# *Anonymous IBE, Leakage Resilience and Circular Security from New Assumptions*

# Anonymous IBE, Leakage Resilience and Circular Security from New Assumptions

Zvika Brakerski* Alex Lombardi† Gil Segev‡ Vinod Vaikuntanathan§

**Abstract**

In *anonymous* identity-based encryption (IBE), ciphertexts not only hide their corresponding messages, but also their target identity. We construct an anonymous IBE scheme based on the Computational Diffie-Hellman (CDH) assumption in general groups (and thus, as a special case, based on the hardness of factoring Blum integers).

Our approach extends and refines the recent tree-based approach of Cho et al. (CRYPTO '17) and Döttling and Garg (CRYPTO '17). Whereas the tools underlying their approach do not seem to provide any form of anonymity, we introduce two new building blocks which we utilize for achieving anonymity: *blind garbled circuits* (which we construct based on any one-way function), and *blind batch encryption* (which we construct based on CDH).

We then further demonstrate the applicability of our newly-developed tools by showing that batch encryption implies a public-key encryption scheme that is both resilient to leakage of a $(1 - o(1))$-fraction of its secret key, and KDM secure (or circular secure) with respect to all linear functions of its secret key (which, in turn, is known to imply KDM security for bounded-size circuits). These yield the first high-rate leakage-resilient encryption scheme and the first KDM-secure encryption scheme based on the CDH or Factoring assumptions.

Finally, relying on our techniques we also construct a batch encryption scheme based on the hardness of the Learning Parity with Noise (LPN) problem, albeit with very small noise rate $\Omega(\log^2(n)/n)$. Although this batch encryption scheme is not blind, we show that it still implies standard (i.e., non-anonymous) IBE, leakage resilience and KDM security. IBE and high-rate leakage resilience were not previously known from LPN, even with extremely low noise.

*Weizmann Institute of Science. Supported by the Israel Science Foundation (Grant No. 468/14) and Binational Science Foundation (Grants No. 2016726, 2014276) and ERC Project 756482 REACT.

†Massachusetts Institute of Technology. Email: alexjl@mit.edu. Supported by an Akamai Presidential Fellowship and the grants of the fourth author.

‡School of Computer Science and Engineering, Hebrew University of Jerusalem, Jerusalem 91904, Israel. Email: segev@cs.huji.ac.il. Supported by the European Union's 7th Framework Program (FP7) via a Marie Curie Career Integration Grant (Grant No. 618094), by the European Union's Horizon 2020 Framework Program (H2020) via an ERC Grant (Grant No. 714253), by the Israel Science Foundation (Grant No. 483/13), by the Israeli Centers of Research Excellence (I-CORE) Program (Center No. 4/11), by the US-Israel Binational Science Foundation (Grant No. 2014632), and by a Google Faculty Research Award.

§Massachusetts Institute of Technology. Email: vinodv@mit.edu. Research supported in part by NSF Grants CNS-1350619 and CNS-1414119, Alfred P. Sloan Research Fellowship, Microsoft Faculty Fellowship, the NEC Corporation, a Steven and Renee Finn Career Development Chair from MIT. This work was also sponsored in part by the Defense Advanced Research Projects Agency (DARPA) and the U.S. Army Research Office under contracts W911NF-15-C-0226 and W911NF-15-C-0236.

# Contents

# 1 Introduction

Identity Based Encryption (IBE) is a form of public key encryption where a user's public key is just his name. Specifically, an authority holding a master secret key $\mathsf{msk}$ can generate individual secret keys for users $\mathsf{sk}_{\mathsf{id}}$ according to their identity $\mathsf{id}$, and encryption is performed using a master public key ($\mathsf{mpk}$) and the identity of the recipient. The notion of IBE was proposed by Shamir [Sha84] but first realized only over 15 years later [BF03, Coc01]. Aside from the obvious utility of using IBE for the purpose for which it was intended, it has also proved to be a useful building block to achieve other cryptographic tasks (e.g. chosen-ciphertext secure encryption [BCHK07]) as well as an inspiration for defining more expressive forms of encryption schemes with *access control*. Most generally, the latter refers to schemes where multiple secret keys can be generated, but each key can only recover encrypted information if some predefined condition holds. The most natural generalization is to attribute based encryption (ABE) [SW05, GPSW06] where secret keys $\mathsf{sk}_f$ correspond to policies $f$, and encryptions are with respect to attributes $x$, so that the message is decryptable only if $f(x) = 1$. IBE is a special case where $f$ is a point function (i.e. $f_a(x) = 1$ if and only if $x = a$).

Very recently, a beautiful work of Garg and Döttling [DG17a] proposed a new *tree based* approach for IBE and showed that it implies a candidate IBE scheme from the computational Diffie-Hellman assumption (CDH), which was previously unknown. Their main building blocks were garbled circuits and a special form of encryption called Chameleon Encryption. In a follow-up work [DG17b] they showed that tree based constructions can also be used to amplify the properties of IBE schemes.

An important variant of IBE is one where it is also required that a ciphertext for recipient $\mathsf{id}$ does not expose $\mathsf{id}$ to an unauthorized decryptor. This property is called *anonymity*. Anonymous IBE is quite useful, e.g. for searchable encryption [BCOP04], and analogously to the connection between IBE and ABE, anonymous IBE is a special case of *attribute hiding* ABE (e.g., as in [KSW08]). The latter has raised much interest recently in the cryptographic literature due to its connection to *functional encryption schemes*. Anonymous IBE schemes can be constructed from pairings [BCOP04, ABC+08, BW06, Gen06], lattices [GPV08, ABB10, CHKP12] and quadratic residuosity [BGH07] (the last one in the random oracle model).

The [DG17a, DG17b] constructions are not anonymous for a fundamental reason. Their construction is based on an implicit exponential-size prefix tree representing the entire space of identities. The encryption operation considers a path from the root to the leaf representing the target $\mathsf{id}$ and constructs a sequence of garbled circuits, each respective to a node along this path. At decryption time, the garbled circuits are evaluated from root to leaf, where the output of each garbled circuit is used to generate the input labels for the next garbled circuit along the path. Therefore, if one tries to decrypt a ciphertext intended for $\mathsf{id}$ using a key for $\mathsf{id}'$, the decryption process will succeed up to the node of divergence between $\mathsf{sk}$ and $\mathsf{sk}'$, at which point the $\mathsf{sk}_{\mathsf{id}'}$ decryptor will not be able to decode the labels that correspond to the next garbled circuit. Thus, this process necessarily reveals the common prefix of $\mathsf{id}$ and (a known) $\mathsf{id}'$.

## 1.1 Our Results

In this work, we present new primitives and techniques showing how to get significantly more mileage out of the tree-based approach. First and most importantly, we build on the tree-based approach using new tools that we call *blind batch encryption* and *blind garbled circuits* to construct

anonymous IBE schemes. Secondly, we show that our building blocks can be constructed from assumptions not previously known to imply IBE at all, in particular, the learning parity with noise (LPN) assumption with extremely small noise. Finally, we show that our building blocks can be used to achieve cryptographic capabilities that are apparently unrelated to IBE, namely leakage resilience and KDM security. We elaborate on all of these contributions below.

**Batch Encryption and New Constructions of IBE.** The recent work of Döttling and Garg [DG17b] show an amplification between notions of identity based encryption. Namely, they show how to go from any selective IBE scheme to a fully secure IBE scheme. We notice that their construction can be repurposed to do something very different. Namely, we show how to start from an IBE scheme which only supports *polynomially many identities* but with short master public key, and construct a full-fledged IBE scheme. In particular, the scheme should support $T = T(\lambda)$ identities with a master public key of size $S = S(\lambda) = T^{1-\epsilon} \cdot \mathsf{poly}(\lambda)$ for some constant $\epsilon > 0$ and a fixed polynomial $\mathsf{poly}$; we call this a weakly compact IBE scheme. We remind the reader that non-compact IBE schemes, namely ones that support $T$ identities and have a master public key that grows linearly with $T$, in fact follow quite easily from any public-key encryption scheme (see, e.g., [DKXY02]).

Weakly compact IBE turns out to be easier to construct using the techniques of [DG17a], and in particular it does not require the full power of their Chameleon Encryption. We show that it is sufficient to start from a building block that we call *batch encryption*. In particular, whereas Chameleon Encryption is required to have a trapdoor, a batch encryption scheme has no trapdoors. Indeed, looking ahead, we remark that this feature of requiring no trapdoors is what enables our IBE construction from the extremely-low-noise LPN assumption. The batch encryption definition takes after the laconic oblivious transfer primitive presented by Cho, Döttling, Garg, Gupta, Miao and Polychroniadou [CDG+17] (a definition that preceded Chameleon Encryption).

A batch encryption scheme is a public key encryption scheme in which key generation is a projection (i.e. the key generation algorithm takes the secret key as input and outputs a shorter string as the public key). For secret keys of length $n$, a batch encryption scheme encrypts an array of $n \times 2$ messages at a time. At decryption, only one out of each pair of messages is recovered, depending on the value of the respective secret key bit. We require that we can instantiate the scheme for any $n$ without increasing the length of the public key. Indeed, batch encryption is very similar to laconic oblivious transfer [CDG+17] and the two are essentially existentially equivalent. The formal definition varies slightly in that laconic OT can more efficiently handle situations where only a subset of the $n$ message pairs are encrypted. Another formal difference is that the laconic OT formulation allows for a randomized receiver message, however since receiver privacy is not a requirement for this primitive this is not actually needed and therefore the analogous component in batch encryption is deterministic. The formulation of batch encryption is more useful for our applications, but our constructions can be seen as simply constructing laconic OT.

We show that batch encryption implies weakly compact IBE (as defined above) and that weakly compact IBE can be bootstrapped to a full-fledged IBE scheme.

**Batch Encryption from CDH and Extremely-Low-Noise LPN.** Batch encryption can be constructed from CDH, using the methods of [DG17a]; it can also be constructed from the Learning with Errors (LWE) assumption in a straightforward manner without using lattice trapdoors. Thus we observe that LWE-based IBE does not require lattice trapdoors, even though they are used

by all previous constructions. We note that the resulting IBE scheme is greatly inefficient, quite probably much less efficient than a trapdoor based construction, however the conceptual difference here could be of interest.

We take an additional step forward and show that even the learning parity with noise (LPN) assumption is sufficient to instantiate batch encryption, although we must rely on LPN with very extreme parameters. The LPN assumption with a constant noise rate implies one-way functions; with a noise rate of $1/\sqrt{n}$ (where $n$ is the dimension of the LPN secret), it implies public-key encryption [Ale11]; and with the extremely low noise rate of $\log^2 n/n$, it implies collision-resistant hash functions [BLVW17]. The latter parameter setting is insecure against quasi-polynomial adversaries, but given the state of the art in algorithms for LPN, presumably secure against polynomial-time adversaries. Indeed, it is ill advised to base cryptographic hardness on the gap between polynomial time adversaries and quasi-polynomial time hardness and we see this result mainly as proof of concept showing that batch encryption can be based on structures that were not considered to imply IBE so far.

**The Blinding Technique and Anonymous IBE.** Our main contribution is a construction of anonymous IBE from the CDH assumption.

To construct anonymous IBE we present techniques that allow us to walk down the identity-space tree at decryption time *blindly*. Namely, in a way that does not reveal to the decryptor whether they are on the correct path until the very end of the process. This allows us to overcome the aforementioned basic obstacle. We present a variety of blind primitives that help us in achieving this goal.

The first building block we introduce is *blind garbled circuits*. Recall that a standard circuit garbling scheme takes a circuit $C$ as input, and outputs a garbled version of the circuit $\widehat{C}$ together with pairs of labels $\mathsf{lab}_{i,b}$ for the input wires. Given $\widehat{C}, \mathsf{lab}_{i,x_i}$, the value $C(x)$ can be computed. For security, there is a simulator that takes $y = C(x)$ and produces a garbled circuit and a set of input labels that are indistinguishable from the original. We augment this definition with a *blindness* property, requiring that the simulated garbled circuit and labels are *completely uniform* when starting with a completely uniform $y$ that is unknown to the distinguisher (indeed, the latter condition is necessary since an attempt to evaluate the simulated garbled circuit should output $y$). We show that blind garbled circuits can be constructed by properly instantiating the "point-and-permute" construction [BMR90, Rog91], based on any one way function. Interestingly, as far as we know, the point-and-permute construction has been used to achieve more efficient garbled circuits, but has never been used to acheive stronger security properties.

We then introduce *blind batch encryption*, which is the blind version of the aforementioned batch encryption primitive. The use of batch encryption in IBE constructions is as a way to encrypt labels for a garbled circuit so that only one label per input wire can be decrypted (i.e. the one corresponding to the batch encryption secret key). Blind batch encryption is a "blindness preserving" counterpart for blind garbled circuits as follows. We require that if a random message is encrypted using a blind batch encryption scheme, then the resulting ciphertext is completely random as well.[1] This combines very naturally with a blind garbling scheme: if we batch encrypt labels to a blind garbled circuit with a random output, then by simulation security this is indistinguishable from encrypting random labels that are independent of the garbled circuit. Therefore, we are guaranteed that the batch ciphertext itself is random as well. At a very high level, this will allow

[1] We actually allow a slight relaxation of this condition.

us to propagate the randomness (blindness) property along the leaf-root path in the tree, and avoid revealing any information via partial decryption.

We show that blind batch encryption can be constructed based on CDH by introducing a modification to the CDH based Chameleon Encryption construction from [DG17a]. Unfortunately, our construction based on extremely low noise LPN is not blind.

We apply these building blocks to anonymize the aforementioned IBE construction from batch encryption. We present a blindness property for IBE that is analogous to the one for batch encryption, requiring that an encryption of random message is indistinguishable from random *even to a user who is permitted to decrypt it.* We show that this notion implies anonymous IBE, and furthermore, the construction of full-fledged IBE from a weakly compact scheme, and a construction of the weakly compact scheme from a batch encrytion scheme both preserve blindness (if we use blind garbled circuits). In fact, formally, to avoid redundancy we only present the reduction in the blind setting, and the non-blind variant follows as a special case.

We find it intriguing that even though we only require anonymous IBE at the end, we have to go through the (apparently stronger) primitive of blind IBE. Roughly speaking, the difference is that anonymous IBE only requires hiding of the identities in settings where the adversary cannot decrypt (namely, he only obtains secret keys for identities $\mathsf{id}$ different from either of the challenge identities $\mathsf{id}_0$ and $\mathsf{id}_1$) while blind IBE requires hiding of the identities even in settings where the adversary can decrypt. Morally, we think of this as the difference between weak attribute-hiding and strong attribute-hiding in predicate encryption (although the details are somewhat different). We also note that weakly compact anonymous IBE can be constructed generically from any weakly compact IBE scheme. Thus, had we been able to bootstrap from a weakly compact anonymous IBE scheme into a full-fledged anonymous IBE, we would have a generic construction of anonymous IBE scheme from any IBE scheme.

**Batch Encryption Implies Leakage Resilience and KDM Security.** We show that the utility of batch encryption schemes go beyond IBE, thus expanding [CDG+17] who showed a variety of applications of laconic OT, mostly in the context of multi-party computation. We show that batch encryption naturally gives rise to a public key encryption scheme with desirable properties such as resilience to high rate $(1-o(1))$ key leakage [AGV09, NS12] and security for key dependent messages [BRS02] (KDM, also known as circular security). This allows us to present constructions from assumptions such as CDH, Factoring and extremely-low-noise LPN that were not known before [AGV09, NS12, BHHO08, ACPS09, BG10, HLWW16]. Note that from [CDG+17] it was not even clear that the (nearly) equivalent notion of laconic OT even implies plain public key encryption (without assuming "receiver privacy"; with receiver privacy, we know that any 2 message OT implies PKE). This further strengthens our impression that batch encryption is a notion worthy of further exploration.

The basic idea is quite straightforward. Recall that a batch encryption scheme encrypts an array of $n \times 2$ bits, and decryption only recovers one out of two pairs. Therefore, if the secret key is $\mathbf{x} \in \{0,1\}^n$ and the encrypted message is $\mathbf{M} \in \{0,1\}^{n\times 2}$, then the decrypted message is equal to $m = \sum_i (M_{i,0}(1 \oplus x_i) \oplus M_{i,1}x_i) = \sum_i M_{i,0} \oplus \sum_i (M_{i,1} \oplus M_{i,0})x_i$. Denote $\alpha_0 = \sum_i M_{i,0}$, $\alpha_i = M_{i,1} \oplus M_{i,0}$. Note that it is sufficient that one out of each pair $M_{i,0}, M_{i,1}$ is random to make all $\{\alpha_i\}_{i>0}$ completely random, this property will be useful for us. To encrypt, we will $n$-out-of-$n$ secret share our message $m = \sum_i \mu_i$ and set $M_{i,0} = M_{i,1} = \mu_i$. Decryption follows by decrypting the batch ciphertext and reconstructing $m$. For security, we notice that the batch security means that we can

convert one out of each pair $M_{i,0}, M_{i,1}$ to random (this will be unnoticed even to a distinguisher who has the key $x$). At this point, we recall that $x$ is in fact information theoretically unknown to the adversary who only sees the projected public key (recall that the projection key generation function is shrinking). Thus the value $\sum_i \alpha_i x_i$ extracts from the remaining entropy in $x$ and is statistically close to uniform (indeed one has to prove that there is no additional usable information in the ciphertext other than the output message $m$). This argument naturally extends to leakage resilience, since we can allow additional leakage on $x$ so long as sufficient information remains to allow for extraction. It appears that security against computationally (sub-exponentially) hard to invert unbounded length leakage ("auxiliary input resilience" [DGK+10]) should follow in a similar manner, however we do not provide a proof.

For KDM security, we notice that for any linear function of $x$ of the form $\alpha_0 \oplus \sum_i \alpha_i x_i$ the above shows how to simulate a ciphertext that decrypts to this message (in fact, how to sample a random such ciphertext). Indeed this ciphertext is not honestly generated but we can show that it is indistinguishable from one. This is the basis for KDM security. We recall that as shown in [BHHI10, App11], KDM security with respect to linear functions can be amplified to KDM security for bounded polynomial functions of the key. Interestingly, this amplification approach also involves batch encrypting labels for a garbled circuit.

### 1.2 Our Techniques

The rest of the paper is organized as follows. In Section 3, we define the notion of (blind) batch encryption and construct it from the CDH assumption. We also provide a construction of the (non-blind) batch encryption from the extremely low noise LPN assumption. We then introduce the notion of blind garbled circuits and construct it in Section 4. Then, in Section 5, we show how to use (blind) batch encryption to construct a weakly compact (blind) IBE scheme. In Section 6, we bootstrap the weakly compact (blind) IBE scheme into a full-fledged (blind) IBE scheme. Finally, in Section 7, we construct from (blind) batch encryption a (blind) public key encryption scheme satisfying (high leakage rate) leakage resilience and KDM security with respect to affine functions of the secret key.

We first provide an overview of the last step of our anonymous IBE construction, namely our bootstrapping theorem for blind IBE, and then the construction of weakly compact IBE from batch encryption.

**Bootstrapping Blind IBE.** We start with bootstrapping a regular IBE scheme, and then describe the additional techniques required to handle blindness.

Suppose we have a blind IBE scheme $\mathcal{WIBE}$ that supports $T = T(\lambda)$ identities and has a master public key whose size is $S = S(\lambda) = T^{1-\epsilon} \cdot p(\lambda)$ for some absolute constant $\epsilon > 0$ and a fixed polynomial $p$. To keep our exposition simple, assume that the ciphertexts in this scheme are truly pseudorandom. We remark that without the restriction on the master public key length, there are generic ways of constructing such schemes from any public-key encryption scheme, resulting in master public key of length $O(T \cdot \lambda)$; see, e.g., [DKXY02]. The key leverage we have in $\mathcal{WIBE}$ is that the master public key grows *sublinearly* with the number of identities the scheme supports.

We will show how to construct another (blind) IBE scheme $\mathcal{WIBE}'$ that supports $2T$ identities without growing the master public key at all. This will not be enough by itself to prove the full bootstrapping theorem by induction because the ciphertext and secret key sizes grow significantly

in the transformation. Nevertheless, all of the necessary ideas for the full bootstrapping theorem are in this toy example already.

We start by picking $T$ to be sufficiently large so that the size of the master public key $T^{1-\epsilon} \cdot p(\lambda)$ is at most $T/4$. The master public key of $\mathcal{WIBE}'$ is a single master public key of $\mathcal{WIBE}$; we will denote it by $\mathsf{mpk}^{(\epsilon)}$ and associate it with the root of a depth-2 tree with branching factor 2 in the first level and $T$ in the second. We will also pick two other master public keys $\mathsf{mpk}^{(0)}$ and $\mathsf{mpk}^{(1)}$, but *will not publish it* as part of the $\mathcal{WIBE}'$ master public key. The master secret key in $\mathcal{WIBE}'$ will, however, include $\mathsf{msk}^{(\epsilon)}$ as well as $\mathsf{mpk}^{(i)}, \mathsf{msk}^{(i)}$.

The two questions we address next is (a) how to encrypt a message $m$ for an identity $\mathsf{id}||\mathsf{id}'$ where $\mathsf{id} \in \{0,1\}$ and $\mathsf{id}' \in \{0,\ldots,T-1\}$ and (b) how to generate identity secret keys.

Let us address the question of secret keys first. The secret key for an identity $\mathsf{id}||\mathsf{id}'$ where $\mathsf{id} \in \{0,1\}$ and $\mathsf{id}' \in \{0,\ldots,T-1\}$ will include as part of it $\mathsf{sk}_{\mathsf{id}'}^{(\mathsf{id})}$, namely the secret key for the identity $\mathsf{id}'$ generated with respect to the master public key $\mathsf{mpk}^{(\mathsf{id})}$. Thus, it makes sense to encrypt a message $m$ under the identity $\mathsf{id}||\mathsf{id}'$ by encrypting it with respect to the identity $\mathsf{id}'$ under the master public key $\mathsf{mpk}^{(\mathsf{id})}$. If the encryptor could do this, decryption indeed works and we are done! However, the big problem here is that the encryptor does not know $\mathsf{mpk}^{(0)}$ or $\mathsf{mpk}^{(1)}$. How can the encryptor generate a ciphertext without knowing the master public key?

It is here that we use the technique of *deferred encryption* similarly to [GKW16] and the aforementioned [DG17a]. That is, instead of having to generate an encryption of $m$ under an unknown master public key, the encryptor simply constructs a circuit $C[m, \mathsf{id}']$ which has the message $m$ and the identity $\mathsf{id}'$ hardcoded. The circuit $C[m, \mathsf{id}']$, on input an $\mathsf{mpk}$, produces an encryption of $m$ under $\mathsf{mpk}$ with identity $\mathsf{id}'$. (The circuit also has the encryption randomness $r$ hardcoded).

The encryptor now does two things. It first garbles this circuit to produce $\widehat{C}$, the garbled circuit, together with $2S$ labels $\mathsf{lab}_{i,b}$ for $i \in [S]$ and $b \in \{0,1\}$. It then encrypts each label $\mathsf{lab}_{i,b}$ using the identity $(\mathsf{id}, i, b)$ under the master public key $\mathsf{mpk}^{(\epsilon)}$. It is here that we use compactness of $\mathcal{WIBE}$ in a crucial way: since $\mathcal{WIBE}$ can support $T > 4S$ identities, it can indeed be used to encrypt these labels.

The identity secret key for $\mathsf{id}||\mathsf{id}'$ now contains two things. As before, it contains the secret key for the identity $\mathsf{id}'$ under the master public key $\mathsf{mpk}^{(\mathsf{id})}$. It also contains the secret keys for the $S$ identities $(\mathsf{id}, i, \mathsf{mpk}^{(\mathsf{id})}[i])$ under the master public key $\mathsf{mpk}^{(\epsilon)}$.

Decryption proceeds by first using the secret keys for the $S$ identities to unlock half the labels for the garbled circuit $\widehat{C}$, namely, the labels corresponding to the input $\mathsf{mpk}^{(\mathsf{id})}$. It then decodes the garbled circuit to produce an encryption of $m$ with identity $\mathsf{id}'$ under the master public key $\mathsf{mpk}^{(\mathsf{id})}$. The first part of the secret key is now precisely what is necessary to decrypt and obtain the message $m$.

We first argue semantic security (IND-ID-CPA security), then show the barriers to achieving blindness/anonymity and how our new techniques overcome them. Let the challenge identity be $\mathsf{id}||\mathsf{id}'$. A ciphertext of a message $m$ under $\mathsf{id}||\mathsf{id}'$ contains the garbled circuit $\widehat{C}$ and encryptions of the labels $L_{i,b}$ under identities $(\mathsf{id}, i, b)$ with respect to the master public key $\mathsf{mpk}^{(\epsilon)}$. Notice first that secret keys for identities that begin with the bit $(1 - \mathsf{id})$ are completely useless in unlocking any of the labels of the garbled circuit. Only secret keys for identities that begin with the bit $\mathsf{id}$ are useful. Even they can only ever unlock half the labels of the garbled circuit. Indeed, this is crucial since otherwise we will not be able to invoke the security of the garbled circuit at all!

The secret keys for identities that begin with the (matching) bit $\mathsf{id}$ unlock the garbled labels

corresponding to the input $\mathsf{mpk}^{(\mathsf{id})}$. One now invokes the security of the garbled circuit which says that the only thing revealed by these labels together with the garbled circuit is the encryption of $m$ under the identity $\mathsf{id}'$ generated with the master public key $\mathsf{mpk}^{(\mathsf{id})}$. Now, since the adversary never obtains the secret key for the challenge identity, she never gets the secret key for $\mathsf{id}'$ under $\mathsf{mpk}^{(\mathsf{id})}$. Thus, the semantic security of $\mathcal{WIBE}$ tells us that the message $m$ remains hidden.

