot)$. The games cannot be distinguished by an adversary as a result of Equation 4.1 in Definition 5 (single-point trapdoor puncturability).
Game 5: The only change in this game is that we set $\beta \leftarrow i\mathcal{O}(F'''_{\mathsf{GenBK}})$ where F'''_{GenBK} is the same as F''_{GenBK} except $\mathsf{sk}_{\mathsf{id}}$ is computed as

$$\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{SimKeyGen}(T', \mathsf{id}; r_1).$$

As a result of Equation 4.1 in Definition 5 (single-point trapdoor puncturability), we have that $F'''_{\mathsf{GenBK}} = F''_{\mathsf{GenBK}}$ and hence their obfuscations are indistinguishable to a PPT adversary by the security of indistinguishability obfuscation.
Game 6: Note that Game 5 removes all references to T. In this game, we produce the challenge ciphertext given to the adversary as an encryption of a uniformly random message $m' \xleftarrow{\$} \mathcal{M}$. The adversary has a zero advantage in this game.

An efficient distinguisher $\mathcal{D}$ that can distinguish between Game 5 and Game 6 can be used to violate Property 3. Let b be the challenger's random bit. Let m_0 and m_1 be the messages chosen by the adversary. Given a challenge instance of Property 3 of the form $(\mathsf{PP}, T', \mathsf{bk}_{\mathsf{id}^*}, c^*)$ where id^* is the adversary's target identity, and c^* is an encryption of either m_b or a uniformly random element in $\mathcal{M}$. Note that PP, T' and $\mathsf{bk}_{\mathsf{id}^*}$ are distributed identically to both Game 5 and Game 6. Hence, we can construct an algorithm to perfectly simulate $\mathcal{D}$'s view, and give c^* to $\mathcal{D}$ as the challenge ciphertext. If c^* encrypts m_b, Game 5 is perfectly simulated; otherwise if c^* encrypts a random message, Game 6 is perfectly simulated. It follows that a non-negligible advantage distinguishing between Game 5 and Game 6 implies a non-negligible advantage distinguishing the LHS and RHS distributions of Property 3. □

C Alternative Approach for Our Compiler: Using an obfuscated program for bootstrapping

Consider the following program $F_{\mathsf{Bootstrap}}$ that performs the bootstrapping operation:

Program $F_{\mathsf{Bootstrap}}(\mathsf{id}, c)$:

1. Compute $r_1 \parallel r_2 \leftarrow \mathsf{PRF.Eval}(K, \mathsf{id})$.
2. Compute $r_3 \leftarrow \mathsf{PRF.Eval}(K, \mathsf{id} \parallel c)$.
3. Compute $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(T, \mathsf{id}; r_1)$.
4. Compute $\mathsf{bk}_{\mathsf{id}} \leftarrow \mathsf{WGenBootstrapKey}(\mathsf{PP}_{\mathcal{E}}, \mathsf{sk}_{\mathsf{id}}; r_2)$.
5. **Output** $\mathsf{Bootstrap}(\mathsf{PP}_{\mathcal{E}}, \mathsf{bk}_{\mathsf{id}}, c; r_3)$.

We define another scheme $\hat{\mathcal{E}}_2$ that is defined in the same way as $\hat{\mathcal{E}}_1$ with the following changes:

1. An obfuscation $\beta \leftarrow i\mathcal{O}(F_{\mathsf{Bootstrap}})$ is generated in the setup algorithm and included in the public parameters.
2. The bootstrapping algorithm $\hat{\mathcal{E}}_2.\mathsf{Bootstrap}$, on input identity id and ciphertext c, simply becomes equivalent to computing $\beta(\mathsf{id}, c)$.

Once again the proof strategy proceeds in the same manner as the previous approach. When we move from T to T', it becomes necessary to ensure that on input an identity $\mathsf{id} \neq \mathsf{id}^*$ to $F_{\mathsf{Bootstrap}}$, performing bootstrapping with a bootstrapping key based on a *different* underlying secret key (i.e. one generated with T' instead of T) can produce an identical ciphertext to the ciphertext outputted by the original $F_{\mathsf{Bootstrap}}$ above. More precisely, what is needed here is an algorithm $\mathsf{SimBootstrap}$ such that for any $\mathsf{id} \in \mathcal{I} \setminus \{\mathsf{id}^*\}$, $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(T, \mathsf{id}; r_1)$, $\mathsf{bk}_{\mathsf{id}} \leftarrow \mathsf{WGenBootstrapKey}(\mathsf{PP}_{\mathcal{E}}, \mathsf{sk}_{\mathsf{id}}; r_2)$, randomness r_1, r_2, r_3, and ciphertext $c \in \mathcal{C}$, it holds that

$$\mathsf{Bootstrap}(\mathsf{PP}_{\mathcal{E}}, \mathsf{bk}_{\mathsf{id}}, c; r_3) = \mathsf{SimBootstrap}(\mathsf{PP}_{\mathcal{E}}, T', \mathsf{id}, c, r_1, r_2, r_3). \tag{C.1}$$

The barrier to realizing such an algorithm $\mathsf{SimBootstrap}$ for presently-known IBFHEs hinges on the fact that a ciphertext c^* obtained from a homomorphic evaluation of a circuit C is dependent on the plaintext inputs to C, even leaving aside the output of C that is encrypted by c^*. Thus, homomorphically evaluating the decryption circuit, as in bootstrapping, with encryptions of two different secret keys (i.e. two different bootstrapping keys) results in two different resultant ciphertexts. However, it is non-trivial to "wipe" from c^* the unique trace left by a secret key $\mathsf{sk}_{\mathsf{id}}$ without access to $\mathsf{sk}_{\mathsf{id}}$.

Note that if $\mathcal{E}$ is *single-point* trapdoor-puncturable (i.e. it satisfies Equation 4.1), then it is easy to construct an algorithm $\mathsf{SimBootstrap}$ that satisfies C.1. Such a $\mathsf{SimBootstrap}$ would use $\mathsf{SimKeyGen}$ with T' and r_1 to generate $\mathsf{sk}_{\mathsf{id}}$, then $\mathsf{sk}_{\mathsf{id}}$ and r_2 to generate $\mathsf{bk}_{\mathsf{id}}$. Hence, single-point trapdoor-puncturability implies security of both approaches. However due to the fact that $F_{\mathsf{Bootstrap}}$ subsumes F_{GenBK}, the additional complexity of $F_{\mathsf{Bootstrap}}$ is extraneous for a single-point trapdoor-puncturable $\mathcal{E}$, since the process of bootstrapping itself can be more efficiently performed directly by the evaluator. As a result, Approach 1 is preferable for a single-point trapdoor-puncturable scheme.