As described in the introduction, this construction does not lead to an anonymous IBE scheme. Indeed, given a ciphertext with respect to the identity $\mathsf{id}_1||\mathsf{id}'_1$ and a secret key for $\mathsf{id}_2||\mathsf{id}'_2 \neq \mathsf{id}_1||\mathsf{id}'_1$, one can easily tell if $\mathsf{id}_1 = \mathsf{id}_2$ or not, simply by seeing if the first decryption step succeeds. Worse, it is unclear if the anonymity of the underlying $\mathcal{WIBE}$ scheme helps here at all. If $\mathsf{id}_1 = \mathsf{id}_2$, the secret keys are authorized to decrypt half the encrypted labels ("first level ciphertexts"), and if $\mathsf{id}_1 \neq \mathsf{id}_2$, the secret keys do not decrypt any of them. Thus, it seems at first glance that we are doomed: one can seemingly always recover the first bit of the identity in any tree-based scheme.

Our *key observation* is that even in the "partly-authorized case", the ciphertexts are encryptions of fresh random labels. (In reality, these labels do appear again in the garbled circuits; in the proof, this is handled by doing the hybrids in the reverse order from the current presentation where pseudorandomness at the leaves comes from the adversary not having the final secret key corresponding to the target identity.) Thus, if the $\mathcal{WIBE}$ scheme is *blind*, the adversary can still not tell the difference between whether she had an authorized key or not. In both cases, the output of the decryption is a bunch of uniformly random strings! Our troubles, unfortunately, do not stop there. The next line of defense, the garbled circuit, could also help the adversary distinguish whether she obtained the right labels in the first step or just random strings. Blindness again comes to the rescue: this time, we use our blind garbled circuits in conjunction with the fact that the output of the circuit we are garbling is actually pseudorandom.

This concludes a sketch of our toy construction and its security proof.

Of course, there was no reason a-priori to have only one level of garbled circuits. One can garble the "inner $\mathcal{WIBE}$" encryptions and do so for every level in the tree. The inputs to each such garbled circuit is a single master public key, so the input labels to this new garbled circuit will be no larger than the previous level's input labels. We can thus build an IBE scheme corresponding to a tree of any $\mathrm{poly}(\lambda)$ depth, allowing us to support exponentially many identities: a full IBE scheme. Of course, we cannot generate exponentially many $\mathcal{WIBE}$ master public keys (one for each node of the tree), but we can implicitly generate them using a PRF.

For full details on our bootstrapping theorem, see Section 6.

**From Batch Encryption to Weakly Compact IBE.** We now provide a high level overview of how to construct weakly compact IBE from batch encryption. Formally, we construct a scheme that supports any polynomial number $T$ of identities with public key size $\lambda$. We focus on the vanilla (non-blind) variant as the blind one follows via a similar construction. We note that batch encryption schemes go hand-in-hand with garbled circuits (a connection that is extensively used in [CDG+17, DG17a]). Consider a batch encryption scheme with secret key $x$ of length $n \gg \lambda$ and public key length $\lambda$. Then we can encrypt an array of $n \times 2$ elements, specifically we can encrypt labels for an $n$-input garbled circuit. The holder of the secret key will be able to evaluate said garbled circuit on the labels that correspond to his secret key. In other words, batch encryption allows us to specify a circuit $C : \{0,1\}^n \to \{0,1\}^m$ and generate a ciphertext that will reveal only $C(x)$, even to an adversary that holds the secret key.

Recall that the only requirement we want from the resulting IBE is short master public key. All

other parameters can depend polynomially on the size of the identity space. We will therefore generate a sequence of $T$ key pairs for a standard public key encryption scheme $(\mathsf{pke.pk}_1, \mathsf{pke.sk}_1), \ldots, (\mathsf{pke.pk}_T, \mathsf{pke.sk}_T)$. For simplicity assume $|\mathsf{pke.pk}_i| = \lambda$. Then we instantiate the batch encryption scheme with $n = T \cdot \lambda$ and generate a batch public key, a projection of $x = \mathsf{pke.pk}_1 \| \cdots \| \mathsf{pke.pk}_T$. The batch public key will serve as $\mathsf{mpk}$ of the weakly compact IBE scheme, and indeed its length is $\lambda$, independent of $T$.

To encrypt a ciphertext to target identity $\mathsf{id} \in [T]$, we generate a garbled circuit that expects as input a sequence of $T$ public keys, and takes the $\mathsf{id}$-th of them and uses it to encrypt the message. The IBE secret key for identity $\mathsf{id}$ will contain the entire sequence $x = \mathsf{pke.pk}_1 \| \cdots \| \mathsf{pke.pk}_T$, indeed in this case the batch encryption secret key is not secret at all! In addition, the IBE secret key for $\mathsf{id}$ will contain $\mathsf{pke.sk}_{\mathsf{id}}$. Given a ciphertext, a decryptor will first use $x$ to evaluate the garbled circuit and recover $C(x)$, which in this case is just a public-key encryption ciphertext with respect to $\mathsf{pke.pk}_{\mathsf{id}}$. The next step is to just use $\mathsf{pke.sk}_{\mathsf{id}}$ to decrypt this ciphertext and recover the message.

Security follows from the security of batch encryption (which conveniently applies also when the batch secret key $x$ is known) and the security of the public key encryption scheme.

# 2 Preliminaries and Definitions

## 2.1 (Anonymous) Identity-Based Encryption

**Definition 2.1** (Identity Based Encryption)**.** *An identity based encryption (IBE) scheme consists of five PPT algorithms* $(\mathsf{Params}, \mathsf{Setup}, \mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec})$ *with the following syntax.*

1. $\mathsf{Params}(1^\lambda, 1^t)$ *takes as input the security parameter* $1^\lambda$ *and an identity length* $1^t$*. It returns public parameters* $\mathsf{pp}$ *(which can be reused to generate multiple master public key/master secret key pairs).*
2. $\mathsf{Setup}(\mathsf{pp})$ *takes as input public parameters* $\mathsf{pp}$ *and returns a master public key* $\mathsf{mpk}$ *and master secret key* $\mathsf{msk}$*.*
3. $\mathsf{Keygen}(\mathsf{pp}, \mathsf{msk}, \mathsf{id})$ *takes as input public parameters* $\mathsf{pp}$ *and the master secret key* $\mathsf{msk}$*. It outputs a secret key* $\mathsf{sk}_{\mathsf{id}}$ *associated to* $\mathsf{id}$
4. $\mathsf{Enc}(\mathsf{pp}, \mathsf{mpk}, \mathsf{id}, m)$ *encrypts a message* $m$ *to a specified identity* $\mathsf{id}$*. It outputs a ciphertext* $\mathsf{ct}$*.*
5. $\mathsf{Dec}(\mathsf{pp}, \mathsf{sk}, \mathsf{ct})$ *decrypts a ciphertext* $\mathsf{ct}$ *with secret key* $\mathsf{sk}$*, outputting a plaintext message* $m'$*.*

*We require that an IBE scheme satisfy the following two properties.*

- *Correctness: with probability* 1 *over the randomness of* $(\mathsf{Params}, \mathsf{Setup}, \mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec})$*, we have that* $\mathsf{Dec}(\mathsf{pp}, \mathsf{sk}_{\mathsf{id}}, \mathsf{Enc}(\mathsf{pp}, \mathsf{mpk}, \mathsf{id}, m)) = m$ *where* $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(\mathsf{pp})$ *and* $\mathsf{sk}_{id} \leftarrow \mathsf{Keygen}(\mathsf{msk}, \mathsf{id})$*.*
- *IND-ID-CPA Security: a PPT adversary* $\mathcal{A}$ *cannot win the following security game with probability greater than* $\frac{1}{2} + \mathrm{negl}(\lambda)$*:*
  1. $\mathsf{pp} \leftarrow \mathsf{Params}(1^\lambda, 1^t)$
  2. $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(\mathsf{pp})$
  3. $(\mathsf{id}^*, m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}^{\mathsf{Keygen}(\mathsf{pp}, \mathsf{msk}, \cdot)}(\mathsf{mpk})$

4. $b \xleftarrow{\$} \{0,1\}$
5. $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{mpk}, \mathsf{id}^*, m_b)$
6. $b' \leftarrow \mathcal{A}^{\mathsf{Keygen}(\mathsf{pp},\mathsf{msk},\cdot)}(\mathsf{st}, \mathsf{ct})$
7. *$\mathcal{A}$ wins if and only if $b' = b$ and $\mathsf{id}^*$ was never queried by $\mathcal{A}$ to its* Keygen *oracle.*

**Definition 2.2** (Anonymous IBE)**.** *An anonymous IBE scheme also has the syntax (*Params, Setup, Keygen, Enc, Dec*) of an IBE scheme. It satisfies the same correctness property as IBE, and has the following stronger notion of security:*

- *IND-ANON-ID-CPA Security: A PPT adversary $\mathcal{A}$ cannot with the following security game with probability greater than $\frac{1}{2} + \mathrm{negl}(\lambda)$:*

1. $\mathsf{pp} \leftarrow \mathsf{Params}(1^\lambda, 1^t)$
2. $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(\mathsf{pp})$
3. $(\mathsf{id}_0, \mathsf{id}_1, m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}^{\mathsf{Keygen}(\mathsf{pp},\mathsf{msk},\cdot)}(\mathsf{mpk})$
4. $b \xleftarrow{\$} \{0,1\}$
5. $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{mpk}, \mathsf{id}_b, m_b)$
6. $b' \leftarrow \mathcal{A}^{\mathsf{Keygen}(\mathsf{pp},\mathsf{msk},\cdot)}(\mathsf{st}, \mathsf{ct})$
7. *$\mathcal{A}$ wins if and only if $b' = b$ and $\mathsf{id}_0, \mathsf{id}_1$ were never queried by $\mathcal{A}$ to its* Keygen *oracle.*

## 2.2 Computational Diffie-Hellman (CDH)

Let $g$ be an element of some group $\mathbb{G}$. We say that $q$ is a $\epsilon$-randomizer for $g$ if the statistical distance between $g^a$ for $a \leftarrow \mathbb{Z}_q$ and $h \leftarrow \langle g \rangle$ is at most $\epsilon$. We note that any $q \geq \mathsf{ord}(g) \cdot \lceil 1/\epsilon \rceil$ is an $\epsilon$-randomizer, so it is sufficient to have an upper bound on the order of $g$ in order to compute a randomizer for any $\epsilon$.

A (possibly randomized) group sampler is a ppt algorithm $\mathcal{G}$ that on input the security parameter outputs a tuple $(\mathbb{G}, g, q) \leftarrow \mathcal{G}(1^\lambda)$ which defines a $\mathbb{G}$ by providing a $\mathrm{poly}(\lambda)$-bit representation for group elements, and a polynomial time algorithm for computing the group operation and inversion (and thus also exponentiation), together with an element $g \in \mathbb{G}$ and a $\mathrm{negl}(\lambda)$-randomizer $q$ for $\langle g \rangle$.

The Computational Diffie-Hellman (CDH) assumption w.r.t $\mathcal{G}$, denoted $\mathsf{CDH}_\mathcal{G}$, is that for every ppt algorithm $\mathcal{A}$ it holds that

$$\mathrm{Adv}_{\mathsf{CDH}_\mathcal{G}}[\mathcal{A}](\lambda) = \Pr_{\substack{(\mathbb{G},g,q)\leftarrow\mathcal{G}(1^\lambda) \\ a_1,a_2\leftarrow\mathbb{Z}_q}}[\mathcal{A}(1^\lambda, (\mathbb{G}, g, q), g^{a_1}, g^{a_2}) = g^{a_1 a_2}] = \mathrm{negl}(\lambda) \ .$$

We sometimes omit the indication of $\mathcal{G}$ when it is clear from the context.

We note that there exists a randomized group sampler such that the hardness of factoring Blum integers reduces to the hardness of the CDH problem [Shm85, McC88, BBR99].

## 2.3 Learning Parity with Noise (LPN)

For all $n \in \mathbb{N}$, row vector $\mathbf{s} \in \{0,1\}^n$ and real value $\epsilon \in [0, 1/2]$, define a randomized oracle $A_{\mathbf{s},\epsilon}$ to be s.t. for every call to $A_{\mathbf{s},\epsilon}$, the oracle samples $\mathbf{a} \leftarrow \{0,1\}^n$, $e \leftarrow \mathsf{Ber}_\epsilon$ (where $\mathsf{Ber}$ is the Bernoulli distribution), and outputs $(\mathbf{a}, \mathbf{s} \cdot \mathbf{a} + e)$ where arithmetics are over the binary field. Note that $A_{\mathbf{s},1/2}$ outputs completely uniform entries for every call.

The Learning Parity with Noise assumption $\mathsf{LPN}_{n,\epsilon}$, for a polynomial function $n : \mathbb{N} \to \mathbb{N}$ and a function $\epsilon : \mathbb{N} \to [0, 1/2]$ is that for every $\mathsf{ppt}$ oracle algorithm $\mathcal{A}$ it holds that

$$\mathrm{Adv}_{\mathsf{LPN}_{n,\epsilon}}[\mathcal{A}](\lambda) = \left| \Pr_{\mathbf{s} \leftarrow \{0,1\}^n} [\mathcal{A}^{A_{\mathbf{s},\epsilon}}(1^\lambda)] - \Pr[\mathcal{A}^{A_{\mathbf{0},1/2}}(1^\lambda)] \right| = \mathrm{negl}(\lambda) \ ,$$

where $n = n(\lambda)$, $\epsilon = \epsilon(\lambda)$.

We note that if $\epsilon = \log n/n$ then $\mathsf{LPN}$ is solvable in polynomial time, but no polynomial time algorithm is known for $\epsilon = \Omega(\log^2 n/n)$.

**The Collision Resistant Hash Family of [BLVW17].** It is shown in [BLVW17] how to create Collision Resistant Hash functions based on the hardness of $\mathsf{LPN}_{n,\epsilon}$ for any polynomial $n$, $\epsilon = \Omega(\log^2 n/n)$. Since this construction is the basis for our LPN-based batch encryption construction, let us elaborate a little on it here.

The key to the hash function is a random matrix $\mathbf{A} \in \{0,1\}^{n \times (2n^2/\log n)}$. To apply the hash function on an input $x \in \{0,1\}^{2n}$, they first preprocess it as follows. Interpret $x$ as a collection of $2n/\log n$ blocks, each containing $\log n$ bits. Then interpret each block as a number in $\{1, \ldots, n\}$ using the usual mapping, so $x \in [n]^{2n/\log n}$. Then define a vector $\hat{\mathbf{x}} \in \{0,1\}^{2n^2/\log n}$ as a concatenation of $2n/\log n$ blocks of $n$-bits, such that each block is a $\{0,1\}^n$ indicator vector of the respective entry in $x$ (i.e. have a single bit equal 1 in the location corresponding to the value of the entry in $x$). Finally output $\mathbf{A}\hat{\mathbf{x}}$. This is shrinking from $2n$ to $n$ bits, and CRH follows since a collision implies a low norm vector $\mathbf{v}$ s.t. $\mathbf{Av} = 0$. The argument of security for our batch encryption scheme is similar to their proof of security of CRH, however we do not use it as black box.

## 2.4 One-Time Encryption Using Goldreich-Levin Hard-Core Bit

We show the following one time encryption scheme based on the Goldreich-Levin hard-core bit [GL89].

**Definition 2.3.** *Define* $\mathsf{gl\text{-}enc}(x, \mu)$ *as a randomized function that on input* $x \in \{0,1\}^\ell$, $\mu \in \{0,1\}$ *samples* $\alpha \in \{0,1\}^\ell$ *and outputs* $(\alpha, \langle \alpha, x \rangle \oplus \mu)$, *where the inner product is over the binary field. Define* $\mathsf{gl\text{-}dec}(x, (\alpha, \sigma))$ *be the function that takes* $x \in \{0,1\}^\ell$ *and* $(\alpha, \sigma) \in \{0,1\}^{\ell+1}$ *and outputs* $\sigma \oplus \langle \alpha, x \rangle$.

By definition, for all $x, \mu$ it holds that $\mathsf{gl\text{-}dec}(x, \mathsf{gl\text{-}enc}(x, \mu)) = \mu$ with probability 1. Furthermore, the Goldreich-Levin Theorem asserts that given an ensemble of joint distributions $\{(X_\lambda, Z_\lambda)\}_\lambda$ s.t. for any polynomial time $\Pr_{(x,z) \leftarrow (X,Z), \mathcal{A}}[\mathcal{A}(1^\lambda, z) = x] = \mathrm{negl}(\lambda)$, then $(z, \mathsf{gl\text{-}enc}(x, \mu))$ is computationally indistinguishable from $(z, U_{\ell+1})$ for any $\mu$ (possibly dependent on $z$). We furthermore note that if $\mu$ is random and unknown to the distinguisher then $\mathsf{gl\text{-}enc}(x, \mu)$ is uniformly random regardless of $x$.

# 3 Blind Batch Encryption and Instantiations

## 3.1 Defining Batch Encryption

A Batch Encryption scheme is an encryption scheme whose key generation is a *projection* function (or a hash function) taking as input a string $x$ to be used as secret key, and outputting a hash value $h$ to be used as public key. The batch encryption scheme is parameterized by a *block size* $B$. The aforementioned string $x$ should be parsed as $x \in [B]^n$. Batch encryption uses the public key $h$ to encrypt an $n \times B$ matrix $\mathbf{M}$ such that a decryptor with secret key $x$ can obtain exactly $M_{i,x_i}$ for all $i \in [n]$; that is, exactly one matrix element from each row of $\mathbf{M}$. Note that when $B = 2$ we can think of $x$ as a bit vector $x \in \{0,1\}^n$ with the natural translation between $\{0,1\}$ and $\{1,2\}$.

In more detail, the syntax of the batch encryption scheme is as follows, where we think of the function $B = B(\lambda, n)$ as a global parameter of the construction.

1. $\mathsf{Setup}(1^\lambda, 1^n)$. Takes as input the security parameter $\lambda$ and key length $n$, and outputs a common reference string $\mathsf{crs}$.

2. $\mathsf{Gen}(\mathsf{crs}, x)$. Using the common reference string, project the secret key $x \in [B]^n$ to a public key $h$.

3. $\mathsf{Enc}(\mathsf{crs}, h, \mathbf{M})$. Takes as input a common reference string $\mathsf{crs}$, the public key $h$, and a matrix $\mathbf{M} \in \{0,1\}^{n \times B}$ and outputs a ciphertext $\mathsf{ct}$. For the purpose of defining the blinding property below, the ciphertext $\mathsf{ct}$ can be written as a concatenation of two parts $\mathsf{ct} = (\mathsf{subct}_1, \mathsf{subct}_2)$.

4. $\mathsf{Dec}(\mathsf{crs}, x, \mathsf{ct})$. Given a ciphertext $\mathsf{ct}$, output a message vector $\mathbf{m}$.

Additionally, a batch encryption scheme supports two *optional* functions.

5. $\mathsf{SingleEnc}(\mathsf{crs}, h, i, \mathbf{m})$. Takes as input a common reference string $\mathsf{crs}$, the public key $h$, an index $i \in [n]$, and a message $\mathbf{m} \in \{0,1\}^B$ and outputs a ciphertext $\mathsf{ct}$. As above, the ciphertext $\mathsf{ct}$ can be written as a concatenation of two parts $\mathsf{ct} = (\mathsf{subct}_1, \mathsf{subct}_2)$ for blindness purposes to be defined below.

6. $\mathsf{SingleDec}(\mathsf{crs}, x, i, \mathsf{ct}_i)$. Takes as input a common reference string $\mathsf{crs}$, the secret key $x$, an index $i \in [n]$, and a ciphertext $\mathsf{ct}_i$ and outputs a message $m \in \{0,1\}$.

Whenever $\mathsf{SingleEnc}$ and $\mathsf{SingleDec}$ are defined, we require that $\mathsf{Enc}(\mathsf{crs}, h, \mathbf{M}) = (\mathsf{ct}_i)_{i \in [n]}$ for $\mathsf{ct}_i \leftarrow \mathsf{SingleEnc}(\mathsf{crs}, h, i, \mathbf{m}_i)$, where $\mathbf{m}_i$ denotes the $i$th row of $\mathbf{M}$. Similarly, we require that for $\mathsf{ct} = (\mathsf{ct}_i)_{i \in [n]}$, the decryption algorithm computes $m_i \leftarrow \mathsf{SingleDec}(\mathsf{crs}, x, i, \mathsf{ct}_i)$ for all $i \in [n]$ and outputs their concatenation.

**Correctness of Batch Encryption.** We define two notions of correctness of a batch encryption scheme, the first stronger than the second.

**Definition 3.1** (Batch Correctness)**.** *Letting* $\mathsf{crs} = \mathsf{Setup}(1^\lambda, 1^n)$*, then for all* $x, \mathbf{M}$*, it holds that taking* $h = \mathsf{Gen}(\mathsf{crs}, x)$*,* $\mathsf{ct} = \mathsf{Enc}(\mathsf{crs}, h, \mathbf{M})$*,* $\mathbf{m}' = \mathsf{Dec}(\mathsf{crs}, x, \mathsf{ct})$*, it holds that* $\mathbf{m}'_i = \mathbf{M}_{i,x_i}$ *for all* $i$ *with probability at least* $1 - 2^\lambda$ *over the randomness of* $\mathsf{Enc}$*.*

**Definition 3.2** ($\delta$-Pointwise-Correctness (for $\mathsf{SingleEnc}$))**.** *Letting* $\mathsf{crs} = \mathsf{Setup}(1^\lambda, 1^n)$*, then for all* $x, i, \mathbf{m}$*, it holds that taking* $h = \mathsf{Gen}(\mathsf{crs}, x)$*,* $\mathsf{ct}_i = \mathsf{SingleEnc}(\mathsf{crs}, h, i, \mathbf{m})$*,* $m' = \mathsf{SingleDec}(\mathsf{crs}, x, i, \mathsf{ct}_i)$*, it holds that* $m' = m_{x_i}$ *with probability at least* $1/2 + \delta$ *over the randomness of* $\mathsf{SingleEnc}$*.*

Note that $1/\mathrm{poly}(\lambda)$-pointwise-correctness implies batch correctness via repetition.

**Succinctness of Batch Encryption.**

**Definition 3.3.** *A batch encryption scheme is* $\alpha$*-succinct if letting* $\mathsf{crs} = \mathsf{Setup}(1^\lambda, 1^n)$*,* $h = \mathsf{Gen}(\mathsf{crs}, x)$ *for some* $x \in [B]^n$*, it holds that* $|h| \leq \alpha n \log B$*.*

**Definition 3.4.** *A batch encryption scheme is fully succinct if letting* $\mathsf{crs} = \mathsf{Setup}(1^\lambda, 1^n)$*,* $h = \mathsf{Gen}(\mathsf{crs}, x)$ *for some* $x \in [B]^n$*, it holds that* $|h| \leq p(\lambda)$ *for some fixed polynomial* $p(\lambda)$*.*

**Semantic Security of Batch Encryption.**

**Definition 3.5** (Batch Encryption Security)**.** *The security of a batch encryption scheme is defined using the following game between a challenger and adversary.*

1. *The adversary takes* $1^\lambda$ *as input, and sends* $1^n, x \in [B]^n$ *to the challenger.*
2. *The challenger generates* $\mathsf{crs} = \mathsf{Setup}(1^\lambda, 1^n)$ *and sends* $\mathsf{crs}$ *to the adversary.*
3. *The adversary generates* $\mathbf{M}^{(0)}, \mathbf{M}^{(1)} \in \{0,1\}^{n \times B}$ *such that* $\mathbf{M}^{(0)}_{i,x_i} = \mathbf{M}^{(1)}_{i,x_i}$ *for all* $i \in [n]$ *and sends them to the challenger.*
4. *The challenger computes* $h = \mathsf{Gen}(\mathsf{crs}, x)$ *and encrypts* $\mathsf{ct} = \mathsf{Enc}(\mathsf{crs}, h, M^{(\beta)})$ *for a random bit* $\beta \in \{0,1\}$*. It sends* $\mathsf{ct}$ *to the adversary.*
5. *The adversary outputs a bit* $\beta'$ *and wins if* $\beta' = \beta$*.*

*The batch encryption scheme is secure if no polynomial time adversary can win the above game with probability* $\geq 1/2 + 1/\mathrm{poly}(\lambda)$*.*

By a standard hybrid argument, the above definition is implied by the following security property for $\mathsf{SingleEnc}$.

**Definition 3.6** ($\mathsf{SingleEnc}$ Security)**.** *We say that a batch encryption scheme satisfies* $\mathsf{SingleEnc}$*-security if no polynomial time adversary can win the following game with probability* $\geq 1/2 + 1/\mathrm{poly}(\lambda)$*:*

1. *The adversary takes* $1^\lambda$ *as input, and sends* $1^n, x \in [B]^n, i \in [n]$ *to the challenger.*
2. *The challenger generates* $\mathsf{crs} = \mathsf{Setup}(1^\lambda, 1^n)$ *and sends* $\mathsf{crs}$ *to the adversary.*
3. *The adversary generates* $\mathbf{m}^{(0)}, \mathbf{m}^{(1)} \in \{0,1\}^B$ *s.t.* $\mathbf{m}^{(0)}_{x_i} = \mathbf{m}^{(1)}_{x_i}$ *and sends them to the challenger.*
4. *The challenger computes* $h = \mathsf{Gen}(\mathsf{crs}, x)$ *and encrypts* $\mathsf{ct} = \mathsf{SingleEnc}(\mathsf{crs}, h, i, \mathbf{m}^{(\beta)})$ *for a random bit* $\beta \in \{0,1\}$*. It sends* $\mathsf{ct}$ *to the adversary.*
5. *The adversary outputs a bit* $\beta'$ *and it wins if* $\beta' = \beta$*.*

**Relation to Chameleon Encryption and Laconic Oblivious Transfer.** For readers familiar with the notions of chameleon encryption [DG17a] and laconic oblivious transfer [CDG+17], we compare the notion of batch encryption to these objects.

First, we note that the notion of batch encryption is a significant weakening of the notion of a *chameleon encryption scheme* defined in [DG17a] in the following two ways. Most significantly, we do not require a trapdoor which supports finding collisions (namely, the "chameleon" part of chameleon encryption); this is crucial because our construction from LPN does not seem to have an associated trapdoor. Nevertheless, we show that batch encryption is sufficient to construct IBE. As well, our security definition is selective in the input $x$ rather than adaptive (that is, the adversary picks $x$ before seeing the $\mathsf{crs}$), which means that batch encryption does not obviously imply collision resistant hash functions (CRHF), but rather only target collision-resistance. In contrast, the hash function implicit in chameleon encryption is a CRHF).

On the other hand, batch encryption is essentially equivalent to laconic oblivious transfer as defined in [CDG+17], as long as you restrict the first message of the OT protocol to be a deterministic function of the $\mathsf{crs}$ and database $D$ (however, since receiver privacy is not required for laconic OT, any laconic OT scheme can be modified to have this property). Our transformations show that batch encryption (or laconic OT) is the right primitive from which to bootstrap and obtain IBE. Additionally, our new blindness property also has an interpretation in the language of laconic OT.

### 3.2 Defining Blind Batch Encryption

Next, we define the additional *blindness property* of a batch encryption scheme, which asserts that when encrypting a random message that is not known to the distinguisher, the ciphertext is "essentially" indistinguishable from uniform. More specifically, we allow a part of the ciphertext to not be indistinguishable from uniform so long as it does not reveal any information on $h$ or on the encrypted message.

**Definition 3.7** (Blindness)**.** *Let $\mathcal{BBENC} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a batch encryption scheme. Furthermore, suppose that $\mathsf{Enc}(\mathsf{crs}, h, \mathbf{M}; r) = E_1(\mathsf{crs}, h, \mathbf{M}; r)||E_2(\mathsf{crs}, h, \mathbf{M}; r)$ is some decomposition of $\mathsf{Enc}(\cdot)$ into two parts. We say that $\mathcal{BBENC}$ is* blind *if (1) the function $E_1(\mathsf{crs}, h, \mathbf{M}; r) = E_1(\mathsf{crs}; r)$ does not depend on the public key $h$ or message $\mathbf{M}$, and (2) no polynomial time adversary can win the following game with probability $\geq \frac{1}{2} + 1/\mathrm{poly}(\lambda)$.*

1. *The adversary takes $1^\lambda$ as input, and sends $1^n, x \in [B]^n$ to the challenger.*
2. *The challenger generates $\mathsf{crs} = \mathsf{Setup}(1^\lambda, 1^n)$ and computes $h = \mathsf{Gen}(\mathsf{crs}, x)$. It samples a random $\beta \leftarrow \{0,1\}$, a random message matrix $\mathbf{M} \leftarrow \{0,1\}^{n \times B}$, and encrypts $(\mathsf{subct}_1, \mathsf{subct}_2) \leftarrow \mathsf{Enc}(\mathsf{crs}, h, \mathbf{M})$. It then generates $\mathsf{ct}$ as follows.*
   - *If $\beta = 0$ then $\mathsf{ct} = (\mathsf{subct}_1, \mathsf{subct}_2)$.*
   - *If $\beta = 1$ then sample a random bit string $\mathsf{subct}_2'$ of the same length as $\mathsf{subct}_2$. Set $\mathsf{ct} = (\mathsf{subct}_1, \mathsf{subct}_2')$.*

   *The challenger sends $\mathsf{crs}, \mathsf{ct}$ to the adversary (note that $\mathbf{M}$ is* not *sent to the adversary).*
3. *The adversary outputs a bit $\beta'$ and it wins if $\beta' = \beta$.*

Again, the above definition of blindness is implied by an analogous blindness property for $\mathsf{SingleEnc}$ via a standard hybrid argument. If $\mathcal{BBENC}$ is a blind batch encryption scheme, we call $\mathsf{Enc} = E_1||E_2$ the *blind decomposition* of $\mathsf{Enc}$ and adopt the notation that outputs of $E_1$ are denoted by $\mathsf{subct}_1$ and outputs of $E_2$ are denoted by $\mathsf{subct}_2$.

**From block size $B$ to block size 2.** Although our construction of batch encryption itself from LPN constructs a scheme with large block size, the lemma below shows that we can work with block size 2, without loss of generality. The proof of the lemma is in Appendix A.1.

**Lemma 3.1.** *Suppose that there is an $\alpha$-succinct (blind) batch encryption scheme with block size $B$. Then, there is an $\alpha$-succinct (blind) batch encryption scheme with block size 2.*

**From $\alpha$-Succinct to Fully Succinct (Blind) Batch Encryption.** We show that fully succinct (blind) batch encryption can be built from 1/2-succinct (blind) batch encryption. The construction and proof are similar to the laconic OT bootstrapping theorem of Cho et al. [CDG+17]. However, to preserve blindness, we make use of blind garbled circuits (defined in Section 4), similar to its use in Section 5 and Section 6. We state the lemma below and provide the proof in Appendix A.2.

**Lemma 3.2.** *Suppose that there is a 1/2-succinct (blind) batch encryption scheme with block size $B = 2$ and a (blind) garbling scheme. Then, there is a fully succinct (blind) batch encryption scheme with block size $B = 2$.*

### 3.3 Blind Batch Encryption from CDH

In this section, we construct blind batch encryption from the CDH assumption. The scheme has perfect correctness, is *fully succinct*, and has block size $B = 2$. This construction is inspired by the Chameleon Encryption construction in [DG17a] but does not require a trapdoor. Let $\mathcal{G}$ be a group sampler as described in Section 2.2. Recall the Goldreich-Levin encoding/decoding procedure as per Section 2.4. The blind batch encryption scheme is as follows.

1. $\mathsf{CDH\text{-}BE.Setup}(1^\lambda, 1^n)$. Sample $(\mathbb{G}, g, q) \leftarrow \mathcal{G}(1^\lambda)$. Sample $\alpha_{i,b} \leftarrow \mathbb{Z}_q$ for $i \in [n]$, $b \in \{0,1\}$. Define $g_{i,b} = g^{\alpha_{i,b}}$. Output $\mathsf{crs} = ((\mathbb{G}, g, q), \{g_{i,b}\}_{i,b})$.
2. $\mathsf{CDH\text{-}BE.Gen}(\mathsf{crs}, x)$. Output $h = \prod_i g_{i,x_i}$.
3. $\mathsf{CDH\text{-}BE.SingleEnc}(\mathsf{crs}, h, i, \mathbf{m})$. Sample $r \leftarrow \mathbb{Z}_q$. For all $j \neq i$ and for all $b \in \{0,1\}$ compute: $\hat{g}_{j,b} = g_{j,b}^r$. Compute $\hat{g}_{i,b} = h^r g_{i,b}^{-r}$, and let $\mu_{i,b} = \mathsf{gl\text{-}enc}(\hat{g}_{i,b}, \mathbf{m}_b)$. Output $\mathsf{ct} = \big(\mathsf{subct}_1 = \{\hat{g}_{j,b}\}_{j\neq i, b\in\{0,1\}}, \mathsf{subct}_2 = \{\mu_{i,b}\}_{b\in\{0,1\}}\big)$.
4. $\mathsf{CDH\text{-}BE.SingleDec}(\mathsf{crs}, x, i, \mathsf{ct})$. Given $\mathsf{ct} = \big(\{\hat{g}_{j,b}\}_{j\neq i, b\in\{0,1\}}, \{\mu_{i,b}\}_{b\in\{0,1\}}\big)$. Compute $\hat{g}_{i,x_i} = \prod_{j\neq i} = \hat{g}_{i,x_i}$. Output $m = \mathsf{gl\text{-}dec}(\hat{g}_{i,x_i}, \mu_{i,x_i})$.

Correctness follows immediately by definition. Moreover, note that this scheme is *fully succinct* (see Definition 3.4; note that $h \in \mathbb{G}$ has a fixed $\mathrm{poly}(\lambda)$ size representation by assumption).

**Lemma 3.3.** *The scheme $\mathsf{CDH\text{-}BE}$ is secure under the $\mathsf{CDH}_\mathcal{G}$ assumption.*

*Proof.* Consider the following game between a challenger and an adversary.

1. The adversary takes $1^\lambda$ as input, and sends $1^n, x \in \{0,1\}^n, i \in [n]$, to the challenger.

2. The challenger generates $\mathsf{crs} = \mathsf{CDH\text{-}BE.Setup}(1^\lambda, 1^n)$, i.e. a group $(\mathbb{G}, g, q)$ and collection of $g_{j,b}$. It computes $h = \mathsf{CDH\text{-}BE.Gen}(x)$. It then samples $r \leftarrow \mathbb{Z}_q$ and computes $\hat{g}_{j,b} = g_{j,b}^r$ for all $j \neq i$, $b \in \{0,1\}$, as well as $\hat{g}_{i,x_i} = h^r g_{i,x_i}^{-r}$. It sends $\mathsf{crs}$ and the computed $\hat{g}$ values to the adversary.

3. The adversary returns $g'$.

4. The challenger declares that the adversary wins if $g' = h^r g_{i,1-x_i}^{-r}$.

We will prove that all polynomial time adversaries have negligible advantage in the above game. By the Goldreich-Levin theorem (see Section 2.4), this implies the security of the scheme as per Definition 3.6.

To see that the above holds, an adversary against the above game, and consider an input to the $\mathsf{CDH}_\mathcal{G}$ problem consisting of $(\mathbb{G}, g, q), g^{a_1}, g^{a_2}$. We will show how to produce a challenger for the above game, so that when the adversary succeeds, the value $g^{a_1 a_2}$ can be computed. The challenger, upon receiving $1^n, x, i$ will do the following. Generate $\alpha_{j,b} \leftarrow \mathbb{Z}_q$ for all $j \neq i$, $b \in \{0,1\}$, and also $\alpha_{i,1-x_i}$. Conceptually, we will associate $a_1$ with the value $r$ to be generated by the challenger, and $a_2$ with the difference $(\alpha_{i,x_i} - \alpha_{i,1-x_i})$.[2]

Following this intuition, the challenger will generate $g_{i,b} = g^{\alpha_{j,b}}$ for all $j \neq i$, $b \in \{0,1\}$ as well as for $(j,b) = (i, 1-x_i)$. Then generate $g_{i,x_i} = g_{i,1-x_i} \cdot g^{a_2}$. Generate $\hat{g}_{j,b} = (g^{a_1})^{\alpha_{j,b}}$ for all $j \neq i$, $b \in \{0,1\}$. We are left with generating $\hat{g}_{i,x_i} = h^r g_{i,x_i}^{-r} = \prod_{j \neq i} g_{j,x_j}^r = \prod_{j \neq i} \hat{g}_{j,x_j}$, which can be derived from previously computed values. Note that the computed values are within negligible statistical distance of their distribution in the real experiment. If the adversary manages to compute $g' = h^r g_{i,1-x_i}^{-r} = \left(\prod_{j \neq i} \hat{g}_{j,x_j}\right) \cdot (g_{i,x_i}/g_{i,1-x_i})^r = \left(\prod_{j \neq i} \hat{g}_{j,x_j}\right) \cdot g^{a_1 a_2}$, then the product $\prod_{j \neq i} \hat{g}_{j,x_j}$ can be canceled out and a solution to $\mathsf{CDH}_\mathcal{G}$ is achieved. □

**Lemma 3.4.** *The scheme* CDH-BE *is blind under the* $\mathsf{CDH}_\mathcal{G}$ *assumption.*

*Proof.* Consider the game in Definition 3.7. We first of all note that in our scheme, $\mathsf{subct}_1$ is independent of $h, \mathbf{m}$ and therefore the marginal distribution of $\mathsf{subct}_1$ is identical regardless of the value of $\beta$. From the properties of gl-enc (see Section 2.4), if $\mathbf{m}$ is uniform then the $\mu_{i,b}$ values are uniformly distributed. It follows that any adversary will have exactly $1/2$ probability to win the blindness game. □

### 3.4 Batch Encryption from LPN

In this section we present a candidate construction from LPN with noise rate $\Omega(\log^2(n)/n)$. Specifically, we will show an LPN based construction which has $\delta$-pointwise correctness for $\delta = 1/\mathrm{poly}(n)$, is $\frac{1}{2}$-*succinct*, and has block size $B = n$. Our construction is based on a collision resistant hash function construction of [BLVW17]. See Section 2.3 for details about the assumption and the CRH candidate. Unfortunately, we are unable to prove blindness for this candidate. As explained above, the $\delta$ point-wise correctness can be amplified, however this amplification does not preserve the

[2] In fact, this correspondence only needs to hold in the exponent. Specifically, note that both $g^{(\alpha_{i,x_i} - \alpha_{i,1-x_i})}$ and $g^{a_2}$ are statistically indistinguishable from uniform in $\langle g \rangle$ and therefore from each other.

blindness property. Therefore, even though our $\delta$-point-wise correct candidate is blind, we cannot amplify it to have batch correctness without giving up blindness.

We introduce the following notation. For any number $j \in [B]$ we define $\mathsf{ind}(j) \in \{0,1\}^B$ to be the vector with 1 in the $j$-th coordinate and 0 in all other coordinates. Note that for a matrix $\mathbf{A} \in \{0,1\}^{k \times B}$ (for arbitrary $k$) it holds that $\mathbf{A} \cdot \mathsf{ind}(j)$ is exactly the $j$-th column of $\mathbf{A}$.

1. $\mathsf{LPN\text{-}BE.Setup}(1^\lambda, 1^n)$. Recall that $B = n$, and assume w.l.o.g that $\lambda \leq n$ (otherwise redefine $n = \lambda$ and proceed with the new value, which only strengthens the constructed object). We define $\tilde{n} = \frac{n \log B}{2} = \frac{n \log n}{2}$ and a parameter $\epsilon = \log n / n = \Omega(\log^2(\tilde{n})/\tilde{n})$ to be used below. Sample $\mathbf{A}_1, \ldots, \mathbf{A}_n \leftarrow \{0,1\}^{\tilde{n} \times B}$ (we will also denote $\mathbf{A} = [\mathbf{A}_1 \| \cdots \| \mathbf{A}_n]$). Output $\mathsf{crs} = \{\mathbf{A}_i\}_{i \in [n]}$.

2. $\mathsf{LPN\text{-}BE.Gen}(\mathsf{crs}, x)$. Output $\mathbf{h} = \sum_{i \in [n]} \mathbf{A}_i \cdot \mathsf{ind}(x_i)$.

3. $\mathsf{LPN\text{-}BE.SingleEnc}(\mathsf{crs}, \mathbf{h}, i, \mathbf{m})$. Define $\mathbf{A}_{-i} = [\mathbf{A}_1 \| \cdots \| \mathbf{A}_{i-1} \| \mathbf{A}_{i+1} \| \cdots \| \mathbf{A}_n]$. For all $j \in [B]$ sample $\mathbf{s}^{(j)} \leftarrow \{0,1\}^{\tilde{n}}$ and $\mathbf{e}^{(j)} \leftarrow \mathsf{Ber}_\epsilon^{(n-1)B+1}$. Compute

$$\mathbf{v}^{(j)} = \mathbf{s}^{(j)} [\mathbf{A}_{-i} \| \mathbf{A}_i \cdot \mathsf{ind}(j) - \mathbf{h}] + \mathbf{e}^{(j)} + [0, \ldots, 0, \mathbf{m}_j] \ .$$

   Output $\mathsf{ct} = \mathsf{subct}_2 = \{\mathbf{v}^{(j)}\}_{j \in [B]}$.

4. $\mathsf{LPN\text{-}BE.SingleDec}(\mathsf{crs}, x, i, \mathsf{ct})$. Given $\mathsf{ct} = \{\mathbf{v}^{(j)}\}_{j \in [B]}$, define

$$\hat{\mathbf{x}}_{-i} = [\mathsf{ind}(x_1) \| \cdots \| \mathsf{ind}(x_{i-1}) \| \mathsf{ind}(x_{i+1}) \| \cdots \| \mathsf{ind}(x_n) \| 1]^\dagger \ ,$$

   where $\dagger$ represents vector transpose. Output $m = \mathbf{v}^{(x_i)} \cdot \hat{\mathbf{x}}_{-i}$.

**Lemma 3.5.** *The scheme* LPN-BE *is* $1/\mathrm{poly}(n)$*-pointwise correct.*

*Proof.* Let $\mathsf{crs}, x, i, \mathbf{m}$ be arbitrary, and consider computing $h = \mathsf{Gen}(\mathsf{crs}, x)$, $\mathsf{ct} = \mathsf{SingleEnc}(\mathsf{crs}, h, i, \mathbf{m})$ and $m' = \mathsf{SingleDec}(\mathsf{crs}, x, i, \mathsf{ct})$, then by definition

$$\begin{aligned} m' &= \left( \mathbf{s}^{(j)} [\mathbf{A}_{-i} \| \mathbf{A}_i \cdot \mathsf{ind}(x_i) - \mathbf{h}] + \mathbf{e}^{(x_i)} + [0, \ldots, 0, \mathbf{m}_j] \right) \hat{\mathbf{x}}_{-i} \\ &= \mathbf{m}_j + \mathbf{e}^{(x_i)} \cdot \hat{\mathbf{x}}_{-i} \ , \end{aligned}$$

but since $\mathbf{e}^{(x_i)}$ is Bernoulli with parameter $\epsilon$, and the hamming weight of $\hat{\mathbf{x}}_{-i}$ is exactly $n$ by definition, then $\mathbf{e}^{(x_i)} \cdot \hat{\mathbf{x}}_{-i}$ is Bernoulli with parameter $\epsilon' \leq 1/2 - e^{-2\epsilon n}$. Since we set $\epsilon = \log n / n$, pointwise correctness follows. □

**Lemma 3.6.** *The scheme* LPN-BE *is secure under the* $\mathsf{LPN}_{\tilde{n},\epsilon}$ *assumption (we recall that* $\epsilon = \Omega(\log^2(\tilde{n})/\tilde{n})$*).*

*Proof.* We consider the SingleEnc security game in Definition 3.6 (recall that this is sufficient for full batch security). We will prove that the view of the adversary is computationally indistinguishable from one where all $\mathbf{v}^{(j)}$ are uniformly random for all $j \neq x_i$. Security will follow.

Consider a challenger that receives an LPN challenge of the form $\mathbf{A}'_1, \ldots, \mathbf{A}'_n \in \{0,1\}^{\tilde{n} \times B}$, $\{\mathbf{b}_{j,k}\}_{j \in [B] \setminus \{x_i\}, k \in [n]]}$, where $\mathbf{b}_{j,k}$ are either all uniform or are of the form $\mathbf{b}_{j,k} = \mathbf{s}^{(j)} \mathbf{A}'_k + \mathbf{e}_{j,k}$. (Note that the challenge does not actually depend on $x_i$, we can just take $j \in [B-1]$ and map the values to $[B] \setminus \{x_i\}$ after the fact.)

Upon receiving $x, i$ from the adversary, the challenger computes $\mathbf{h} = \sum_{i\in[n]} \mathbf{A}'_i \cdot \mathsf{ind}(x_i)$. Then, for all $k \neq i$ it sets $\mathbf{A}_k = \mathbf{A}'_k$, and then sets $\mathbf{A}_i$ as follows. Set $\mathbf{A}_i \cdot \mathsf{ind}(x_i) = \mathbf{A}'_i \cdot \mathsf{ind}(x_i)$ (recall that multiplying by $\mathsf{ind}(j)$ is equivalent to selecting the $j$-th column), and for all $j \neq x_i$ set $\mathbf{A}_i \cdot \mathsf{ind}(j) = \mathbf{A}'_i \cdot \mathsf{ind}(j) + \mathbf{h}$. Note that since $\mathbf{A}_i \cdot \mathsf{ind}(x_i) = \mathbf{A}'_i \cdot \mathsf{ind}(x_i)$ it holds that $\mathbf{h} = \sum_{i\in[n]} \mathbf{A}_i \cdot \mathsf{ind}(x_i)$, and indeed $\mathsf{crs} = \{\mathbf{A}_1, \ldots, \mathbf{A}_n\}, \mathbf{h}, x, i$ are distributed identically to the original game.

The challenger sends $\mathsf{crs}, \mathbf{h}$ to the adversary and receives the message vectors. It then samples $\mathbf{s}^{(x_i)}$, $\mathbf{e}^{(x_i)}$ itself and generates $\mathbf{v}^{(x_i)}$ properly. For all $j \neq x_i$ generate

$$\mathbf{v}^{(j)} = [\mathbf{b}_{j,[n]\setminus\{i\}} \| \mathbf{b}_{j,i} \cdot \mathsf{ind}(j)] + [0, \ldots, 0, \mathbf{m}_j] \ ,$$

where $\mathbf{b}_{j,[n]\setminus\{i\}}$ is the concatenation of all $\mathbf{b}_{j,k}$ for $k \neq i$ in order. We notice that if the vectors $\{\mathbf{b}_{j,k}\}$ were generated from an LPN distribution, then $\mathbf{v}^{(j)}$ has the correct distribution. This is because we defined $\mathbf{A}_i \cdot \mathsf{ind}(j) - \mathbf{h} = \mathbf{A}'_i \cdot \mathsf{ind}(j)$. On the other hand, if $\{\mathbf{b}_{j,k}\}$ are uniform then all $\mathbf{v}^{(j)}$, $j \neq x_i$ are uniform. Security thus follows. □

# 4 Blind Garbled Circuits

In this section, we define the notion of a blind garbled circuit and show a construction assuming only one-way functions. Indeed, we observe that the widely used "point-and-permute" garbled circuit construction [BMR90, Rog91] is in fact blind. We start with the definition of standard garbled circuits and proceed to define and construct blind garbled circuits.

## 4.1 Definitions

**Definition 4.1** (Garbled Circuits)**.** *A garbling scheme consists of three algorithms* $(\mathsf{Garble}, \mathsf{Eval}, \mathsf{Sim})$ *where:*

1. $\mathsf{Garble}(1^\lambda, 1^n, 1^m, C)$ *is a PPT algorithm that takes as input the security parameter* $\lambda$ *and a circuit* $C : \{0,1\}^n \to \{0,1\}^m$*, and outputs a garbled circuit* $\widehat{C}$ *along with input labels* $(\mathsf{lab}_{i,b})_{i\in[n],b\in\{0,1\}}$ *where each label* $\mathsf{lab}_{i,b} \in \{0,1\}^\lambda$*.*

2. $\mathsf{Eval}(1^\lambda, \widehat{C}, \widehat{L})$ *is a deterministic algorithm that takes as input a garbled circuit* $\widehat{C}$ *along with a set of* $n$ *labels* $\widehat{L} = (\mathsf{lab}_i)_{i\in[n]}$*, and outputs a string* $y \in \{0,1\}^m$*.*

3. $\mathsf{Sim}(1^\lambda, 1^{|C|}, 1^n, y)$ *is a PPT algorithm that takes as input the security parameter, the description length of* $C$*, an input length* $n$ *and a string* $y \in \{0,1\}^m$*, and outputs a simulated garbled circuit* $\widetilde{C}$ *and labels* $\widetilde{L}$*.*

*We often omit the first input to these algorithms (namely,* $1^\lambda$*) when it is clear from the context. We require that the garbling scheme satisfies two properties:*

1. *Correctness: For all circuits* $C$*, inputs* $x$*, and all* $(\widehat{C}, (\mathsf{lab}_{i,b})_{i,b}) \leftarrow \mathsf{Garble}(C, x)$ *and* $\widehat{L} = (\mathsf{lab}_{i,x_i})_{i\in[n]}$*, we have that* $\mathsf{Eval}(\widehat{C}, \widehat{L}) = C(x)$*.*

2. *Simulation Security: for all circuits* $C : \{0,1\}^n \to \{0,1\}^m$ *and all inputs* $x \in \{0,1\}^n$*, the following two distributions are computationally indistinguishable:*

$$\left\{(\widehat{C}, \widehat{L}) : (\widehat{C}, \mathsf{lab}_{i,b})_{i,b}) \leftarrow \mathsf{Garble}(C, x), \widehat{L} = (\mathsf{lab}_{i,x_i})_{i\in[n]}\right\}$$
$$\approx_c \left\{(\widetilde{C}, \widetilde{L}) : (\widetilde{C}, \widetilde{L}) \leftarrow \mathsf{Sim}(1^\lambda, 1^{|C|}, 1^n, C(x))\right\} \ .$$

The traditional notion of security of a garbled circuit requires that the garbling $\widehat{C}$ of a circuit $C$ and the garbled labels $\widehat{L}$ corresponding to an input $x$ together reveal $C(x)$ and nothing more (except the size of the circuit $C$ and the input $x$). Formally, this is captured by a simulation definition which requires that a simulator who is given only $C(x)$ can faithfully simulate the joint distribution of $\widehat{C}$ and $\widehat{L}$. Blindness requires that the simulator's output is *uniformly random*. Of course, this is simply unachievable if the distinguisher is given the circuit $C$ and the input $x$, or if the distribution of $C(x)$ is not uniformly random. However, blindness only refers to the setting where the distribution of $C(x)$ is uniformly random.

**Definition 4.2** (Blind Garbled Circuits)**.** *A garbling scheme* $(\mathsf{Garble}, \mathsf{Eval}, \mathsf{Sim})$ *is called* blind *if the distribution* $\mathsf{Sim}(1^\lambda, 1^c, 1^n, U_m)$*, representing the output of the simulator on a completely uniform output, is indistinguishable from a completely uniform bit string. (Note that the distinguisher must not know the random output value that was used for the simulation.)*

## 4.2 A Blind Garbled Circuit Construction from Any PRG

Our blind garbled circuits are essentially identical to the point-and-permute garbled circuits of [BMR90, Rog91]. The details of the construction will be important for our purposes so we state them explicitly. We note that without loss of generality we can garble a universal circuit $\mathcal{U} : \{0,1\}^n \to \{0,1\}^m$ of the appropriate size.

As a building block we use a family of pseudorandom functions $\mathsf{PRF}$ with uniform seeds in $\{0,1\}^\lambda$. The input space we will use is actually only *polynomial* (proportional to the size of the circuit being garbled) so in fact a full blown PRF is not required and a PRG would suffice, however it would be more convenient to present the garbled circuit construction using the PRF notation. The output space we use is $\{0,1\}^{\lambda+1}$.

1. $\mathsf{BGC.Garble}(1^\lambda, \mathcal{U})$. For every wire $w$ in $\mathcal{U}$, sample two PRF seeds $s_{w,b} \leftarrow \{0,1\}^\lambda$ for $b \in \{0,1\}$, as well as a permutation bit $\alpha_w \in \{0,1\}$.

   Let $f_g : \{0,1\}^2 \to \{0,1\}$ be the function computed by the Boolean gate $g$. For every $\beta_1, \beta_2 \in \{0,1\}$, let $\beta_3 := f_g(\alpha_1 \oplus \beta_1, \alpha_2 \oplus \beta_2) \oplus \alpha_3$. For every gate $g$ with input wires $w_1, w_2$ and output wire $w_3$, and for all $\beta_1, \beta_2 \in \{0,1\}$, we compute the table entry

   $$T_g^{(\beta_1,\beta_2)} = (s_{w_3,\alpha_3\oplus\beta_3} \| \beta_3) \;\oplus\; \mathsf{PRF}_{s_{w_1,\alpha_1\oplus\beta_1}}(g \| \beta_1 \| \beta_2) \;\oplus\; \mathsf{PRF}_{s_{w_2,\alpha_2\oplus\beta_2}}(g \| \beta_1 \| \beta_2).$$

   Let $\mathsf{in}_1, \ldots, \mathsf{in}_n$ denote the input wires, and $\mathsf{out}_1, \ldots, \mathsf{out}_m$ denote the output wires. Then, the garbled circuit $\widehat{C}$ contains all table entries $T = \{T_g^{(\beta_1,\beta_2)}\}_{g\in\mathcal{U},\beta_1,\beta_2\in\{0,1\}}$ as well as all values $A = \{\alpha_{\mathsf{out}_j}\}_{j\in[m]}$. The input labels are $\mathsf{lab}_{i,b} = (s_{\mathsf{in}_i,b} \| b \oplus \alpha_{\mathsf{in}_i})$. This is equivalent to computing $\beta = b \oplus \alpha_{\mathsf{in}_i}$ and then setting $\mathsf{lab}_{i,b} = (s'_{\mathsf{in}_i,\beta} \| \beta)$.

2. $\mathsf{BGC.Eval}(1^\lambda, \widehat{C}, \widehat{L})$. Given a garbled circuit $\widehat{C} = (T, A)$ together with a set of labels $\widehat{L} = \{\mathsf{lab}_i\}_{i\in[n]}$, do the following. Parse $\mathsf{lab}_i$ as $(s'_{\mathsf{in}_i} \| \beta_{\mathsf{in}_i})$. For every gate $g$ in topological order, let $w_1, w_2$ be its input wires and let $w_3$ be its output wire. Then given $(s'_{w_1} \| \beta_{w_1})$, $(s'_{w_2} \| \beta_{w_2})$, compute

   $$(s'_{w_3} \| \beta_{w_3}) = T_g^{(\beta_{w_1},\beta_{w_2})} \oplus \mathsf{PRF}_{s'_{w_1,\beta_{w_1}}}(g \| \beta_{w_1} \| \beta_{w_2}) \oplus \mathsf{PRF}_{s'_{w_2,\beta_{w_2}}}(g \| \beta_{w_1} \| \beta_{w_2}) \;.$$

   Finally, upon obtaining $\beta_{\mathsf{out}_j}$ for all $j$, output $\mathsf{out}_j = \beta_{\mathsf{out}_j} \oplus \alpha_{\mathsf{out}_j}$ for all $j \in [m]$.

3. $\mathsf{BGC.Sim}(1^\lambda, 1^{|\mathcal{U}|}, 1^n, y)$. For every wire $w$ in the universal circuit $\mathcal{U}$ sample random values $\beta^*_w \leftarrow \{0,1\}$ and $s'_{w,\beta^*_w} \leftarrow \{0,1\}^\lambda$. Generate the table entries $T_g^{(\beta_{w_1},\beta_{w_2})}$ as follows. If $(\beta_{w_1},\beta_{w_2}) \neq (\beta^*_{w_1},\beta^*_{w_2})$, then set $T_g^{(\beta_{w_1},\beta_{w_2})}$ to a completely uniform string. Otherwise set

$$T_g^{(\beta^*_1,\beta^*_2)} = (s'_{w_3,\beta^*_3} \| \beta^*_3) \oplus \mathsf{PRF}_{s'_{w_1,\beta^*_1}}(g \| \beta^*_1 \| \beta^*_2) \oplus \mathsf{PRF}_{s'_{w_2,\beta^*_2}}(g \| \beta^*_1 \| \beta^*_2) \ .$$

Finally set $\alpha_{\mathsf{out}_j} = \beta^*_{\mathsf{out}_j} \oplus y_j$. Let $\widetilde{C}$ be the resulting $(T, A)$, and let $\mathsf{lab}_i = (s'_{\mathsf{in}_i,\beta^*_{\mathsf{in}_i}} \| \beta^*_{\mathsf{in}_i})$.

Since our construction is an instantiation of the point-and-permute construction, its correctness and simulation security have already been established in previous works [BMR90, Rog91]. We will prove blindness next.

**Lemma 4.1.** *The garbling scheme* $\mathsf{BGC}$ *is blind.*

*Proof.* We prove that if use a random $y \leftarrow U_m$ then the output of the simulator is perfectly uniformly random. Consider a random $y$ and let $(\widetilde{C}, \widetilde{L}) = \mathsf{Sim}(1^\lambda, 1^c, 1^n, y)$. Recall that in our construction $\widetilde{C} = (T, A)$ with structure as described in the construction. We notice that if $y$ is uniformly random, then $A$ is uniformly random and independent of $T$. It is therefore only left to look at the marginal distribution of $T$ and prove that it is uniform.

We again recall that $T$ is a collection of gate table entries $T_g^{(\beta_1,\beta_2)}$ and furthermore, by the definition of the simulator algorithm, each wire $w$ is associated with $\beta^*_w$ and it holds that for all $g$ with inputs $w_1, w_2$ and output $w_3$, all entries $T_g^{(\beta_1,\beta_2)}$ except $T_g^{(\beta^*_{w_1},\beta^*_{w_2})}$ are already uniform and independent of all other values in $\widetilde{C}$. We recall that the value $T_g^{(\beta^*_{w_1},\beta^*_{w_2})}$ contains an XOR with $(s'_{w_3,\beta^*_{w_3}} \| \beta^*_{w_3})$.

Now we consider scanning the gates of $\mathcal{U}$ in a reverse topological order, and show that all gates we encounter in the scan, the value $T_g^{(\beta^*_{w_1},\beta^*_{w_2})}$ is uniform and independent of all other values. We maintain the invariant that if our scan is currently at gate $g$ with inputs $w_1, w_2$ and output $w_3$ then $(s'_{w_3,\beta^*_{w_3}} \| \beta^*_{w_3})$ are uniform and independent of all gate tables we scanned so far. Clearly this holds for output gates at the beginning of the scan, since we randomized $A$. If this invariant holds, then $(s'_{w_3,\beta^*_{w_3}} \| \beta^*_{w_3})$ is uniform and therefore completely randomizes $T_g^{(\beta^*_{w_1},\beta^*_{w_2})}$. Furthermore, this value now becomes independent of $(s'_{w_1,\beta^*_{w_1}} \| \beta^*_{w_1})$ and $(s'_{w_2,\beta^*_{w_2}} \| \beta^*_{w_2})$. It follows that the invariant is thus maintained since the output of the next gate in the reverse topological order can only be an input to gates that were already scanned and therefore its $(s'_{w,\beta^*_w} \| \beta^*_w)$ value is uniform and independent of the table entries of all scanned gates. Propagating this invariant all the way to the input wires, the blindness property follows. □

# 5 Weakly Compact Blind IBE

## 5.1 Defining Weakly Compact Blind IBE

We now begin our construction of anonymous IBE from blind batch encryption and blind garbled circuits; along the way, we will also construct IBE from batch encryption. As noted earlier, we construct anonymous IBE as a consequence of building a stronger object which we call *blind IBE*. Similar in nature to the blindness property of batch encryption (Definition 3.7), we say that an IBE

scheme is blind if, when encrypting (under some identity $\mathsf{id}^*$) a random message that is not known to the distinguisher, the ciphertext is "essentially" indistinguishable from uniform, even given any polynomial number of secret keys $\{\mathsf{sk}_{\mathsf{id}}\}$ possibly including $\mathsf{sk}_{\mathsf{id}^*}$.

**Definition 5.1** (Blind IBE)**.** *An IBE scheme satisfies IND-BLIND-ID-CPA security if (1) it satisfies IND-ID-CPA security and (2) the function* $\mathsf{Enc}(\mathsf{pp}, \mathsf{mpk}, \mathsf{id}, m; r)$ *can be expressed as a concatenation* $E_1(\mathsf{pp}; r)||E_2(\mathsf{pp}, \mathsf{mpk}, \mathsf{id}, m; r)$ *such that no PPT adversary* $\mathcal{A}$ *can win the following game with probability greater than* $\frac{1}{2} + \mathrm{negl}(\lambda)$*:*

1. $\mathsf{pp} \leftarrow \mathsf{Params}(1^\lambda||1^t)$
2. $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(\mathsf{pp})$
3. $(\mathsf{id}^*, \mathsf{st}) \leftarrow \mathcal{A}^{\mathsf{Keygen}(\mathsf{pp},\mathsf{msk},\cdot)}(\mathsf{mpk})$
4. $m \xleftarrow{\$} \mathcal{M}$
5. $(\mathsf{subct}_1, \mathsf{subct}_2) \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{mpk}, \mathsf{id}^*, m) = (E_1(\mathsf{pp}; r), E_2(\mathsf{pp}, \mathsf{mpk}, \mathsf{id}^*, m; r))$
6. $\beta \xleftarrow{\$} \{0,1\}$. *If* $\beta = 1$, $\mathsf{subct}_2 \xleftarrow{\$} \{0,1\}^{|\mathsf{subct}_2|}$.
7. $\beta' \leftarrow \mathcal{A}^{\mathsf{Keygen}(\mathsf{pp},\mathsf{msk},\cdot)}(\mathsf{st}, (\mathsf{subct}_1, \mathsf{subct}_2))$
8. $\mathcal{A}$ *wins if and only if* $\beta' = \beta$.

*We call* $\mathsf{Enc} = E_1||E_2$ *the* blind decomposition *of* $\mathsf{Enc}$.

**Lemma 5.1.** *Any blind IBE scheme is also an anonymous IBE scheme.*

*Proof.* Consider an adversary $\mathcal{A}$ playing the IND-ANON-ID-CPA security game; $\mathcal{A}$ is eventually given a challenge $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{mpk}, \mathsf{id}_b, m_b)$ where $(\mathsf{id}_0, m_0)$ and $(\mathsf{id}_1, m_1)$ are the challenge id-message pairs chosen by $\mathcal{A}$. For each $b \in \{0,1\}$, it is certainly the case that $\mathcal{A}$ cannot distinguish whether it was given $\mathsf{ct}_{\mathsf{id}_b,m_b} \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{mpk}, \mathsf{id}_b, m_b)$ or $\mathsf{ct}_{\mathsf{id}_b,m} \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{mpk}, \mathsf{id}_b, m)$ where $m \xleftarrow{\$} \mathcal{M}$ is a uniformly random message; this follows from ordinary IBE security. Additionally, by blind IBE security, $\mathcal{A}$ also cannot distinguish whether it is given $\mathsf{ct}_{\mathsf{id}_b,m}$ as above or $\tilde{\mathsf{ct}}_{\mathsf{id}_b,m} \leftarrow E_1(\mathsf{pp}; r)||C$ for $C \xleftarrow{\$} \{0,1\}^{|E_2(\mathsf{pp},\mathsf{mpk},\mathsf{id}_b,m;r)|}$. But $\tilde{\mathsf{ct}}_{\mathsf{id}_0,m}$ and $\tilde{\mathsf{ct}}_{\mathsf{id}_1,m}$ are drawn from identical distributions, so we conclude that $\mathcal{A}$ cannot distinguish whether it was given $\mathsf{ct}_{\mathsf{id}_0,m_0}$ or $\mathsf{ct}_{\mathsf{id}_1,m_1}$, as desired. □

Our overall goal is to construct (blind) IBE from (blind) batch encryption; this will be done in two steps. In this section, we construct what we call *weakly compact (blind) IBE*, which is intuitively an IBE scheme for any $T = \mathrm{poly}(\lambda)$ identities which is at least slightly more efficient than the trivial "IBE scheme" consisting of $T$ independent PKE schemes (one for each identity), which has $|\mathsf{mpk}| = T \cdot \mathrm{poly}(\lambda)$. Indeed, all we require is that $|\mathsf{mpk}|$ grows sublinearly with $T$. In Section 6, we show that full (blind) IBE can be bootstrapped from weakly compact (blind) IBE.

**Definition 5.2** (Weakly Compact IBE)**.** *A weakly compact IBE scheme consists of five PPT algorithms* $(\mathsf{Params}, \mathsf{Setup}, \mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec})$ *with the same syntax as an IBE scheme. What distinguishes a weakly compact IBE scheme from a full IBE scheme is the following weakened* efficiency *requirements:*

- $\mathsf{Params}$ *now takes as input* $1^\lambda||1^T$ *where* $T = 2^t$ *is the* number of identities. *This means that all five algorithms now run in time* $\mathrm{poly}(T, \lambda)$ *rather than* $\mathrm{poly}(\log T, \lambda)$.[3]
- Weak Compactness: *we require that* $|\mathsf{mpk}| = O(T^{1-\epsilon}\mathrm{poly}(\lambda))$ *for some* $\epsilon > 0$.
- *Security still holds with respect to adversaries running in time* $\mathrm{poly}(\lambda)$, *not* $\mathrm{poly}(\lambda, T)$.[3]

**Definition 5.3.** *A weakly compact blind IBE scheme is a weakly compact IBE scheme satisfying IND-BLIND-ID-CPA security.*

We will construct weakly compact (blind) IBE from the following building blocks: (1) (blind) batch encryption, (2) (blind) garbled circuits, and (3) (blind) public key encryption, where blind PKE is defined as follows.

**Definition 5.4** (Blind Public Key Encryption)**.** *An blind public key encryption scheme* $(\mathsf{Params}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *which is IND-CPA secure and satisfies the following additional security property: the function* $\mathsf{Enc}(\mathsf{pp}, \mathsf{pk}, m; r)$ *can be expressed as a concatenation* $E_1(\mathsf{pp}; r)||E_2(\mathsf{pp}, \mathsf{pk}, m; r)$ *such that the distribution* $\{\mathsf{pp} \leftarrow \mathsf{Params}(1^\lambda), (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(\mathsf{pp}), m \xleftarrow{\$} \{0,1\}^n : (\mathsf{pp}, \mathsf{pk}, \mathsf{sk}, \mathsf{Enc}(\mathsf{pp}, \mathsf{pk}, m))\}$ *is computationally indistinguishable from the distribution* $\{\mathsf{pp} \leftarrow \mathsf{Params}(1^\lambda), (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(\mathsf{pp}), m \xleftarrow{\$} \mathcal{M}, L = |E_2(\mathsf{pp}, \mathsf{pk}, m; r)|, \mathsf{subct}_2 \xleftarrow{\$} \{0,1\}^L : (\mathsf{pp}, \mathsf{pk}, \mathsf{sk}, E_1(\mathsf{pp}; r)||\mathsf{subct}_2)\}$. *That is, encryptions of random messages are pseudorandom (along with some function independent of the public key) even given the secret key.*

We note here that blind public key encryption can be constructed generically from blind batch encryption; indeed, blind batch encryption can be used to build a blind PKE scheme satisfying stronger security notions such as *leakage resilience* and *key-dependent message (KDM) security* (see Section 7).

## 5.2 The Construction

The construction of our weakly compact blind IBE scheme $\mathcal{WBIBE}$ uses three ingredients:

- A blind public-key encryption scheme

  $$\mathcal{BPKE} = (\mathsf{BPKE.Params}, \mathsf{BPKE.Gen}, \mathsf{BPKE.Enc}, \mathsf{BPKE.Dec})$$

  where the encryption algorithm can be decomposed into $\mathsf{BPKE}.E_1$ and $\mathsf{BPKE}.E_2$ as in Definition 5.4;
- A blind garbling scheme

  $$\mathcal{BGBL} = (\mathsf{BGC.Garble}, \mathsf{BGC.Eval}, \mathsf{BGC.Sim});\ \text{and}$$

- A blind batch encryption scheme

  $$\mathcal{BBENC} = (\mathsf{Batch.Setup}, \mathsf{Batch.Gen}, \mathsf{Batch.Enc}, \mathsf{Batch.Dec})$$

  where the encryption algorithm can be decomposed into $\mathsf{Batch}.E_1$ and $\mathsf{Batch}.E_2$ as in Definition 3.7. Moreover, we assume that $\mathcal{BBENC}$ is fully succinct.

[3]This is only a technical difference, since we only consider weakly compact IBE schemes with $T = \mathrm{poly}(\lambda)$

The construction works as follows.

1. $\mathsf{WBIBE.Params}(1^T)$: Given a bound $T$ on the number of identities, the parameter generation algorithm $\mathsf{Params}$ first obtains blind public-key encryption parameters $\mathsf{bpke.pp} \leftarrow \mathsf{BPKE.Params}(1^\lambda)$. Letting $n$ be the length of the public keys generated by $\mathsf{BPKE.Gen}$, it then obtains a common reference string $\mathsf{batch.crs} \leftarrow \mathsf{Batch.Setup}(1^\lambda, 1^{nT})$. The output is

$$\mathsf{wbibe.pp} = (\mathsf{bpke.pp}, \mathsf{batch.crs})$$

2. $\mathsf{WBIBE.Setup}(\mathsf{wbibe.pp})$: On input the public parameters, the setup algorithm first obtains $T$ key pairs $(\mathsf{bpke.pk}_i, \mathsf{bpke.sk}_i) \leftarrow \mathsf{BPKE.Gen}(\mathsf{bpke.pp})$. Secondly, it compresses the sequence of $\mathcal{BPKE}$ public keys into a $\mathcal{BBENC}$ public key:

$$h \leftarrow \mathsf{Batch.Gen}(\mathsf{batch.crs}, (\mathsf{bpke.pk}_0, \mathsf{bpke.pk}_1, \ldots, \mathsf{bpke.pk}_{T-1})).$$

   The output is the pair $(\mathsf{wbibe.mpk}, \mathsf{wbibe.msk})$ where

$$\mathsf{wbibe.mpk} = h \quad \text{and}$$
$$\mathsf{wbibe.msk} = (\mathsf{bpke.pk}_0, \ldots, \mathsf{bpke.pk}_{T-1}, \mathsf{bpke.sk}_0, \ldots, \mathsf{bpke.sk}_{T-1})$$

3. $\mathsf{WBIBE.Keygen}(\mathsf{wbibe.pp}, \mathsf{wbibe.msk}, \mathsf{id})$: On input the public parameters, the master secret key and an identity $\mathsf{id} \in \{0, 1, \ldots, T-1\}$, the key generation algorithm outputs

$$\mathsf{wbibe.sk}_{\mathsf{id}} = (\mathsf{id}, \mathsf{bpke.pk}_0, \mathsf{bpke.pk}_1, \ldots, \mathsf{bpke.pk}_{T-1}, \mathsf{bpke.sk}_{\mathsf{id}}).$$

4. $\mathsf{WBIBE.Enc}(\mathsf{wbibe.pp}, \mathsf{wbibe.mpk}, \mathsf{id}, m)$: On input the public parameters, master public key, an identity $\mathsf{id}$ and a message $m$, the encryption algorithm does the following.

   First, sample a uniformly random string $r$ and compute

$$\mathsf{ct}_0 = \mathsf{BPKE}.E_1(\mathsf{bpke.pp}; r) \ .$$

   Secondly, let $C[\mathsf{bpke.pp}, m, r]$ be a circuit with blind public parameters $\mathsf{bpke.pp}$ (contained as part of $\mathsf{wbibe.pp}$), the message $m$ and the random string $r$ hardcoded. $C$ takes as input a blind public key and outputs the encryption of $m$ under the public key using randomness $r$. That is,

$$C[\mathsf{bpke.pp}, m, r](\mathsf{bpke.pk}) = \mathsf{BPKE}.E_2(\mathsf{bpke.pp}, \mathsf{bpke.pk}, m; r)$$

   Compute

$$\left(\widehat{C}, \overline{\mathsf{lab}}\right) \leftarrow \mathsf{BGC.Garble}(1^\lambda, 1^n, 1^\ell, C[\mathsf{bpke.pp}, m, r])$$

   where $\overline{\mathsf{lab}} \in (\{0,1\}^\lambda)^{n \times 2}$ and $\ell$ is defined to be the ouput length of $C$. Set $\mathsf{ct}_1 := \widehat{C}$.

   Finally, let $\mathbf{M} \in (\{0,1\}^\lambda)^{nT \times 2}$ be a uniformly random $nT$-by-2 matrix and then *redefine* $\mathbf{M}[\mathsf{id} \cdot n + j, b] = \overline{\mathsf{lab}}[j, b]$ for all $1 \le j \le n, b \in \{0,1\}$. Compute

$$(\mathsf{ct}_2, \mathsf{ct}_2') \leftarrow \mathsf{Batch.Enc}(\mathsf{batch.crs}, h, \mathbf{M}) \ .$$

   Output the ciphertext $\mathsf{wbibe.ct} = (\mathsf{ct}_0, \mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_2')$.

5. $\mathsf{WBIBE.Dec}(\mathsf{wbibe.pp}, \mathsf{wbibe.sk}, \mathsf{wbibe.ct})$: On input the public parameters, a secret key and a ciphertext, the decryption algorithm parses the secret key as $\mathsf{wbibe.sk} = (\mathsf{id}, \mathsf{bpke.pk}_0, \ldots, \mathsf{bpke.pk}_{T-1}, \mathsf{bpke.sk}_{\mathsf{id}})$, and parses the ciphertext as $\mathsf{wbibe.ct} = (\mathsf{ct}_0, \mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}'_2)$. It then does three things.

   First, it computes

$$\mathbf{m} \leftarrow \mathsf{Batch.Dec}(\mathsf{batch.crs}, (\mathsf{bpke.pk}_0, \mathsf{bpke.pk}_1, \ldots, \mathsf{bpke.pk}_{T-1}), \mathsf{ct}_2 || \mathsf{ct}'_2),$$

   Secondly, it defines $\widehat{L} = (L_j)_{j \in [n]} \in (\{0,1\}^\lambda)^n$ by $L_j = \mathbf{m}[\mathsf{id} \cdot n + j]$ and computes $\mathsf{ct}'_0 \leftarrow \mathsf{BGC.Eval}(\mathsf{ct}_1, \widehat{L})$. Finally, it computes and outputs

$$m \leftarrow \mathsf{BPKE.Dec}(\mathsf{bpke.pp}, \mathsf{bpke.sk}_{\mathsf{id}}, \mathsf{ct}_0 || \mathsf{ct}'_0) \ .$$

We now show that this scheme is a weakly compact blind IBE scheme.

**Theorem 5.2.** *Suppose that $\mathcal{BPKE}$ is a blind public-key encryption scheme, $\mathcal{BBENC}$ is a blind batched encryption scheme, and $\mathcal{BGBL}$ is a blind garbling scheme. Then, $\mathcal{WBIBE}$ is a weakly compact blind IBE scheme.*

We first show in Section 5.3 that the scheme is correct and weakly compact. The proof of IND-ID-CPA security is in Section 5.4 and the proof of blindness is in Section 5.5.

### 5.3 Correctness and Compactness

Correctness of $\mathcal{WBIBE}$ is shown by the following sequence of steps. First, we note that $\mathsf{WBIBE.Dec}$ computes a vector $\mathbf{m}$ and labels $L_j$ which, by the correctness of the batch encryption algorithm and the definition of $\mathbf{M}$, satisfies

$$L_j := \mathbf{m}[\mathsf{id} \cdot n + j] = \overline{\mathsf{lab}}[j, \mathsf{bpke.pk}_{\mathsf{id}}[j]]$$

That is, the vector $\widehat{L}$ contains precisely the garbled circuit labels corresponding to the input $\mathsf{bpke.pk}_{\mathsf{id}}$. Next, the second step runs $\mathsf{BGC.Eval}(\widehat{C}, \widehat{L})$ and computes $\mathsf{ct}'_0 := \mathsf{BPKE}.E_2(\mathsf{bpke.pp}, \mathsf{bpke.pk}_{\mathsf{id}}, m; r)$ by the correctness of $\mathcal{BGBL}$. Since $\mathsf{ct}_0$ is by definition $\mathsf{BPKE}.E_1(\mathsf{bpke.pp}; r)$, their concatenation $\mathsf{ct}_0 || \mathsf{ct}'_0$ is a blind public-key encryption of the message $m$ under the public key $\mathsf{bpke.pk}_{\mathsf{id}}$. Thus, the final step decrypts this using $\mathsf{bpke.sk}_{\mathsf{id}}$ and recovers the message $m$.

Weak compactness follows from the fact that the master public key of $\mathcal{WBIBE}$ is simply the output $h$ of $\mathsf{Batch.Gen}$ and therefore has length $\mathrm{poly}(\lambda)$ (independent of $T$) by the succinctness of $\mathcal{BBENC}$. (Recall that weak compactness refers to the size of the master public key alone, and not the other objects such as the secret key or the ciphertext).

The blind decomposition of the encryption algorithm $\mathsf{WBIBE.Enc}$ is defined as follows. $\mathsf{WBIBE}.E_1$ consists of $\mathsf{ct}_0$ as well as $\mathsf{ct}_2 = \mathsf{Batch}.E_1(\mathsf{batch.crs}; R)$, and $\mathsf{WBIBE}.E_2$ consists of $\mathsf{ct}_1$ as well as $\mathsf{ct}'_2 = \mathsf{Batch}.E_2(\mathsf{batch.crs}, h, \mathbf{M}; R)$, for a uniformly random string $R$.

### 5.4 Semantic Security

We first prove IND-ID-CPA security of our scheme. In fact, this part of the proof only requires that $\mathcal{BPKE}$ is a semantically secure PKE scheme, $\mathcal{BBENC}$ is a secure batch encryption scheme,

and $\mathcal{BGBL}$ is a secure garbling scheme (i.e. no blindness is required).[4] As a corollary, we therefore obtain a construction of weakly compact IBE from batch encryption.

Suppose that $\mathcal{A}$ is an efficient adversary playing the IND-ID-CPA security game (as in Definition 2.1), and let $H_0$ denote the probability space underlying the IND-ID-CPA security game. We now define the hybrids $H_1, H_2$, and $H_2'$, which denote probability spaces underlying altered security games in which the challenge ciphertext (computed on challenge $(\mathsf{id}, m_\beta)$) is modified in various ways.

- In hybrid $H_1$, the pair $(\widehat{C}, \overline{\mathsf{lab}}) \leftarrow \mathsf{BGC.Garble}(1^\lambda, 1^n, 1^\ell, C[\mathsf{bpke.pp}, m_\beta, r])$ is computed as originally, but after choosing a uniformly random matrix $\mathbf{M} \in (\{0,1\}^\lambda)^{nT \times 2}$ we instead redefine $\mathbf{M}[\mathsf{id} \cdot n + j, \mathsf{bpke.pk_{id}}[j]] = \mathsf{lab}_{j, \mathsf{bpke.pk_{id}}[j]}$ for all $j \in [n]$ and output $\mathsf{wbibe.ct} = (\mathsf{ct}_0, \mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_2')$ for $\mathsf{ct}_0 = \mathsf{BPKE}.E_1(\mathsf{bpke.pp}, r)$, $\mathsf{ct}_1 = \widehat{C}$, and $(\mathsf{ct}_2, \mathsf{ct}_2') = \mathsf{Batch.Enc}(\mathsf{batch.crs}, h, \mathbf{M})$. That is, we now ignore the labels $\mathsf{lab}_{j, 1-\mathsf{bpke.pk_{id}}[j]}$
- In hybrid $H_2$, the challenge ciphertext is computed in the following steps. First, compute a $\mathcal{BPKE}$ ciphertext $\mathsf{ct}_0' = \mathsf{BPKE}.E_2(\mathsf{bpke.pp}, \mathsf{bpke.pk_{id}}, m_\beta; r)$ and compute a *simulated* garbled circuit $(\widetilde{C}, \widetilde{L}) \leftarrow \mathsf{BGC.Sim}(1^\lambda, 1^{|C[\mathsf{bpke.pp}, m_\beta, r]|}, 1^n, \mathsf{ct}_0')$. Then, choose a random matrix $\mathbf{M}$ as before, redefine $\mathbf{M}[\mathsf{id} \cdot n + j, \mathsf{bpke.pk_{id}}[j]] = \widetilde{L}_j$ for all $j \in [n]$ and output $\mathsf{wbibe.ct} = (\mathsf{ct}_0, \mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_2')$ for $\mathsf{ct}_0 = \mathsf{BPKE}.E_1(\mathsf{bpke.pp}; r)$, $\mathsf{ct}_1 = \widetilde{C}$, and $(\mathsf{ct}_2, \mathsf{ct}_2') = \mathsf{Batch.Enc}(\mathsf{batch.crs}, h, \mathbf{M})$.
- In hybrid $H_2'$, we replace the challenge message $m_\beta$ with a uniformly random message $m$ (and keep the rest as in $H_2$).

Clearly $\Pr_{H_2'}[\mathcal{A} \text{ wins}] = \frac{1}{2}$, as the challenge ciphertext received by the adversary is independent of the bit $b$. Thus, it suffices to prove that $\mathcal{A}$'s views in $H_0, H_1, H_2$, and $H_2'$ are computationally indistinguishable, which we denote by the shorthand $H_0 \approx_c H_1 \approx_c H_2 \approx_c H_2'$.

**Claim 5.3.** $H_0 \approx_c H_1$.

*Proof.* This follows from the security of $\mathcal{BBENC}$. More formally, consider the following adversary $\mathcal{A}'$ which plays the batch encryption security game: it obtains $\mathsf{bpke.pp} \leftarrow \mathsf{BPKE.Params}(1^\lambda)$, calls $\mathsf{BPKE.Gen}(\mathsf{bpke.pp})$ $T$ times to obtain $(\mathsf{bpke.pk}_i, \mathsf{bpke.sk}_i)$ as in the $\mathsf{WBIBE.Setup}$ algorithm (without knowing the batch encryption CRS), and chooses a batch encryption secret key $x = \mathsf{bpke.pk}_0 || \mathsf{bpke.pk}_1 || \ldots || \mathsf{bpke.pk}_{T-1}$. Then, after obtainining $\mathsf{batch.crs}$, $\mathcal{A}'$ computes $h = \mathsf{Batch.Gen}(\mathsf{batch.crs}, x)$ and runs the algorithm $\mathcal{A}$ with public parameters $\mathsf{wbibe.pp} = \mathsf{bpke.pp} || \mathsf{batch.crs}$ and keys $(\mathsf{wbibe.mpk}, \mathsf{wbibe.msk}) = (h, \mathsf{bpke.pk}_0 || \ldots || \mathsf{bpke.pk}_{T-1} || \mathsf{bpke.sk}_0 || \ldots || \mathsf{bpke.sk}_{T-1})$.

$\mathcal{A}'$ can clearly answer $\mathcal{A}$'s secret key queries because $\mathcal{A}'$ has the entire master secret key $\mathsf{wbibe.msk}$. The main detail of this $\mathcal{A}$-simulation is how $\mathcal{A}'$ computes a challenge ciphertext for $\mathcal{A}$. When $\mathcal{A}$ issues a challenge $(\mathsf{id}, m_0, m_1)$, $\mathcal{A}'$ chooses a random $\beta \xleftarrow{\$} \{0,1\}$, computes $(\widehat{C}, \overline{\mathsf{lab}}) \leftarrow \mathsf{BGC.Garble}(1^\lambda, 1^n, 1^\ell, C[\mathsf{bpke.pp}, m_\beta, r])$, and picks a uniformly random matrix $\mathbf{M} \in (\{0,1\}^\lambda)^{nT \times 2}$. Then, it issues $(\mathbf{M}^{(0)}, \mathbf{M}^{(1)})$ as its batch encryption challenge, where $\mathbf{M}^{(0)}$ is constructed from $(\mathbf{M}, \overline{\mathsf{lab}})$ as in $H_0$, and $\mathbf{M}^{(1)}$ is constructed from $(\mathbf{M}, \overline{\mathsf{lab}})$ as in $H_1$. Note that $\mathbf{M}^{(0)}_{j, x_j} = \mathbf{M}^{(1)}_{j, x_j}$ for all $j \in [nT]$, where again $x = \mathsf{bpke.pk}_0 || \ldots || \mathsf{bpke.pk}_{T-1}$. Upon receiving its challenge ciphertext $\mathsf{batch.ct}$, $\mathcal{A}'$ sends $\mathsf{ct}_0 || \mathsf{ct}_1 || \mathsf{batch.ct}$ to $\mathcal{A}$, where $\mathsf{ct}_0 = \mathsf{BPKE}.E_1(\mathsf{bpke.pp}; r)$ and $\mathsf{ct}_1 = \widehat{C}$.

[4] For non-blind schemes, we use the convention that a decomposition $\mathsf{Enc} = E_1 || E_2$ is by default the trivial decomposition $E_1 \equiv 0$, $E_2 = \mathsf{Enc}$.

After running its $\mathcal{A}$-simulation, $\mathcal{A}'$ obtains a bit $\beta'$ and outputs 1 if and only if $\beta' = \beta$. By construction, when $\mathcal{A}'$ receives a batch encryption of $\mathbf{M}^{(0)}$ its output distribution is exactly drawn according to $H_0$, while when $\mathcal{A}'$ receives a batch encryption of $\mathbf{M}^{(1)}$ its output distribution is exactly drawn according to $H_1$. Thus, by the security $\mathcal{BBENC}$, we conclude that $H_0 \approx_c H_1$, as desired. □

**Claim 5.4.** $H_1 \approx_c H_2$.

*Proof.* This follows directly from the simulation security of $\mathcal{BGBL}$. Namely, an adversary $\mathcal{A}$ which distinguishes between $H_1$ and $H_2$ can be used to break the simulation security of $\mathcal{BGBL}$ in the following way: run the $H_1$-security game in its entirety (choosing all of the $\mathcal{WBIBE}$ parameters) until $\mathcal{A}$ returns a challenge $(\mathsf{id}, m_0, m_1)$. Then, choose a random $\beta \xleftarrow{\$} \{0,1\}$, build the circuit $C = C[\mathsf{bpke.pp}, m_\beta, r]$, and query $(C, \mathsf{bpke.pk_{id}})$ to the $\mathcal{BGBL}$ security game. Upon obtaining $(\widehat{C}, \widehat{L})$ (either real or simulated), complete the construction of the challenge ciphertext $\mathsf{wbibe.ct}$ according to $H_1/H_2$ (the steps are the same), and output 1 if and only if $\mathcal{A}$ (given $\mathsf{wbibe.ct}$) returns $\beta' = \beta$. Distinguishing between $H_1$ and $H_2$ then corresponds exactly to distinguishing between a real pair $(\widehat{C}, \widehat{L})$ and a simulated pair $(\widetilde{C}, \widetilde{L})$, where the key detail is that only labels $L_{j,\mathsf{bpke.pk_{id}}[j]}$ are used in the remainder of the experiment, so we conclude that $H_1 \approx_c H_2$ by the simulation secruity of $\mathcal{BGBL}$.

□

**Claim 5.5.** $H_2 \approx_c H_2'$.

*Proof.* This follows from the semantic security of $\mathcal{BPKE}$. More formally, an adversary $\mathcal{A}'$ given public parameters $\mathsf{bpke.pp}$ and a public key $\mathsf{bpke.pk}$ but not the corresponding secret key $\mathsf{bpke.sk}$ can simulate $\mathcal{A}$ playing the $H_2/H_2'$ security games, respectively, in the following way. First, $\mathcal{A}'$ chooses an identity $\mathsf{id}^*$ at random, sets $\mathsf{bpke.pk_{id^*}} = \mathsf{bpke.pk}$, and chooses $(\mathsf{bpke.pk_{id}}, \mathsf{bpke.sk_{id}}) \leftarrow \mathsf{BPKE.Gen}(\mathsf{bpke.pp})$ (for all $\mathsf{id} \neq \mathsf{id}^*$) and $\mathsf{batch.crs} \leftarrow \mathsf{Batch.Setup}(1^\lambda, 1^{n \cdot T})$ itself. $\mathcal{A}'$ then runs $\mathcal{A}$; by construction, $\mathcal{A}'$ can answer any secret key query for $\mathsf{id} \neq \mathsf{id}^*$. If $\mathcal{A}'$ ever receives an $\mathsf{id}^*$ query, it returns a uniformly random bit as output for the entire $\mathcal{BPKE}$ security game. $\mathcal{A}$ will eventually issue a challenge $(\mathsf{id}, m_0, m_1)$; if $\mathsf{id} \neq \mathsf{id}^*$, $\mathcal{A}'$ again returns a uniformly random bit for the $\mathcal{BPKE}$ security game. Otherwise, $\mathcal{A}'$ issues a challenge ciphertext to $\mathcal{A}$ by appropriately choosing challenge messages (one equal to $m_\beta$ for random $\beta$, and the other a uniformly random message $m$) for its $\mathcal{BPKE}$ challenger, setting $(\mathsf{ct}_0, \mathsf{ct}_0') = (\mathsf{bpke.subct}_1, \mathsf{bpke.subct}_2)$ (where $\mathcal{A}'$'s received challenge ciphertext is $(\mathsf{bpke.subct}_1, \mathsf{bpke.subct}_2)$), and then computing the resulting challenge ciphertext for $\mathcal{A}$ as in $H_2/H_2'$ (the steps are the same). By outputting 1 if and only if its $\mathcal{A}$-simulation wins the hybrid security game (i.e. $\beta' = \beta$), $\mathcal{A}'$ distinguishes which $\mathcal{BPKE}$ challenge message was encrypted if and only if $\mathcal{A}$ distinguishes between $H_2$ and $H_2'$ (losing a factor of $\frac{1}{T}$ in the advantage because we conditioned on the event $\mathsf{id} = \mathsf{id}^*$, which is fine since $T$ is polynomial in $\lambda$). Thus, we see that $H_2 \approx_c H_2'$ by the semantic security of $\mathcal{BPKE}$. □

## 5.5 Blindness

Next, we show that $\mathcal{WBIBE}$ is blind assuming that $\mathcal{BBENC}, \mathcal{BPKE}$, and $\mathcal{BGBL}$ are all blind. Note that syntatically, $\mathsf{WBIBE}.E_1$ does not depend on $\mathsf{mpk}$, $\mathsf{id}$, or $m$, as desired.

Suppose that $\mathcal{A}$ is an efficient adversary playing the blind-ID security game (as in Definition 5.1), and let $H_0$ denote the probability space underlying the blind-ID security game. We define the hybrids $H_1$ and $H_2$ completely analogously as in Section 5.4 (although there is no $m_\beta$; only a

random message $m$ chosen by the challenger), so that $H_0 \approx_c H_1 \approx_c H_2$ (by the same proof). We additionally define hybrids $H_3, H_4, H_5$, which again denote probability spaces underlying altered security games in which the challenge ciphertext has been modified.

- In hybrid $H_3$, as compared to $H_2$, we replace the $\mathcal{BPKE}$ ciphertext $\mathsf{ct}'_0 = \mathsf{BPKE}.E_2(\mathsf{bpke.pp}, \mathsf{bpke.pk_{id}}, m; r)$ with a uniformly random ciphertext $\mathsf{ct}'_0$ of length $|\mathsf{ct}'_0| = |\mathsf{BPKE}.E_2(\mathsf{bpke.pp}, \mathsf{bpke.pk_{id}}, m; r)|$. The hybrid ($\mathcal{WBIBE}$) challenge ciphertext is then computed as in $H_2$.
- In hybrid $H_4$, as compared to $H_3$, we replace the pair $(\widetilde{C}, \widetilde{L}) \leftarrow \mathsf{BGC.Sim}(1^\lambda, 1^{|C[\mathsf{bpke.pp}, m, r]|}, 1^n, \mathsf{ct}'_0)$ by a uniformly random string $(\widetilde{C}, \widetilde{L})$ of length $|(\widetilde{C}, \widetilde{L})| = |\mathsf{BGC.Sim}(1^\lambda, 1^{|C[\mathsf{bpke.pp}, m, r]|}, 1^n, \mathsf{ct}'_0)|$.
- In hybrid $H_5$, as compared to $H_4$, we replace $(\mathsf{ct}_2, \mathsf{ct}'_2) = \mathsf{Batch.Enc}(\mathsf{batch.crs}, h, \mathbf{M}; R))$ with $(\mathsf{ct}_2, \mathsf{ct}'_2) = (\mathsf{Batch}.E_1(\mathsf{batch.crs}; R), R')$, where $R'$ is a uniformly random string of length $|\mathsf{Batch}.E_2(\mathsf{batch.crs}, h, \mathbf{M}; R)|$.

It is clear that $\Pr_{H_5}[\mathcal{A} \text{ wins}] = \frac{1}{2}$, as in $H_5$ the challenge ciphertext received by $\mathcal{A}$ no longer depends on the value of $\beta$ (in fact, its distribution matches the original (blind) challenge ciphertext distribution in $\beta = 1$ mode). Thus, it suffices to show that $H_2 \approx_c H_3 \approx_c H_4 \approx_c H_5$.

**Claim 5.6.** $H_2 \approx_c H_3$

*Proof.* This follows from the blindness of $\mathcal{BPKE}$ by a similar proof as that of Claim 5.5. More formally, an adversary $\mathcal{A}'$ given $(\mathsf{bpke.pp}, \mathsf{bpke.pk}, \mathsf{bpke.sk}, \mathsf{bpke.subct}_1 || \mathsf{bpke.subct}_2)$ as in the blind-ID security game can simulate $\mathcal{A}$ playing the $H_2/H_3$ security games, respectively, by choosing an identity $\mathsf{id}^*$ at random, setting $(\mathsf{bpke.pk_{id^*}}, \mathsf{bpke.sk_{id^*}}) = (\mathsf{bpke.pk}, \mathsf{bpke.sk})$, and choosing $(\mathsf{bpke.pk_{id}}, \mathsf{bpke.sk_{id}}) \leftarrow \mathsf{BPKE.Gen}(\mathsf{bpke.pp})$ (for all $\mathsf{id} \neq \mathsf{id}^*$) and $\mathsf{batch.crs} \leftarrow \mathsf{Batch.Setup}(1^\lambda, 1^{n \cdot T})$ itself. $\mathcal{A}'$ then runs $\mathcal{A}$; by construction, $\mathcal{A}'$ can answer any secret key query, as it has the entire master secret key $\mathsf{wbibe.msk}$. $\mathcal{A}$ will eventually issue a challenge identity $\mathsf{id}$; if $\mathsf{id} \neq \mathsf{id}^*$, $\mathcal{A}'$ returns a uniformly random bit as output for the entire $\mathcal{BPKE}$ security game. Otherwise, $\mathcal{A}'$ issues a challenge ciphertext to $\mathcal{A}$ by setting $(\mathsf{ct}_0, \mathsf{ct}'_0) = (\mathsf{bpke.subct}_1, \mathsf{bpke.subct}_2)$, and then computing the resulting challenge ciphertext for $\mathcal{A}$ as in $H_2/H_3$ (the steps are the same). By outputting 1 if and only if its $\mathcal{A}$-simulation wins the hybrid security game (i.e. $\beta' = \beta$), $\mathcal{A}'$ distinguishes a real $\mathcal{BPKE}$ challenge ciphertext from a real $\mathsf{bpke.subct}_1$ and random $\mathsf{bpke.subct}_2$ if and only if $\mathcal{A}$ distinguishes between $H_2$ and $H_3$ (losing a factor of $\frac{1}{T}$ in the advantage again, which is fine since $T$ is polynomial in $\lambda$). Thus, we see that $H_2 \approx_c H_3$ by the blindness of $\mathcal{BPKE}$. □

**Claim 5.7.** $H_3 \approx_c H_4$

*Proof.* This follows from the blindness of $\mathcal{BGBL}$, because the "circuit output" $\mathsf{ct}'_0$ plugged into $\mathsf{BGC.Sim}$ in hybrid $H_3$ is uniformly random and independent of all other parameters. Namely, an adversary $\mathcal{A}$ which distinguishes between $H_3$ and $H_4$ can be used to break the blindness property of $\mathcal{BGBL}$ in the following way: run the $H_3$-security game in its entirety (choosing all of the $\mathcal{WBIBE}$ parameters) until $\mathcal{A}$ returns a challenge identity $\mathsf{id}$. Then, generate $\mathsf{ct}_0$ as in $H_3$, build the circuit $C = C[\mathsf{bpke.pp}, m, r]$ (for, say, a random message $m$; we only care about the size of $C$), and query $(1^{|C|}, 1^n, 1^c)$ to the garbled circuit blindness security game (which only involves $\mathsf{BGC.Sim}$; here $c$ is the ouptput length of $C$). Upon obtaining $(\widetilde{C}, \widetilde{L})$ (either uniformly random or simulated from a random $\mathsf{ct}'_0$), complete the construction of the challenge ciphertext $\mathsf{wbibe.ct}$ according to $H_3/H_4$ (the steps are the same and, crucially, independent of $\mathsf{ct}'_0$). By outputting 1 if and only if $\mathcal{A}$ wins

its security game (i.e. $\beta' = \beta$), $\mathcal{A}$ distinguishing between $H_3$ and $H_4$ then corresponds exactly to this algorithm distinguishing between a simulated pair $(\widetilde{C}, \widetilde{L})$ and a random string $(\widetilde{C}, \widetilde{L})$, so we conclude that $H_3 \approx_c H_4$ by the blindness of $\mathcal{BGBL}$. □

**Claim 5.8.** $H_4 \approx_c H_5$

This follows from the blindness of $\mathcal{BBENC}$, because the matrix $\mathbf{M}$ batch encrypted in hybrid $H_4$ is a uniformly random matrix. The reduction proceeds analogously to the reduction in Claim 5.3 (although here $\mathcal{A}'$ does not have to choose a challenge message because we are simulating the blindness security game).

Having proved the above claims, we have completed the proof of Theorem 5.2.

# 6 Bootstrapping (Blind) IBE

Our bootstrapping theorem converting a weakly compact (blind) IBE scheme into a full-fledged (blind) IBE scheme follows the ideas of [DG17a, DG17b] and is essentially a way to achieve *domain extension* of the space of identities. The bootstrapping scheme is described in Section 6.1 and analyzed in Sections 6.2, 6.3, and 6.4. Recall that a high level overview was provided in the introduction (Section 1.2).

## 6.1 The Bootstrapping Theorem

Let $\mathcal{WBIBE}$ denote a weakly compact blind IBE scheme supporting $T = T(\lambda)$ identities with a master public key of size $S = S(\lambda)$ bits. By compactness, we may choose $T = \text{poly}(\lambda)$ large enough so that $S < T/4$. Additionally, let $\mathcal{BGBL} = (\mathsf{BGC.Garble}, \mathsf{BGC.Eval}, \mathsf{BGC.Sim})$ denote a blind garbling scheme. We construct a full-fledged blind IBE scheme $\mathcal{BIBE}$ as follows.

- $\mathsf{BIBE.Params}(1^\lambda, 1^n)$: On input the length $n$ of the identities supported by the system, the parameter generation algorithm generates parameter $\mathsf{wbibe.pp} \leftarrow \mathsf{WBIBE.Params}(1^\lambda, 1^T)$ and outputs $\mathsf{bibe.pp} = (1^n, \mathsf{wbibe.pp})$.

- $\mathsf{BIBE.Setup}(\mathsf{bibe.pp})$: On input the public parameters, the setup algorithm chooses a seed $s$ for a PRF family $f_s : \{0,1\}^{\leq n} \to \{0,1\}^r$ where $r$ is the number of random bits used by the $\mathsf{Setup}$ algorithm of $\mathcal{WBIBE}$. $\mathsf{BIBE.Setup}$ then obtains

$$(\mathsf{wbibe.mpk}^{(\epsilon)}, \mathsf{wbibe.msk}^{(\epsilon)}) \leftarrow \mathsf{WBIBE.Setup}(\mathsf{wbibe.pp}; f_s(\epsilon))$$

  where $\epsilon$ denotes the empty string. The output is

$$\mathsf{bibe.mpk} = \mathsf{wbibe.mpk}^{(\epsilon)} \quad \text{and} \quad \mathsf{bibe.msk} = s \ .$$

- $\mathsf{BIBE.Keygen}(\mathsf{bibe.pp}, \mathsf{bibe.msk}, \mathsf{id})$: On input the public parameters, the master secret key and an $n$-bit identity $\mathsf{id} = \mathsf{id}_1||\mathsf{id}_2||...\mathsf{id}_n$, the key generation algorithm does the following.

  First, for each prefix $\mathsf{id}[\leq i] = \mathsf{id}_1||\mathsf{id}_2||\ldots||\mathsf{id}_i \in \{0,1\}^i$, compute the master public key $\mathsf{wbibe.mpk}^{(\leq i)}$ and the master secret key $\mathsf{wbibe.msk}^{(\leq i)}$:

$$(\mathsf{wbibe.mpk}^{(\leq i)}, \mathsf{wbibe.msk}^{(\leq i)}) \leftarrow \mathsf{WBIBE.Setup}(\mathsf{wbibe.pp}; f_s(\mathsf{id}[\leq i])).$$

(By convention, $\mathsf{id}[\leq 0] = \epsilon$)

For each $0 \leq i \leq n-1$ and $j \in [S]$, define $\mathsf{id}'_{i,j} := \mathsf{id}_{i+1}||j||b_{i+1,j} \in \{0,1\} \times [S] \times \{0,1\}$, where $b_{i+1,j} := \mathsf{wbibe.mpk}^{(\leq i+1)}[j]$. Compute

$$\mathsf{sk}_{i,j} \leftarrow \mathsf{WBIBE.Keygen}(\mathsf{wbibe.pp}, \mathsf{wbibe.msk}^{(\leq i)}, \mathsf{id}'_{i,j}).$$

Finally, compute

$$\mathsf{sk}_{\mathrm{leaf}} \leftarrow \mathsf{WBIBE.Keygen}(\mathsf{wbibe.pp}, \mathsf{wbibe.msk}^{(\leq n)}, \mathsf{id}_{\mathrm{null}}),$$

where $\mathsf{id}_{\mathrm{null}} = 0^T$ is a default identity, and output

$$\mathsf{bibe.sk}_{\mathsf{id}} = \left( \left(\mathsf{wbibe.mpk}^{(\leq i)}\right)_{0 \leq i \leq n}, \left(\mathsf{sk}_{i,j}\right)_{j \in [S], 0 \leq i \leq n-1}, \mathsf{sk}_{\mathrm{leaf}} \right).$$

- $\mathsf{BIBE.Enc}(\mathsf{bibe.pp}, \mathsf{bibe.mpk}, \mathsf{id}, m)$: On input the public parameters, the master public key, an $n$-bit identity $\mathsf{id}$, and a message $m$, the encryption algorithm does the following.

  Let $C[\mathsf{wbibe.pp}, \eta, \overline{\mathsf{lab}}, \overline{\mathbf{r}}]$ be a circuit that computes the function

  $$C[\mathsf{wbibe.pp}, \eta, \overline{\mathsf{lab}}, \overline{\mathbf{r}}]\,(\mathsf{wbibe.mpk}) = \left(\mathsf{WBIBE}.E_2(\mathsf{wbibe.pp}, \mathsf{wbibe.mpk}, \eta||j||b, \mathsf{lab}_{j,b}; r_{j,b})\right)_{j \in [S], b \in \{0,1\}}$$

  where $\overline{\mathbf{r}}$ is the collection of all $r_{j,b}$ and $\overline{\mathsf{lab}}$ is the collection of all $\mathsf{lab}_{j,b}$. Let $C'[\mathsf{wbibe.pp}, m, r]$ be a circuit that computes the function

  $$C'[\mathsf{wbibe.pp}, m, r](\mathsf{wbibe.mpk}) = \mathsf{WBIBE}.E_2(\mathsf{wbibe.pp}, \mathsf{wbibe.mpk}, \mathsf{id}_{\mathrm{null}}, m; r)\ .$$

  Choose random strings $r$, $\overline{\mathbf{r}}^{(1)}, \ldots, \overline{\mathbf{r}}^{(n)}$.

  Compute $(\widehat{C}_n, \overline{\mathsf{lab}}^{(n)}) \leftarrow \mathsf{BGC.Garble}\big(C'[\mathsf{wbibe.pp}, m, r]\big)$. For $i = n-1$ to 0, compute

  $$(\widehat{C}_i, \overline{\mathsf{lab}}^{(i)}) \leftarrow \mathsf{BGC.Garble}\big(C[\mathsf{wbibe.pp}, \mathsf{id}_{i+1}, \overline{\mathsf{lab}}^{(i+1)}, \overline{\mathbf{r}}^{(i+1)}]\big)$$

  Compute $\mathsf{ct}_{n+1} \leftarrow \mathsf{WBIBE}.E_1(\mathsf{wbibe.pp}; r)$, and for $i = 1$ to $n$, compute

  $$\mathsf{ct}_{i,j,b} \leftarrow \mathsf{WBIBE}.E_1(\mathsf{wbibe.pp}; r^{(i)}_{j,b}),$$

  and let $\mathsf{ct}_i := (\mathsf{ct}_{i,j,b})_{j,b}$.

  Output the following as the ciphertext:

  $$\mathsf{bibe.ct} = \left(\widehat{C}_0, \ldots, \widehat{C}_{n-1}, \widehat{C}_n, \mathsf{ct}_1, \ldots, \mathsf{ct}_n, \mathsf{ct}_{n+1}, \overline{\mathsf{lab}}^{(0)}[\mathsf{wbibe.mpk}^{(\epsilon)}]\right),$$

  where $\overline{\mathsf{lab}}^{(0)}[\mathsf{wbibe.mpk}^{(\epsilon)}]$ is short-hand for $(\mathsf{lab}^{(0)}_{j,b_{0,j}})_{j \in [S]}$.

- $\mathsf{BIBE.Dec}(\mathsf{bibe.pp}, \mathsf{bibe.sk}_{\mathsf{id}}, \mathsf{bibe.ct})$: On input the public parameters, an identity secret key and a ciphertext, the decryption algorithm does the following.

  Let $\widehat{L}^{(0)} := \left(\mathsf{lab}^{(0)}_{j,b_{0,j}}\right)_{j \in [S]}$. For $1 \leq i \leq n$, do the following steps one after the other.

– Compute $\mathsf{ct}'_i \leftarrow \mathsf{Eval}(\widehat{C}_{i-1}, \widehat{L}^{(i-1)})$ which itself consists of ciphertexts $\mathsf{ct}'_{i,j,b}$ for $j \in [S]$ and $b \in \{0,1\}$.

– Compute $L_j^{(i)} \leftarrow \mathsf{WBIBE.Dec}(\mathsf{wbibe.pp}, \mathsf{sk}_{i,j}, \mathsf{ct}_{i,j,b_{i,j}} || \mathsf{ct}'_{i,j,b_{i,j}})$ for all $j \in [S]$ and $b_{i,j} = \mathsf{wbibe.mpk}^{(\leq i)}[j]$. Let $\widehat{L}^{(i)}$ denote the collection of all $L_j^{(i)}$.

Finally, compute $\mathsf{ct}'_{n+1} \leftarrow \mathsf{Eval}(\widehat{C}_n, \widehat{L}^{(n)})$ and output

$$m' \leftarrow \mathsf{WBIBE.Dec}(\mathsf{wbibe.pp}, \mathsf{sk}_{\mathrm{leaf}}, \mathsf{ct}_{n+1} || \mathsf{ct}'_{n+1}) \ .$$

- The blind decomposition of $\mathsf{BIBE.Enc}$ is as follows: $\mathsf{BIBE}.E_1(\mathsf{bibe.pp}; \mathbf{R})$ is defined to be the collection $(\mathsf{ct}_1, \mathsf{ct}_2, \ldots, \mathsf{ct}_{n+1})$, while $\mathsf{BIBE}.E_2(\mathsf{bibe.pp}, \mathsf{bibe.mpk}, \mathsf{id}, m; \mathbf{R})$ is defined to be the collection $\left(\widehat{C}_0, ..., \widehat{C}_n, \mathsf{lab}^{(0)}[\mathsf{bibe.mpk}]\right)$.

**Theorem 6.1.** *Suppose that $\mathcal{WBIBE}$ is a weakly compact blind IBE scheme and that $\mathcal{BGBL}$ is a blind garbling scheme. Then, $\mathcal{BIBE}$ is a blind IBE scheme. Additionally, even without the blindness assumptions, $\mathcal{BIBE}$ is an IBE scheme.*

**Remark 6.1.** *In our discussion of the above construction and its proof of security, we will refer to a tree of depth $n$ in which each node of the tree is labelled by a string $v \in \{0,1\}^{\leq n}$ and is assigned $(\mathsf{wbibe.mpk}^{(v)}, \mathsf{wbibe.msk}^{(v)})$, a specific $\mathcal{WBIBE}$ scheme to be executed "locally at node $v$". In this language, an encryption $\mathsf{BIBE.Enc}(\mathsf{wbibe.pp}, \mathsf{wbibe.mpk}^{(\epsilon)}, \mathsf{id}, m)$ computes a garbled circuit $(G_i, \overline{\mathsf{lab}}^{(i)})$ for each node $\mathsf{id}[\leq i] = \mathsf{id}_1 || \ldots || \mathsf{id}_i$ corresponding to a prefix of $\mathsf{id}$ in the sense that an honest decryptor will end up evaluting $G_i$ using the input labels for $\mathsf{wbibe.mpk}^{(\leq i)} = \mathsf{wbibe.mpk}^{(\mathsf{id}[\leq i])}$. We will also maintain the notation $\mathsf{wbibe.mpk}^{(\leq i)} = (b_{i,j})_{j \in [S]}$.*

We first show in Section 6.2 that $\mathcal{BIBE}$ satisfies correctness of an IBE scheme. The proof of IND-ID-CPA security is in Section 6.3 and the proof of blindness is in Section 6.4.

## 6.2 Proof of Correctness

Correctness crucially uses the fact that $S \leq \frac{T}{4}$, so that $\mathcal{WBIBE}$ supports identities of the form $\mathsf{id}_i || j || b \in \{0,1\} \times [S] \times \{0,1\}$ (as $j$ here is referring to the $j$th bit of a master public key). Given this fact, we see by an inductive argument that if we are given $\mathsf{bibe.ct} \leftarrow \mathsf{BIBE.Enc}(\mathsf{bibe.pp}, \mathsf{bibe.mpk}, \mathsf{id}, m)$, then for each step $1 \leq i \leq n$ of the decryption process (when decrypting $\mathsf{bibe.ct}$),

$$\mathsf{ct}'_{i,j,b} = \mathsf{WBIBE}.E_2(\mathsf{wbibe.pp}, \mathsf{wbibe.mpk}^{(\leq i-1)}, \mathsf{id}_i || j || b, \mathsf{lab}_{j,b}^{(i)}; r_{j,b}^{(i)})$$

by the correctness of $\mathcal{BGBL}$, and so

$$L_j^{(i)} = \mathsf{lab}_{j,b_{i,j}}^{(i)}$$

for each $j$ by the correctness of $\mathcal{WBIBE}$. Thus, after step $n+1$, we also have that

$$\mathsf{ct}'_{n+1} = \mathsf{WBIBE}.E_2(\mathsf{wbibe.pp}, \mathsf{wbibe.mpk}^{(\leq n)}, \mathsf{id}_{\mathrm{null}}, m; r)$$

by the correctness of $\mathcal{BGBL}$, so that $m' = m$ as desired.

### 6.3 Semantic Security

We first prove IND-ID-CPA security of our scheme, which only requires that $\mathcal{WBIBE}$ is a secure (weakly compact) IBE scheme and $\mathcal{BGBL}$ is a secure garbling scheme (i.e. no blindness is required).

Suppose that $\mathcal{A}$ is an efficient adversary playing the IND-ID-CPA security game (as defined in Definition 2.1), let $Q = Q(n, \lambda)$ be a (polynomial) bound on the runtime of $\mathcal{A}$, and let $\tilde{H}_0$ denote the probability space underlying the IND-ID-CPA security game. We first define the hybrid security game $H_0 = H_{0,0}$, in which all invocations of the PRF $f_s(x)$ (made by the challenger) are replaced by the values $F(x)$ of a truly random function $F : \{0,1\}^{\leq n} \to \{0,1\}^r$. Moreover, we have that $\tilde{H}_0 \approx_c H_0$ (or more precisely that $\Pr_{\tilde{H}_0}[\mathcal{A} \text{ wins}] = \Pr_{H_0}[\mathcal{A} \text{ wins}] + \text{negl}(\lambda)$) by the security of the PRF family $\{f_s\}$. We now define the hybrids $H_{i,1}$ and $H_{i+1,0}$ for all $0 \leq i \leq n-1$, which denote probability spaces underlying altered security games in which the challenge ciphertext is modified in various ways.

- In hybrid $H_{i,1}$, as compared to $H_{i,0}$, we compute (for each $j \in [S], b \in \{0,1\}$)

$$\mathsf{ct}'_{i+1,j,b} = \mathsf{WBIBE}.E_2\left(\mathsf{wbibe.pp}, \mathsf{wbibe.mpk}^{(\leq i)}, \mathsf{id}_{i+1}||j||b, \mathsf{lab}_{j,b}^{(i+1)}; r_{j,b}^{(i+1)}\right)$$

  and replace the honestly generated garbled circuit $(\widehat{C}_i, \widehat{L}^{(i)})$ with a simulated garbled circuit

$$(\widetilde{C}_i, \widetilde{L}^{(i)}) \leftarrow \mathsf{BGC.Sim}\left(1^\lambda, 1^{|C_i|}, 1^S, \mathsf{ct}'_{i+1}\right)$$

  for $|C_i| = \left|C[\mathsf{wbibe.pp}, \mathsf{id}_{i+1}, \overline{\mathsf{lab}}^{(i+1)}, \overline{\mathbf{r}}^{(i+1)}]\right|$. By induction, we maintain the invariant that by hybrid $H_{i,0}$, only the labels $L_j^{(i)} = \mathsf{lab}_{j,b_{i,j}}^{(i)}$ are used in the modified challenge ciphertext.

- In hybrid $H_{i+1,0}$, as compared to $H_{i,1}$, we remove half of the labels $\overline{\mathsf{lab}}_{j,b}^{(i+1)}$; that is, we redefine

$$\mathsf{ct}'_{i+1,j,1-b_{i+1,j}} = \mathsf{WBIBE}.E_2\left(\mathsf{wbibe.pp}, \mathsf{wbibe.mpk}^{(\leq i)}, \mathsf{id}_{i+1}||j||(1-b_{i+1,j}), R_{j,1-b_{i+1,j}}^{(i+1)}; r_{j,1-b_{i+1,j}}^{(i+1)}\right),$$

  where each $R_{j,b}^{(i+1)} \in \{0,1\}^\lambda$ is a uniformly random string.

Finally, we define hybrid game $H_{n,1}$, in which compared to $H_{n,0}$, we first compute

$$\mathsf{ct}'_{n+1} = \mathsf{WBIBE}.E_2\left(\mathsf{wbibe.pp}, \mathsf{wbibe.mpk}^{(\leq n)}, \mathsf{id}_{\text{null}}, m_\beta; r\right)$$

and then replace the honestly generated garbled circuit $(\widehat{C}_n, \widehat{L}^{(n)})$ with a simulated garbled circuit

$$(\widetilde{C}_n, \widetilde{L}^{(n)}) \leftarrow \mathsf{BGC.Sim}\left(1^\lambda, 1^{|C_n = C'|}, 1^S, \mathsf{ct}'_{n+1}\right).$$

We also define hybrid game $\tilde{H} = H_{n+1,0}$, in which we replace $\mathsf{ct}_{n+1}||\mathsf{ct}'_{n+1}$ with an encryption $\mathsf{ct}_{n+1}||\mathsf{ct}'_{n+1} = \mathsf{WBIBE.Enc}\left(\mathsf{wbibe.pp}, \mathsf{wbibe.mpk}^{(\leq n)}, \mathsf{id}_{\text{null}}, m; r\right)$ of a uniformly random message $m \xleftarrow{\$} \mathcal{M}$.

Clearly $\Pr_{\tilde{H}}[\mathcal{A} \text{ wins}] = \frac{1}{2}$, as the challenge ciphertext received by the adversary is independent of the bit $\beta$. Thus, it suffices to prove that $\mathcal{A}$'s views in $H_{i,0}$ and $H_{i,1}$ are computationally indistinguishable (for each $i$) and that $\mathcal{A}$'s views in $H_{i,1}$ and $H_{i+1,0}$ are computationally indistinguishable (for each $i$).

**Claim 6.2.** *$H_{i,0} \approx_c H_{i,1}$ for all $0 \leq i \leq n-1$.*

This follows directly from the simulation security of $\mathcal{BGBL}$, crucially using the fact that only the labels $\mathsf{lab}^{(i)}_{j,b_{i,j}}$ are used in $H_{i,0}$. The proof is very similar to that of Claim 5.4.

**Claim 6.3.** *$H_{i,1} \approx_c H_{i+1,0}$ for all $0 \leq i \leq n-1$.*

*Proof.* This follows from the security of $\mathcal{WBIBE}$; in particular, the security of $\mathcal{WBIBE}$ with public key $\mathsf{wbibe.mpk}^{(\leq i)}$. For simplicity we will work with a "multiple challenge identities/messages" variant of IBE security (for $\mathcal{WBIBE}$) which follows from the definition of IBE security by a standard hybrid argument.

More formally, consider the following adversary $\mathcal{A}'$ which plays the $\mathcal{WBIBE}$ security game: it is given $\mathsf{wbibe.pp} \leftarrow \mathsf{WBIBE.Params}(1^\lambda, 1^S)$ along with $\mathsf{wbibe.mpk}^*$ as generated by $\mathsf{WBIBE.Setup}(\mathsf{wbibe.pp})$. $\mathcal{A}'$ then chooses a timestep $1 \leq q \leq Q$ at random and runs the algorithm $\mathcal{A}$, generating $(\mathsf{wbibe.mpk}, \mathsf{wbibe.msk})$ pairs in lazy fashion as they become necessary in $\mathcal{A}$'s secret key id queries (i.e. whenever a new node in the tree is traversed, generate a new $(\mathsf{wbibe.mpk}, \mathsf{wbibe.msk})$ pair for this node). However, when the $q$th unique node $v_q \in \{0,1\}^{\leq n}$ of the tree is first accessed, $\mathcal{A}'$ sets $\mathsf{wbibe.mpk}^{(v_q)} := \mathsf{wbibe.mpk}^*$ instead of generating a fresh $(\mathsf{wbibe.mpk}, \mathsf{wbibe.msk})$ pair; then, in order to answer secret key queries corresponding to paths including the node $v_q$, $\mathcal{A}'$ makes the appropriate secret key queries to its own $\mathsf{WBIBE.Keygen}$ oracle. Finally, when $\mathcal{A}$ sends a challenge $(\mathsf{id}, m_0, m_1)$, $\mathcal{A}'$ checks if $\mathsf{id}[\leq i] = v_q$; if not, $\mathcal{A}'$ immediately returns a uniformly random bit (as the result of the entire IBE security game). If $v_q = \mathsf{id}[\leq i]$, $\mathcal{A}'$ chooses a random $\beta \xleftarrow{\$} \{0,1\}$ and executes steps $n, n-1, ..., i+1$ of the $\mathsf{BIBE.Enc}(\mathsf{bibe.pp}, \mathsf{bibe.mpk}, \mathsf{id}, m_\beta)$ algorithm, yielding $(\widehat{C}_{i+1}, \overline{\mathsf{lab}}^{(i+1)})$. Then, $\mathcal{A}'$ issues the following challenge messages/identities to its $\mathsf{WBIBE}$ challenger: $\mathcal{A}'$ requests that it either be given encryptions of $\mathsf{lab}^{(i+1)}_{j,1-b_{i+1,j}}$ under identity $\mathsf{id}_{i+1}||j||(1-b_{i+1,j})$ for each $1 \leq j \leq S$, or that it be given encryptions of $R^{(i+1)}_{j,1-b_{i+1,j}}$ under identity $\mathsf{id}_{i+1}||j||(1-b_{i+1,j})$ for each $1 \leq j \leq S$. Upon receiving its challenge ciphertexts $(\mathsf{ct}_{i+1,j,1-b_{i+1,j}}, \mathsf{ct}'_{i+1,j,1-b_{i+1,j}})_{1 \leq j \leq S}$, $\mathcal{A}'$ completes the modified encryption algorithm using these ciphertexts (plugging them into the simulator, etc.) as described in hybrid $H_{i+1,0}$. It returns the resulting "ciphertext" to its $\mathcal{A}$-simulation.

After running its $\mathcal{A}$-simulation, $\mathcal{A}'$ obtains a bit $\beta'$ and outputs 1 if and only if $\beta' = \beta$. By construction, conditioned on the event that $\mathsf{id}[\leq i] = v_q$, we see that when $\mathcal{A}'$ receives challenge ciphertexts of the form $\mathsf{WBIBE.Enc}\left(\mathsf{wbibe.pp}, \mathsf{wbibe.mpk}^{(\leq i)}, \mathsf{id}_{i+1}||j||(1-b_{i+1,j}), \mathsf{lab}^{(i+1)}_{j,1-b_{i+1,j}}\right)$ (for all $1 \leq j \leq S$), $\mathcal{A}'$'s output matches the condition that $\beta' = \beta$ when $\mathcal{A}$ plays hybrid game $H_{i,1}$, while when $\mathcal{A}'$ receives challenge ciphertexts of the form $\mathsf{WBIBE.Enc}\left(\mathsf{wbibe.pp}, \mathsf{wbibe.mpk}^{(\leq i)}, \mathsf{id}_{i+1}||j||(1-b_{i+1,j}), R^{(i+1)}_{j,1-b_{i+1,j}}\right)$ (for all $1 \leq j \leq S$), $\mathcal{A}'$'s output distribution matches the condition that $\beta' = \beta$ when $\mathcal{A}$ plays hybrid game $H_{i+1,0}$. Moreover, no matter what queries $\mathcal{A}$ makes, $\mathcal{A}'$ *never* makes any invalid queries (that is, a query for identity $\mathsf{id}_{i+1}||j||(1-b_{i+1,j})$ in the $\mathcal{WBIBE}$ scheme at node $v_q$) in the above experiment, because $\mathcal{BIBE}$ secret keys never include these $\mathcal{WBIBE}$ secret keys. Thus, we conclude that by the IND-ID-CPA security of $\mathcal{WBIBE}$, hybrids $H_{i,1}$ and $H_{i+1,0}$ are computationally indistinguishable, incurring a loss in distinguishing advantage of $\frac{1}{Q}$ (which is fine because $Q$ is polynomial in $(n, \lambda)$). □

**Claim 6.4.** *$H_{n,0} \approx_c H_{n,1}$.*

This also follows from the simulation security of $\mathcal{BGBL}$, just as in Claim 6.2.

**Claim 6.5.** $H_{n,1} \approx_c H_{n+1,0}$.

This also follows from the security of $\mathcal{WBIBE}$. The reduction is very similar to that of Claim 6.3, although in this step, we make use of the fact that whenever $\mathcal{A}$ does not make an invalid secret key query (that is, the challenge identity $\mathsf{id}$) in hybrid $H_{n,1}$ (respectively, $H_{n+1,0}$), the constructed adversary $\mathcal{A}'$ will not make a secret key query for $\mathsf{id}_{\text{null}}$ in the $\mathcal{WBIBE}$ scheme at node $v = \mathsf{id}$ (if $\mathcal{A}$ does make an invalid query, then so will $\mathcal{A}'$).

This completes the proof of semantic security.

## 6.4 Blindness

Next, we show that our IBE scheme is blind assuming that the underlying weakly compact IBE scheme is blind. Note that syntatically, $\mathsf{BIBE}.E_1$ does not depend on $\mathsf{bibe.mpk}$, $\mathsf{id}$, or $m$, as desired.

Suppose that $\mathcal{A}$ is an efficient adversary playing the blind-ID security game (as defined in Definition 5.1), and let $\tilde{H}_0$ denote the probability space underlying the blind-ID security game. We again define the hybrids $H_0 = H_{0,0}$, $H_{i,1}$, and $H_{i+1,0}$ for $0 \le i \le n-1$ analogously to Section 6.3 (along with $H_{n,1}$), so that $\tilde{H}_0 \approx_c H_{0,0} \approx_c H_{0,1} \approx_c H_{1,0} \approx_c \ldots \approx_c H_{n,0} \approx_c H_{n,1}$ (by the same proof). Next, we additionally define hybrids $H_{i,2}$, $H_{i,3}$ for each $n \ge i \ge 0$:

- In hybrid $H_{n,2}$, as compared to $H_{n,1}$, we replace $\mathsf{ct}'_{n+1} = \mathsf{WBIBE}.E_2(\mathsf{wbibe.pp}, \mathsf{wbibe.mpk}^{(\le n)}, \mathsf{id}_{\text{null}}, m; r)$ with a uniformly random string $\mathsf{ct}'_{n+1}$ of length
$$|\mathsf{ct}'_{n+1}| = \left|\mathsf{WBIBE}.E_2(\mathsf{wbibe.pp}, \mathsf{wbibe.mpk}^{(\le n)}, \mathsf{id}_{\text{null}}, m; r)\right|.$$
- In hybrid $H_{i,3}$ for all $n \ge i \ge 0$, as compared to $H_{i,2}$, we replace the simulated pair $(\widetilde{C}_i, \widetilde{L}^{(i)}) \leftarrow \mathsf{BGC.Sim}\left(1^\lambda, 1^{|C_i|}, 1^S, \mathsf{ct}'_{i+1}\right)$ by a uniformly random string $\left(\widetilde{C}_i, \left(R^{(i)}_{j,b_{i,j}}\right)_{j\in[S]}\right)$ of the same length.
- In hybrid $H_{i,2}$ for all $n-1 \ge i \ge 0$, as compared to $H_{i+1,3}$, we replace (for each $b \in \{0,1\}, j \in [S]$) the ciphertexts $\mathsf{ct}'_{i+1,j,b} = \mathsf{WBIBE}.E_2\left(\mathsf{wbibe.pp}, \mathsf{wbibe.mpk}^{(\le i)}, \mathsf{id}_{i+1}||j||b, R^{(i+1)}_{j,b}; r^{(i+1)}_{j,b}\right)$ with uniformly random strings $\mathsf{ct}'_{i+1,j,b}$ of the same length.

It is clear that $\Pr_{H_{0,3}}[\mathcal{A} \text{ wins}] = \frac{1}{2}$, as in $H_{0,3}$ the challenge ciphertext recieved by $\mathcal{A}$ no longer depends on the value of $\beta$ (in fact, it is always the original challenge ciphertext in $\beta = 1$ mode). Thus, it suffices to show that all adjacent hybrids as defined above are computationally indistinguishable.

**Claim 6.6.** $H_{n,1} \approx_c H_{n,2}$

This follows from the blindness of $\mathcal{WBIBE}$, crucially using the fact that the message $m$ being encrypted (to create $(\mathsf{ct}_{n+1}, \mathsf{ct}'_{n+1})$) in $H_{n,1}$ is uniformly random. The proof is similar to that of Claim 6.8 below.

**Claim 6.7.** *$H_{i,2} \approx_c H_{i,3}$ for all $n \ge i \ge 0$.*

This follows from the blindness of $\mathcal{BGBL}$, crucially using the fact that $\mathsf{ct}'_{i+1}$, the input to $\mathsf{BGC.Sim}$ in hybrid $H_{i,2}$, is a uniformly random string independent of the rest of the $\mathcal{BIBE}$ experiment. The proof is very similar to that of Claim 5.7.

**Claim 6.8.** *$H_{i+1,3} \approx_c H_{i,2}$ for all $n-1 \geq i \geq 1$.*

*Proof.* This follows from the blindness of $\mathcal{WBIBE}$, crucially using the fact that the message matrix $\mathbf{R}^{(i+1)}$ being encrypted in Hybrid $H_{i+1,3}$ is a uniformly random matrix independent of the rest of the $\mathcal{BIBE}$ experiment. Again, we will reduce to a "multiple challenge identities" variant of blindness for $\mathcal{WBIBE}$ which follows immediately from Definitition 5.1 by a standard hybrid argument. The following proof is fairly similar to that of Claim 6.3.

Consider the following adversary $\mathcal{A}'$ which plays the $\mathcal{WBIBE}$ blindness game: it is given $\mathsf{wbibe.pp} \leftarrow \mathsf{WBIBE.Params}(1^\lambda, 1^S)$ along with $\mathsf{wbibe.mpk}^*$ as generated by $\mathsf{WBIBE.Setup}(\mathsf{wbibe.pp})$. $\mathcal{A}'$ then chooses a timestep $1 \leq q \leq Q$ at random and runs the algorithm $\mathcal{A}$, generating $(\mathsf{wbibe.mpk}, \mathsf{wbibe.msk})$ pairs in lazy fashion as they become necessary in $\mathcal{A}$'s secret key id queries (i.e. whenever a new node in the tree is traversed, generate a new $(\mathsf{wbibe.mpk}, \mathsf{wbibe.msk})$ pair for this node). However, when the $q$th unique node $v_q \in \{0,1\}^{\leq n}$ of the tree is first accessed, $\mathcal{A}'$ sets $\mathsf{wbibe.mpk}^{(v_q)} := \mathsf{wbibe.mpk}^*$ instead of generating a fresh $(\mathsf{wbibe.mpk}, \mathsf{wbibe.msk})$ pair; then, in order to answer secret key queries corresponding to paths including the node $v_q$, $\mathcal{A}'$ makes the appropriate secret key queries to its own $\mathsf{WBIBE.Keygen}$ oracle.

Finally, when $\mathcal{A}$ chooses a challenge identity $\mathsf{id}$, $\mathcal{A}'$ checks if $\mathsf{id}[\leq i] = v_q$; if not, $\mathcal{A}'$ immediately returns a uniformly random bit (as the result of the entire blind IBE security game). If $v_q = \mathsf{id}[\leq i]$, $\mathcal{A}'$ computes $\mathsf{ct}_{n+1} = \mathsf{WBIBE}.E_1(\mathsf{wbibe.pp}; r)$, $\mathsf{ct}_{\ell,j,b} = \mathsf{WBIBE}.E_1(\mathsf{wbibe.pp}; r^{(\ell)}_{j,b})$ for all $\ell \geq i+1$, and issues the following challenge identities to its $\mathsf{WBIBE}$ challenger: $\mathcal{A}'$ requests that it either be given encryptions $(\mathsf{wbibe.subct}_{1,j,b}, \mathsf{wbibe.subct}_{2,j,b})$ of (unknown to $\mathcal{A}'$) uniformly random strings $R^{(i+1)}_{j,b}$ under identity $\mathsf{id}_{i+1}||j||b$ for each $1 \leq j \leq S, b \in \{0,1\}$, or that it be given $(\mathsf{wbibe.subct}_{1,j,b}, \mathsf{wbibe.subct}_{2,j,b})$ for uniformly random $\mathsf{wbibe.subct}_{2,j,b}$ under identity $\mathsf{id}_{i+1}||j||b$ for each $(j,b)$. Upon receiving its challenge ciphertexts $(\mathsf{subct}_{1,j,b}, \mathsf{subct}_{2,j,b})_{j,b}$, $\mathcal{A}'$ completes the modified encryption algorithm, using $\mathsf{ct}_{i,j,b} = \mathsf{subct}_{1,j,b}$ and $\mathsf{ct}'_{i,j,b} = \mathsf{subct}_{2,j,b}$, as described in hybrid $H_{i+1,3}$. It then picks $\beta \xleftarrow{\$} \{0,1\}$ as in the $\mathcal{BIBE}$ blindness game and returns the resulting "ciphertext" (depending on the value of $\beta$) to its $\mathcal{A}$-simulation.

After running its $\mathcal{A}$-simulation, $\mathcal{A}'$ obtains a bit $\beta'$ and outputs 1 if and only if $\beta' = \beta$. By construction, conditioned on the event that $\mathsf{id}[\leq i] = v_q$, we see that when $\mathcal{A}'$ receives challenge ciphertexts of the form $\mathsf{WBIBE.Enc}\left(\mathsf{wbibe.pp}, \mathsf{wbibe.mpk}^{(\leq i)}, \mathsf{id}_{i+1}||j||b, R^{(i+1)}_{j,b}\right)$ (for all $1 \leq j \leq S, b \in \{0,1\}$), $\mathcal{A}'$'s output matches the condition that $\beta' = \beta$ when $\mathcal{A}$ plays hybrid game $H_{i+1,3}$, while when $\mathcal{A}'$ instead receives ciphertexts with each $\mathsf{wbibe.subct}_{2,j,b}$ replaced by a random string, $\mathcal{A}'$'s output distribution matches the condition that $\beta' = \beta$ when $\mathcal{A}$ plays hybrid game $H_{i,2}$. There is no issue of invalid identity queries, because all queries are valid in the blindness security games for $\mathcal{WBIBE}$ and $\mathcal{BIBE}$. Thus, we conclude that by the blindness of $\mathcal{WBIBE}$, hybrids $H_{i+1,3}$ and $H_{i,2}$ are computationally indistinguishable, incurring a loss in distinguishing advantage of $\frac{1}{Q}$ (which is fine because $Q$ is polynomial in $(n, \lambda)$). □

Having proved the above claims, we have completed the proof of Theorem 6.1.

# 7 Leakage Resilient and KDM Secure Public Key Encryption from Batch Encryption

In this section, we construct a public key encryption scheme $\mathcal{PKE}$ generically from a batch encryption scheme $\mathcal{BENC}$. Moreover, the public key encryption scheme $\mathcal{PKE}$ simultaneously satisfies

many special properties: it is secure against leakage of $1 - o(1)$ bits of the secret key $\mathsf{pke.sk}$, and it satisfies key-dependent message (KDM) security with respect to affine functions of the secret key. By prior works [BHHI10, App11], $\mathcal{PKE}$ can additionally be used to build a public key encryption scheme satisfying KDM security with respect to arbitrary, a priori bounded size functions of the secret key.

In Section 7.1 we recall the relevant definitions. The construction of the scheme $\mathcal{PKE}$ is in Section 7.2, and its proofs of security are in Sections 7.3 and 7.4. Finally, in Section 7.5, we show that if $\mathcal{BENC}$ is furthermore a *blind* batch encryption scheme, then $\mathcal{PKE}$ is a blind public key encryption scheme. This is used in Section 5 to prove that blind batch encryption (existentially) implies weakly compact blind IBE.

## 7.1 Definitions

A leakage resilient public key encryption scheme $\mathcal{PKE}$ satisfies a security property capturing the intuition that an adversary should not be able to break the semantic security of $\mathcal{PKE}$ even if given a bounded amount of information about the secret key $\mathsf{sk}$. The formal definition is stated below.

**Definition 7.1** (Leakage Resilient Public Key Encryption)**.** *A public key encryption scheme* $\mathcal{PKE} = (\mathsf{Params}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *satisfies* $L(\cdot)$ *leakage CPA-security if no polynomial time adversary* $\mathcal{A}$ *can win the following game with probability greater than* $\frac{1}{2} + \mathrm{negl}(\lambda)$*:*

1. $\mathsf{pp} \leftarrow \mathsf{Params}(1^\lambda)$
2. $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(\mathsf{pp})$
3. $(f, \mathsf{st}_1) \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{pk})$*, where* $f : \{0,1\}^{|\mathsf{sk}|} \to \{0,1\}^{L(|\mathsf{sk}|)}$*.*
4. $m_0, m_1, \mathsf{st}_2 \leftarrow \mathcal{A}(\mathsf{st}_1, f(\mathsf{sk}))$
5. $\beta \xleftarrow{\$} \{0,1\}$
6. $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{pk}, m_\beta)$
7. $\beta' \leftarrow \mathcal{A}(\mathsf{st}_2, \mathsf{ct})$
8. $\mathcal{A}$ *wins if* $\beta' = \beta$*.*

Meanwhile, a KDM secure public key encryption scheme satisfies the security property that an adversary should not be able to distinguish encryptions of 0 from encryptions of specified functions (chosen by the adversary) of the secret key. This strictly subsumes semantic security because the adversary can query the constant "1" function as part of the security game. The formal definition is below.

**Definition 7.2** (KDM Secure Public Key Encryption)**.** *A public key encryption scheme* $\mathcal{PKE} = (\mathsf{Params}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *satisfies (bounded function size) KDM security if no PPT adversary* $\mathcal{A}$ *can win the following game with probability greater than* $\frac{1}{2} + \mathrm{negl}(\lambda)$*.*

1. $\mathsf{pp} \leftarrow \mathsf{Params}(1^\lambda)$*.*
2. $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(\mathsf{pp}||1^S)$*, where* $S$ *is a bound on the size of circuits to be queried by* $\mathcal{A}$*.*

3. $\beta \xleftarrow{\$} \{0,1\}$.

4. $\mathsf{st}_0 = \mathsf{pk}$

5. *Iterate the following many times*

   (a) $(f_i, \mathsf{st}'_i) \leftarrow \mathcal{A}(\mathsf{st}_i)$, *where* $f_i$ *is a circuit of size* $\leq S$ *computing a function from* $\{0,1\}^{|\mathsf{sk}|} \to \{0,1\}$.

   (b) *If* $\beta = 0$, $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, 0)$. *If* $\beta = 1$, $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, f_i(\mathsf{sk}))$.

   (c) $\mathsf{st}_{i+1} \leftarrow \mathcal{A}(\mathsf{st}'_i, \mathsf{ct})$

6. $\beta' \leftarrow \mathcal{A}(\mathsf{st}_{final})$

7. $\mathcal{A}$ *wins if* $\beta' = \beta$.

*We say that* $\mathcal{PKE}$ *is KDM-secure with respect to affine functions of the secret key if no PPT adversary can win (with probability greater than* $\frac{1}{2} + \mathrm{negl}(n)$*) the security game above where the functions* $f_i$ *are required to be affine (over* $\mathbf{F}_2$*).*[5]

### 7.2 The Leakage-Resilient KDM-Secure Encryption Scheme

Suppose that $\mathcal{BENC} = (\mathsf{Batch.Setup}, \mathsf{Batch.Gen}, \mathsf{Batch.Enc}, \mathsf{Batch.Dec})$ is a (fully succinct) batch encryption scheme, which without loss of generality has public keys of size $|h| = \lambda$. We then construct, for any polynomial function $n = n(\lambda)$, a public key encryption scheme $\mathcal{PKE} = (\mathsf{PKE.Params}, \mathsf{PKE.Gen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$, defined as follows.

1. $\mathsf{PKE.Params}(1^\lambda)$ outputs $\mathsf{pp} = \mathsf{crs} \leftarrow \mathsf{Batch.Setup}(1^\lambda || 1^n)$.

2. $\mathsf{PKE.Gen}(\mathsf{pp})$ interprets $\mathsf{pp} = \mathsf{crs}$ as a $\mathcal{BENC}$ common reference string, chooses $x \xleftarrow{\$} \{0,1\}^n$ and computes $h = \mathsf{Batch.Gen}(\mathsf{pp}, x)$. It then outputs $(\mathsf{pk}, \mathsf{sk}) = (h, x)$.

3. $\mathsf{PKE.Enc}(\mathsf{pp}, \mathsf{pk}, m)$ first computes an XOR secret sharing $m = \bigoplus_{i=1}^n \mathbf{r}_i$, sets $\mathbf{M}_{i,b} = \mathbf{r}_i$ for each $b \in \{0,1\}$, and outputs $\mathsf{ct} = \mathsf{Batch.Enc}(\mathsf{pp}, \mathsf{pk}, \mathbf{M})$.

4. $\mathsf{PKE.Dec}(\mathsf{pp}, \mathsf{sk}, \mathsf{ct})$ computes $\mathbf{r}' = \mathsf{Batch.Dec}(\mathsf{pp}, x, \mathsf{ct})$ (for $x = \mathsf{sk}$) and outputs $m' = \oplus_i \mathbf{r}'_i$.

**Theorem 7.1.** *For any* $n = \mathrm{poly}(\lambda) > (1+\epsilon)\lambda$*, the scheme* $\mathcal{PKE}$ *is (1) a leakage-resilient public key encryption scheme with leakage rate* $(1 - \frac{2\lambda}{n})$ *and (2) KDM-secure with respect to affine functions of the secret key. In particular, if* $n = \omega(\lambda)$ *then this scheme achieves leakage resilience with leakage rate* $1 - o(1)$.

Correctness of the encryption scheme is clear; namely, for any vector $\mathbf{r}$ such that $\bigoplus_{i=1}^n \mathbf{r}_i = m$, if we set $\mathbf{M}_{i,b} = \mathbf{r}_i$ for each $(i,b)$, then by the correctness of $\mathcal{BENC}$, we have that

$$\mathsf{Batch.Dec}(\mathsf{pp}, \mathsf{sk} = x, \mathsf{Batch.Enc}(\mathsf{pp}, \mathsf{pk} = h, \mathbf{M})) = (\mathbf{M}_{i,x_i})_{i\in[n]} = (\mathbf{r}_i)_{i\in[n]},$$

so that $\mathsf{PKE.Dec}(\mathsf{pp}, \mathsf{sk}, \mathsf{PKE.Enc}(\mathsf{pp}, \mathsf{pk}, m)) = \bigoplus_{i=1}^n \mathbf{r}_i = m$, as desired.

[5] Note that in this case, the circuit size bound $S$ can be eliminated, and we will think of function queries as instead vector queries representing an affine function.

### 7.3 Proof of Leakage Resilience

We want to show that the scheme $\mathcal{PKE}$ above is $(1 - \frac{2\lambda}{n})$-leakage resilient.

Let $\mathcal{A}$ be an efficient adversary playing the $(1-\frac{2\lambda}{n})$ leakage-resilience game (as in Definition 7.1), and let $H_0$ denote the probability space underlying this leakage resilience game. Let us recall what a challenge ciphertext $\mathsf{ct} = \mathsf{PKE.Enc}(\mathsf{pp}, \mathsf{pk}, m_b)$ looks like: for $(\mathsf{pp}, \mathsf{pk}, \mathsf{sk}) = (\mathsf{crs}, h, x)$ as obtained from $\mathcal{BENC}$, $\mathsf{ct}$ is a batch encryption $\mathsf{Batch.Enc}(\mathsf{crs}, h, \mathbf{M})$ where $\mathbf{M}_{i,b} = \mathbf{r}_i$ (for all $i, b$) and $\bigoplus_i \mathbf{r}_i = m_b$ is an additive secret sharing of $m_b$. We now define the hybrids $H_1$ and $H_2$, which denote probability spaces underlying two modified security games in which the challenge ciphertext $\mathsf{ct}$ has been changed.

- In hybrid $H_1$, the challenger instead sends $\mathsf{ct} = \mathsf{Batch.Enc}(\mathsf{crs}, h, \mathbf{M})$, where $\mathbf{M}$ is a uniformly random matrix subject to the condition that $\bigoplus_i \mathbf{M}_{i,x_i} = m_\beta$; in other words, we have changed the entries $\mathbf{M}_{i,1-x_i}$ to be independent uniformly random bits, with $\mathbf{M}_{i,x_i} = \mathbf{r}_i$ as before.
- In hybrid $H_2$, the challenger instead sends $\mathsf{ct} = \mathsf{Batch.Enc}(k, h, \mathbf{M})$, where $\mathbf{M}$ is a uniformly random $n \times 2$ matrix.

Clearly $\Pr_{H_2}[\mathcal{A} \text{ wins}] = \frac{1}{2}$, as $\mathcal{A}$'s view in the game $H_2$ is independent of the bit $\beta$. Thus, it suffices to show that $\mathcal{A}$'s views in games $H_0, H_1$, and $H_2$ are computationally indistinguishable, which we denote by the shorthand $H_0 \approx_c H_1 \approx_c H_2$.

**Claim 7.2.** $H_0 \approx_c H_1$

*Proof.* This follows from the security of $\mathcal{BENC}$. Since the difference between $H_0$ and $H_1$ consists of changing $\mathbf{M}$ at entries $\mathbf{M}_{i,1-x_i}$, this should be the case even if the adversary $\mathcal{A}$ knows *the entire secret key* $\mathsf{sk} = x$. The reduction goes as follows: consider the following adversary $\mathcal{A}'$ playing the $\mathcal{BENC}$ batch encryption security game.

1. $\mathcal{A}'$ chooses a uniformly random $x \xleftarrow{\$} \{0,1\}^n$ and sends it to the challenger.
2. $\mathcal{A}'$ obtains the public parameters $\mathsf{crs} \leftarrow \mathsf{Batch.Setup}(1^n || 1^\lambda)$.
3. $\mathcal{A}'$ runs the adversary $\mathcal{A}$ with inputs $(\mathsf{pp}, \mathsf{pk}) = (\mathsf{crs}, h = \mathsf{Batch.Gen}(\mathsf{crs}, x))$.
4. On query function $f : \{0,1\}^n \to \{0,1\}^{L(n)}$, $\mathcal{A}'$ computes $f(x)$ and returns the value to $\mathcal{A}$.
5. On challenge messages $m_0, m_1$ provided by $\mathcal{A}$, $\mathcal{A}'$ chooses a random bit $\beta$, computes a secret sharing $m_\beta = \bigoplus_i \mathbf{r}_i$, and then chooses its own challenge matrices $\mathbf{M}^{(0)}, \mathbf{M}^{(1)}$ to be the matrices used in hybrids $H_0$ and $H_1$ respectively. Note that $\mathbf{M}^{(0)}$ and $\mathbf{M}^{(1)}$ differ only in $(i, 1 - x_i)$ entries.
6. Upon receiving a challenge ciphertext $\mathsf{ct}$ from its own challenger, $\mathcal{A}'$ runs $\mathcal{A}$ on challenge ciphertext $\mathsf{ct}$ and returns 1 if and only if $\beta' = \beta$ (that is, if and only if $\mathcal{A}$ wins its game).

By construction, the output distribution of $\mathcal{A}'$ when it is sent an encryption of $\mathbf{M}^{(0)}$ is drawn exactly according to $H_0$, while the output distribution of $\mathcal{A}'$ when it is sent an encryption of $\mathbf{M}^{(1)}$ is drawn exactly according to $H_1$. Therefore, by the security of $\mathcal{BENC}$, we conclude that $H_0 \approx_c H_1$, as desired. □

On the other hand, we will show that $H_1$ is *statistically* indistinguishable from $H_2$.

**Claim 7.3.** $H_1 \approx_s H_2$

*Proof.* This claim follows from the average min-entropy variant of the leftover hash lemma. For any fixed message $m$, consider the distribution of matrices $\mathbf{M}$ as used in hybrid $H_1$: the entries $\mathbf{M}_{i,b}$ are drawn independently and uniformly at random subject to the condition that $\bigoplus_{i=1}^{n} \mathbf{M}_{i,x_i} = m$. This is equivalent to the condition that

$$\bigoplus_i ((1-x_i)\mathbf{M}_{i,0} \oplus x_i \mathbf{M}_{i,1}) = m,$$

or that

$$\mathbf{M}_{n,0} = m \oplus \left(\bigoplus_{i<n} \mathbf{M}_{i,0}\right) \oplus \left(\bigoplus_i x_i(\mathbf{M}_{i,0} \oplus \mathbf{M}_{i,1})\right).$$

To be done, we simply have to show that the distribution of $\mathbf{M}_{n,0}$ is statistically close to uniform given $\mathbf{M}_{i,b}$ for all $(i,b) \neq (n,0)$ along with the triple $(\mathsf{crs}, h, f(x))$. This follows directly from the leftover hash lemma: the conditional distribution $x \mid \mathsf{crs}, h, f(x)$ has average min-entropy at least $n - \lambda - (n - 2\lambda)) = \lambda$, so by the leftover hash lemma, the distribution

$$\{\mathsf{crs} \leftarrow \mathsf{PKE.Params}(1^\lambda), x \xleftarrow{\$} \{0,1\}^n, (\mathbf{S}_i := \mathbf{M}_{i,0} \oplus \mathbf{M}_{i,1})_i \xleftarrow{\$} \{0,1\}^n : (\mathbf{S}, \bigoplus_i x_i \mathbf{S}_i, \mathsf{crs}, h, f(x))\}$$

is statistically indistinguishable from

$$\{\mathsf{crs} \leftarrow \mathsf{PKE.Params}(1^\lambda), x \xleftarrow{\$} \{0,1\}^n, (\mathbf{S}_i := \mathbf{M}_{i,0} \oplus \mathbf{M}_{i,1})_i \xleftarrow{\$} \{0,1\}^n, b \xleftarrow{\$} \{0,1\} : (\mathbf{S}, b, \mathsf{crs}, h, f(x))\},$$

so the matrix distribution $\mathbf{M}$ as used in $H_1$ is statistically indistinguishable from a uniformly random $\{0,1\}$-matrix, even given $(\mathsf{crs}, h, f(x))$. Thus, $H_1 \approx_s H_2$, proving the claim. □

Since we showed that $H_0 \approx_c H_1 \approx_s H_2$ and $\Pr_{H_2}[\mathcal{A} \text{ wins}] = \frac{1}{2}$, we conclude that $\Pr_{H_0}[\mathcal{A} \text{ wins}] < \frac{1}{2} + \mathrm{negl}(\lambda)$, completing the proof of leakage resilience.

## 7.4 Proof of KDM Security

Finally, we show that our scheme $\mathcal{PKE}$ is KDM-secure with respect to affine functions of the secret key.

First, note that the work in Section 7.3 already implies that our scheme is secure with respect to a *one query* KDM security game. To see this, suppose that $\mathsf{pp} \leftarrow \mathsf{PKE.Params}(1^\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.Gen}(\mathsf{pp})$, and an efficient adversary $\mathcal{A}(\mathsf{pp}, \mathsf{pk})$ produces a vector $a \in \{0,1\}^{n+1}$. An encryption of $\langle a, \mathsf{sk}||1\rangle = \langle a, x||1\rangle$ is a batch encryption $\mathsf{Batch.Enc}(\mathsf{pp}, \mathsf{pk}, \mathbf{M})$, where $\mathbf{M}_{i,b} = \mathbf{r}_i$ for each $(i,b)$ and $\bigoplus_i \mathbf{r}_i = \langle a, x||1\rangle$ is a linear secret sharing of $\langle a, x||1\rangle$. Just as in Claim 7.2, this is computationally indistinguishable (even if $\mathcal{A}$ fully knew the secret key $x$) from a batch encryption $\mathsf{Batch.Enc}(\mathsf{pp}, \mathsf{pk}, \mathbf{M})$ where $\mathbf{M}$ is a uniformly random matrix subject to the condition that

$$\bigoplus_i M_{i,x_i} = \langle a, x||1\rangle,$$

or equivalently,

$$\mathbf{M}_{n,0} = \langle a, x||1\rangle \oplus \left(\bigoplus_{i<n} \mathbf{M}_{i,0}\right) \oplus \left(\bigoplus_i x_i(\mathbf{M}_{i,0} \oplus \mathbf{M}_{i,1})\right).$$

Again, the rightmost term $\bigoplus_i x_i(\mathbf{M}_{i,0} \oplus \mathbf{M}_{i,1})$ is statistically close to a uniformly random bit independent of $((\mathbf{S}_i = \mathbf{M}_{i,0} \oplus \mathbf{M}_{i,1})_i, \mathsf{pp} = \mathsf{crs}, \mathsf{pk} = h)$ because $x \mid \mathsf{crs}, h$ has average min-entropy at least $n - \lambda$, so we conclude that $\mathbf{M}$ is statistically close to a uniformly random matrix.

In fact, the above argument does not rely on the function $f(x) = \langle a, x||1\rangle$ being affine; we could have used any function at all. However, we cannot proceed with a standard hybrid argument for the following reason: in the statistical step above, we rely on the secret key $x$ having high average min-entropy in the adversary's view. However, if the adversary has access to an additional $\mathrm{poly}(\lambda)$ encryptions of functions of the secret key, it may be the case that $x$ has *no entropy* in the adversary's view; it may be information-theoretically determined.

To circumvent this, we prove that encryptions of affine functions $\langle a, x||1\rangle$ can be *efficiently simulated* without knowledge of the secret key $x$, where simulation security holds even with respect to an adversary who knows $x$.

**Lemma 7.4.** *There is an efficient simulator* $\mathsf{PKE.Sim}(\mathsf{pp} = \mathsf{crs}, \mathsf{pk} = h, a \in \{0,1\}^{n+1})$ *such that for any* $a \in \{0,1\}^{n+1}$*, the distribution* $\{\mathsf{crs} \leftarrow \mathsf{PKE.Params}(1^\lambda), (h, x) \leftarrow \mathsf{PKE.Gen}(\mathsf{crs}) : (\mathsf{crs}, h, x, \mathsf{PKE.Enc}(\mathsf{crs}, h, x, \langle a, x||1\rangle))\}$ *is computationally indistinguishable from the distribution* $\{\mathsf{crs} \leftarrow \mathsf{PKE.Params}(1^\lambda), (h, x) \leftarrow \mathsf{PKE.Gen}(\mathsf{crs}) : (\mathsf{crs}, h, x, \mathsf{PKE.Sim}(\mathsf{pp}, h, a))\}$*.*

*Proof.* The simulator $\mathsf{PKE.Sim}(\mathsf{crs}, h, a)$ computes an additive secret sharing $\bigoplus_{i=1}^n \mathbf{r}_i = 0$ of 0, defines

$$\mathbf{M}_{i,b} = \mathbf{r}_i \oplus b a_i$$

for all $i < n$, and defines $\mathbf{M}_{n,b} = \mathbf{r}_n \oplus b a_n \oplus a_{n+1}$. It then outputs $\mathsf{ct} = \mathsf{Batch.Enc}(\mathsf{crs}, h, \mathbf{M})$.

It follows that $\mathsf{PKE.Sim}(\mathsf{crs}, h, a) \approx_c \mathsf{PKE.Enc}(\mathsf{crs}, h, \langle a, x||1\rangle)$ given $\mathsf{crs}, h$, and $x$ (for every $a$) by the security of $\mathcal{BENC}$. In particular, the marginal distributions of $(\mathbf{M}_{i,x_i})_{i\in[n]}$ are identical in the real and simulated cases (they are both an additive secret sharing of $\langle a, x||1\rangle$), so security follows by a completely analogous reduction as in Claim 7.2. □

We are now ready to fully prove KDM-security by combining our simulator $\mathsf{Sim}$ with the ideas of Section 7.3.

Let $\mathcal{A}$ be an efficient adversary playing the KDM-security game (as defined in Definition 7.2), and let $H_0$ denote the probability space underlying this security game. Additionally, let $Q = \mathrm{poly}(\lambda)$ be a bound on the number of encryption queries that $\mathcal{A}$ makes (at least with $1 - \mathrm{negl}(\lambda)$ probability). We now define the hybrids $\{H_{0,i}, 0 \le i \le Q\}, \{H_{1,i}, 0 \le i \le Q\}$, which denote various modified security games:

- In hybrid $H_{0,i}$ (for $0 \le i \le Q$), the challenger's responses to the adversary's first $i$ queries $\{a_j \in \{0,1\}^{n+1}, 1 \le j \le i\}$ have been changed. For these $i$ queries, the challenger does as follows: if $\beta = 0$, send an encryption $\mathsf{PKE.Enc}(\mathsf{pp}, \mathsf{pk}, 0)$ as before; if $\beta = 1$, send *a simulated ciphertext* $\mathsf{ct} = \mathsf{PKE.Sim}(\mathsf{pp}, \mathsf{pk}, a)$.
- In hybrid $H_{1,0} = H_{0,Q}$, all KDM ciphertexts $\mathsf{PKE.Enc}(\mathsf{pp}, \mathsf{pk}, \langle a_j, x||1\rangle)$ have been replaced by simulated ciphertexts $\mathsf{PKE.Sim}(\mathsf{pp}, \mathsf{pk}, a_j)$.

- In hybrid $H_{1,i}$ (for $0 \leq i \leq Q$), the challenger's responses to the adversary's first $i$ queries $\{a_j \in \{0,1\}^{n+1}\}$ encryption queries have been changed *as compared to* $H_{1,0}$. For these $i$ queries, the challenger always sends $\mathsf{ct} \leftarrow \mathsf{PKE.Enc}(\mathsf{pp}, \mathsf{pk}, 0)$, an encryption of 0.

Clearly $\Pr_{H_{1,Q}}[\mathcal{A} \text{ wins}] = \frac{1}{2}$, as $\mathcal{A}$'s view in this game is independent of the bit $\beta$. Thus, it suffices to show that $H_{0,i} \approx_c H_{0,i+1}$ for all $0 \leq i \leq Q-1$ and $H_{1,i} \approx_c H_{1,i+1}$ for all $0 \leq i \leq Q-1$.

**Claim 7.5.** *$H_{0,i} \approx_c H_{0,i+1}$ for all $0 \leq i \leq Q-1$*

*Proof.* This follows from the simulation security of $\mathsf{PKE.Sim}$ as proved in Lemma 7.4. That is, an adversary $\mathcal{A}'$ given $(\mathsf{crs}, h, x, a)$, and either a ciphertext $\mathsf{ct} = \mathsf{ct}_1 \leftarrow \mathsf{PKE.Enc}(\mathsf{crs}, h, \langle a, x||1\rangle)$ or a simulated ciphertext $\mathsf{ct} = \mathsf{ct}_2 \leftarrow \mathsf{PKE.Sim}(\mathsf{crs}, h, a)$ can run $\mathcal{A}$, answering $\mathcal{A}$'s first $i$ queries by generating simulated ciphertexts itself, answering $\mathcal{A}$'s $i+1$th query with $\mathsf{ct}$ (if $\beta = 1$ in $\mathcal{A}$'s security game), and answering $\mathcal{A}$'s later queries by generating KDM ciphertexts itself (since $\mathcal{A}'$ knows $x$). $\mathcal{A}'$ then returns 1 if and only if $\beta' = \beta$. It is clear that $\mathcal{A}'$'s output distribution follows $\mathcal{A}$'s win distribution in $H_{0,i}$ when $\mathcal{A}'$ is given $\mathsf{ct}_1$ while $\mathcal{A}'$'s output distribution follows $\mathcal{A}$'s win distribution in $H_{0,i+1}$ when $\mathcal{A}'$ is given $\mathsf{ct}_2$. By Lemma 7.4, we know that $\mathcal{A}'$ cannot distinguish between the cases $\mathsf{ct} = \mathsf{ct}_1$ and $\mathsf{ct} = \mathsf{ct}_2$, so we conclude that $H_{0,i} \approx_c H_{0,i+1}$, as desired. □

**Claim 7.6.** *$H_{1,i} \approx_c H_{1,i+1}$ for all $0 \leq i \leq Q-1$.*

*Proof.* This is equivalent to the *one query* KDM-security proved at the beginning of this subsection. More explicitly, we define one further hybrid $H'_{1,i}$ in which the response to $\mathcal{A}$'s $i+1$th query $a_{i+1}$ is changed from $\mathsf{Batch.Enc}(\mathsf{crs}, h, \mathbf{M})$ with honestly generated $\mathbf{M}_{i,b} = \mathbf{r}_i$ to $\mathsf{Batch.Enc}(\mathsf{crs}, h, \mathbf{M})$ with $\mathbf{M}$ a uniformly random matrix subject to $\bigoplus_j \mathbf{M}_{j,x_j} = \langle a_{i+1}, x||1\rangle$.

As before, we then show that $H_{1,i} \approx_c H'_{1,i} \approx_s H_{1,i+1}$. The fact that $H_{1,i} \approx_c H'_{1,i}$ follows from an analogous argument to Claim 7.2, while the fact that $H'_{1,i} \approx_s H_{1,i+1}$ follows from an analogous argument to Claim 7.3 because all auxiliary ciphertexts $\mathsf{PKE.Enc}(\mathsf{crs}, h, 0)$ and simulated ciphertexts $\mathsf{PKE.Sim}(\mathsf{crs}, h, a)$ are (randomized) functions of $(\mathsf{crs}, h)$ and do not depend on $x$. □

## 7.5 Blind PKE from Blind Batch Encryption

**Theorem 7.7.** *Suppose that the underlying batch encryption scheme $\mathcal{BENC}$ (which we will now rename to $\mathcal{BBENC}$) in Theorem 7.1 is* blind. *Then, again for any $n > (1+\epsilon)\lambda$, the public key encryption scheme $\mathcal{PKE}$ defined in Section 7.2 is a blind PKE scheme.*

*Proof.* We already showed in Theorem 7.1 that the scheme is a correct, CPA-secure PKE scheme, so all we have to show is blindness. The blind decomposition of our encryption scheme is as follows: we define $\mathsf{PKE}.E_1(\mathsf{crs}; r) := \mathsf{Batch}.E_1(\mathsf{crs}; r)$ and $\mathsf{PKE}.E_2(\mathsf{crs}, h, m; r) = \mathsf{Batch}.E_2(\mathsf{crs}, h, \mathbf{M}; r)$, where $\mathbf{M}_{i,b} = \mathbf{r}_i$ is as in the Section 7.2 construction (surpressing the additional randomness of choosing $\mathbf{r}$). To prove that this is a valid blind PKE scheme, consider the distribution

$$\{\mathsf{pp} = \mathsf{crs} \leftarrow \mathsf{PKE.Params}(1^\lambda), (h, x) \leftarrow \mathsf{PKE.Gen}(\mathsf{pp}), m \xleftarrow{\$} \{0,1\} : (\mathsf{crs}, h, x, \mathsf{PKE.Enc}(\mathsf{crs}, h, m; r))\}.$$

This distribution is computationally indistinguishable from the distribution

$$\{\mathsf{pp} = \mathsf{crs} \leftarrow \mathsf{PKE.Params}(1^\lambda), (h, x) \leftarrow \mathsf{PKE.Gen}(\mathsf{pp}), \mathbf{M} \xleftarrow{\$} \{0,1\}^{n\times 2} : (\mathsf{crs}, h, x, \mathsf{Batch}.E_1(\mathsf{crs}; r)||\mathsf{Batch}.E_2(\mathsf{crs}, h, \mathbf{M}; r))\}$$

by the semantic security of $\mathcal{BBENC}$ (see Claim 7.2; in particular, the indistinguishability result held even if the adversary knows $x$). But this distribution is in turn computationally indistinguishable from the distribution

$$\{\mathsf{pp} = \mathsf{crs} \leftarrow \mathsf{PKE.Params}(1^\lambda), (h, x) \leftarrow \mathsf{PKE.Gen}(\mathsf{pp}), \mathbf{M} \xleftarrow{\$} \{0,1\}^{n\times 2}, \\ \mathsf{ct} \xleftarrow{\$} \{0,1\}^{|\mathsf{Batch}.E_2(\mathsf{crs},h,\mathbf{M};r)|} : (\mathsf{crs}, h, x, \mathsf{Batch}.E_1(\mathsf{crs}; r)||\mathsf{ct})\}$$

by the blindness of $\mathcal{BBENC}$. This proves the desired blindness property of our PKE scheme. □

## References


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

[ABC+08] Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi, *Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions*, J. Cryptology **21** (2008), no. 3, 350–391.

[ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai, *Fast cryptographic primitives and circular-secure encryption based on hard learning problems*, Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings (Shai Halevi, ed.), Lecture Notes in Computer Science, vol. 5677, Springer, 2009, pp. 595–618.

[AGV09] Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan, *Simultaneous hardcore bits and cryptography against memory attacks*, Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings, 2009, pp. 474–495.

[Ale11] Michael Alekhnovich, *More on average case vs approximation complexity*, Computational Complexity **20** (2011), no. 4, 755–786.

[App11] Benny Applebaum, *Key-dependent message security: Generic amplification and completeness*, Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings (Kenneth G. Paterson, ed.), Lecture Notes in Computer Science, vol. 6632, Springer, 2011, pp. 527–546.

[BBR99] Eli Biham, Dan Boneh, and Omer Reingold, *Breaking generalized diffie-hellmann modulo a composite is no easier than factoring*, Inf. Process. Lett. **70** (1999), no. 2, 83–87.

[BCHK07] Dan Boneh, Ran Canetti, Shai Halevi, and Jonathan Katz, *Chosen-ciphertext security from identity-based encryption*, SIAM J. Comput. **36** (2007), no. 5, 1301–1328.

[BCOP04] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano, *Public key encryption with keyword search*, Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings, 2004, pp. 506–522.

[BF03] Dan Boneh and Matthew K. Franklin, *Identity-based encryption from the weil pairing*, SIAM J. Comput. **32** (2003), no. 3, 586–615.

[BG10] Zvika Brakerski and Shafi Goldwasser, *Circular and leakage resilient public-key encryption under subgroup indistinguishability - (or: Quadratic residuosity strikes back)*, Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings (Tal Rabin, ed.), Lecture Notes in Computer Science, vol. 6223, Springer, 2010, pp. 1–20.

[BGH07] Dan Boneh, Craig Gentry, and Michael Hamburg, *Space-efficient identity based encryption without pairings*, 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), October 20-23, 2007, Providence, RI, USA, Proceedings, 2007, pp. 647–657.

[BHHI10] Boaz Barak, Iftach Haitner, Dennis Hofheinz, and Yuval Ishai, *Bounded key-dependent message security*, 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, 2010, pp. 423–444.

[BHHO08] Dan Boneh, Shai Halevi, Michael Hamburg, and Rafail Ostrovsky, *Circular-secure encryption from decision diffie-hellman*, Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings, 2008, pp. 108–125.

[BLVW17] Zvika Brakerski, Vadim Lyubashevsky, Vinod Vaikuntanathan, and Daniel Wichs, *On lpn and other noisy decoding problems (working title)*, Personal Communication, 2017.

[BMR90] Donald Beaver, Silvio Micali, and Phillip Rogaway, *The round complexity of secure protocols (extended abstract)*, Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA, 1990, pp. 503–513.

[BRS02] John Black, Phillip Rogaway, and Thomas Shrimpton, *Encryption-scheme security in the presence of key-dependent messages*, Selected Areas in Cryptography, 9th Annual International Workshop, SAC 2002, St. John's, Newfoundland, Canada, August 15-16, 2002. Revised Papers, 2002, pp. 62–75.

[BW06] Xavier Boyen and Brent Waters, *Anonymous hierarchical identity-based encryption (without random oracles)*, Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings, 2006, pp. 290–307.

[CDG+17] Chongwon Cho, Nico Döttling, Sanjam Garg, Divya Gupta, Peihan Miao, and Antigoni Polychroniadou, *Laconic oblivious transfer and its applications*, Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part II, 2017, pp. 33–65.

[CHKP12] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert, *Bonsai trees, or how to delegate a lattice basis*, J. Cryptology **25** (2012), no. 4, 601–639.

[Coc01] Clifford Cocks, *An identity based encryption scheme based on quadratic residues*, Proceedings of the 8th IMA International Conference on Cryptography and Coding, 2001.

[DG17a] Nico Döttling and Sanjam Garg, *Identity-based encryption from the diffie-hellman assumption*, Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I, 2017, pp. 537–569.

[DG17b] ———, *From selective ibe to full ibe and selective hibe*, Theory of Cryptography Conference, 2017, To appear.

[DGK+10] Yevgeniy Dodis, Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan, *Public-key encryption schemes with auxiliary inputs*, Theory of Cryptography, 7th Theory of Cryptography Conference, TCC 2010, Zurich, Switzerland, February 9-11, 2010. Proceedings (Daniele Micciancio, ed.), Lecture Notes in Computer Science, vol. 5978, Springer, 2010, pp. 361–381.

[DKXY02] Yevgeniy Dodis, Jonathan Katz, Shouhuai Xu, and Moti Yung, *Key-insulated public key cryptosystems*, Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002, Proceedings, 2002, pp. 65–82.

[Gen06] Craig Gentry, *Practical identity-based encryption without random oracles*, Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings, 2006, pp. 445–464.

[GKW16] Rishab Goyal, Venkata Koppula, and Brent Waters, *Semi-adaptive security and bundling functionalities made generic and easy*, Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II (Martin Hirt and Adam D. Smith, eds.), Lecture Notes in Computer Science, vol. 9986, 2016, pp. 361–388.

[GL89] Oded Goldreich and Leonid A. Levin, *A hard-core predicate for all one-way functions*, STOC, ACM, 1989, pp. 25–32.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters, *Attribute-based encryption for fine-grained access control of encrypted data*, Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, Ioctober 30 - November 3, 2006, 2006, pp. 89–98.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan, *Trapdoors for hard lattices and new cryptographic constructions*, Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008, 2008, pp. 197–206.

[HLWW16] Carmit Hazay, Adriana López-Alt, Hoeteck Wee, and Daniel Wichs, *Leakage-resilient cryptography from minimal assumptions*, J. Cryptology **29** (2016), no. 3, 514–551.

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters, *Predicate encryption supporting disjunctions, polynomial equations, and inner products*, Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings, 2008, pp. 146–162.

[McC88] Kevin S. McCurley, *A key distribution system equivalent to factoring*, J. Cryptology **1** (1988), no. 2, 95–105.

[NS12] Moni Naor and Gil Segev, *Public-key cryptosystems resilient to key leakage*, SIAM J. Comput. **41** (2012), no. 4, 772–814.

[Rog91] Philip Rogaway, *The round-complexity of secure protocols*, Ph.D. thesis, MIT, 1991.

[Sha84] Adi Shamir, *Identity-based cryptosystems and signature schemes*, Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings, 1984, pp. 47–53.

[Shm85] Zahava Shmuely, *Composite diffie-hellman public-key generating systems are hard to break*, Technion Technical Report, 1985, http://www.cs.technion.ac.il/users/wwwb/cgi-bin/tr-get.cgi/1985/CS/CS0356.pdf.

[SW05] Amit Sahai and Brent Waters, *Fuzzy identity-based encryption*, Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings, 2005, pp. 457–473.

# A Blind Batch Encryption: Missing Proofs

## A.1 Proof of Lemma 3.1

Let $\mathcal{BBENC} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a batch encryption scheme with block size $B$ and succinctness $\alpha$, and let $\mathsf{Enc} = E_1||E_2$ be a decomposition such that $E_1(\mathsf{crs}; r)$ does not depend on $\mathbf{M}$ or $h$.[6] We then construct a batch encryption scheme $\mathcal{BBENC}' = (\mathsf{Setup}', \mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ as follows.

1. $\mathsf{Setup}'(1^\lambda, 1^{nt})$ calls $\mathsf{Setup}(1^\lambda, 1^n)$, outputting a common reference string $\mathsf{crs}$ used for hashing strings of length $nt$, where $t = \log(B)$.

2. $\mathsf{Gen}'_{\mathsf{crs}} = \mathsf{Gen}_{\mathsf{crs}} : \{0,1\}^{nt} \to \{0,1\}^{\alpha nt}$ is the same as before. We will interpret inputs $x \in \{0,1\}^{nt} \simeq [B]^n$ so that $x_i \in [B]$ for each $i \in [n]$.

3. $\mathsf{Enc}'(\mathsf{crs}, h, \mathbf{A})$, to batch encrypt an $nt \times 2$ matrix $\mathbf{A}$, defines the following $n \times B$ matrix $\mathbf{M}$:

$$\mathbf{M}_{i,\mathbf{b}} = (\mathbf{A}_{i||j,\mathbf{b}_j})_{j\in[t]} \text{ for all } i \in [n], \mathbf{b} \in [B] = \{0,1\}^t.$$

   It then outputs $\mathsf{Enc}(\mathsf{crs}, h, \mathbf{M})$.

4. The function $\mathsf{Dec}'(\mathsf{crs}, x, \mathsf{ct})$ computes

$$\mathbf{m}' = \mathsf{Dec}(\mathsf{crs}, x, \mathsf{ct}) \in (\{0,1\}^t)^n,$$

   and outputs $\mathbf{m}'$, interpreted as a vector in $\{0,1\}^{nt}$.

We want to prove the following two results: (1) $\mathcal{BBENC}'$ is a secure batch encryption scheme with block size 2 and succinctness $\alpha$, and (2) If $\mathcal{BBENC}$ is blind then so is $\mathcal{BBENC}'$, with blind decomposition inherited from $\mathcal{BBENC}$.

*Proof.* If $\mathcal{BBENC}$ satisfies batch correctness, then with probability $1 - \mathrm{negl}(\lambda)$, we have that for all $i \in [n]$,

$$\mathbf{m}'_i = \mathbf{M}_{i,x_i} = (\mathbf{A}_{i||j,(x_i)_j})_{j\in[t]},$$

where $x \in \{0,1\}^{nt}$ is arbitrary, $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$, $\mathbf{M}$ is constructed from $\mathbf{A}$ as described in $\mathsf{Enc}'(\mathsf{crs}, h, \mathbf{A})$, and $\mathbf{m}' = \mathsf{Dec}'(\mathsf{ct})$ for $\mathsf{ct} \leftarrow \mathsf{Enc}'(\mathsf{crs}, h, \mathbf{A})$. This is exactly the batch correctness condition we wanted for $\mathcal{BBENC}'$.

Security of $\mathcal{BBENC}'$ is inherited directly from the security of $\mathcal{BBENC}$. This is because for any pair of matrices $\mathbf{A}^{(0)}, \mathbf{A}^{(1)} \in \{0,1\}^{nt\times 2}$ such that $\mathbf{A}^{(0)}_{i||j,(x_i)_j} = \mathbf{A}^{(1)}_{i||j,(x_i)_j}$ for all $(i,j) \in [n] \times [t]$, the matrices $\mathbf{M}^{(0)}, \mathbf{M}^{(1)} \in (\{0,1\}^t)^{n\times B}$ constructed from $\mathbf{A}^{(0)}$ and $\mathbf{A}^{(1)}$ respectively have the property that $\mathbf{M}^{(0)}_{i,x_i} = \mathbf{M}^{(1)}_{i,x_i}$ for all $i \in [n]$. Thus, any adversary breaking the batch encryption security of $\mathcal{BBENC}'$ can clearly be used to break the (many message) batch encryption security of $\mathcal{BBENC}$ as well.

To see that blindness is preserved, note that if $\mathbf{A}$ is a uniformly random $nt \times 2$ matrix, then the matrix $\mathbf{M}$ constructed from $\mathbf{A}$ has the property that for any $x \in \{0,1\}^{nt}$, the marginal distribution on $(\mathbf{M}_{i,x_i})_{i\in[n]}$ is the uniform distribution on $(\{0,1\}^t)^n$. Thus, by ordinary batch encryption security, we have that $E'_2(\mathsf{crs}, h, \mathbf{A}; r) = E_2(\mathsf{crs}, h, \mathbf{M}; r)$ is computationally indistinguishable (within the

[6] Note that the trivial decomposition $E_1 \equiv 0$, $E_2 = \mathsf{Enc}$ satisfies this property.

batch encryption security game) from $E_2(\mathsf{crs}, h, \mathbf{M}'; r)$ for a uniformly random matrix $\mathbf{M}'$. Blindness now follows from the blindness of the original scheme.

The fact that $\mathcal{BBENC}$ and $\mathcal{BBENC}'$ have the same succinctness follows from the fact that the key generation algorithms $\mathsf{Gen}'$ and $\mathsf{Gen}$ are identical. □

## A.2 Proof of Lemma 3.2

Let $\mathcal{BBENC} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ denote a batch encryption scheme with block size 2 and succinctness $\frac{1}{2}$, and let $\mathsf{Enc} = E_1||E_2$ be a decomposition such that $E_1(\mathsf{crs}; r)$ does not depend on $\mathbf{M}$ or $h$. Additionally, let $\mathcal{BGBL} = (\mathsf{BGC.Garble}, \mathsf{BGC.Eval}, \mathsf{BGC.Sim})$ be a blind garbling scheme. We then construct a batch encryption scheme $\mathcal{BBENC}' = (\mathsf{Setup}', \mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ as follows.

1. $\mathsf{Setup}'(1^\lambda, 1^n)$ calls $\mathsf{Setup}(1^\lambda, 1^{2\lambda})$ $d = \log(\frac{n}{\lambda})$ times, obtaining and outputting $\mathsf{crs}' = (\mathsf{crs}_0, \mathsf{crs}_1, \ldots, \mathsf{crs}_{d-1})$. Note that each $\mathsf{crs}_i$ supports key generation from $\{0,1\}^{2\lambda} \to \{0,1\}^\lambda$.

2. $\mathsf{Gen}'(\mathsf{crs}', x \in \{0,1\}^n = (\{0,1\}^\lambda)^D)$ breaks $x = (y_0, ..., y_{D-1})$ into blocks of size $D = \frac{n}{\lambda} = 2^d$, so that $(y_i)_j = x_{i\lambda+j}$. Then, it defines a hash tree of depth $d$ in the following way: each node $v \in \{0,1\}^{\leq d}$ is labelled by a string $h_v \in \{0,1\}^\lambda$ such that $h_v = y_v$ for all $v \in \{0,1\}^d$ and for all $v \in \{0,1\}^i$ (for $i < d$), we have

$$h_v = \mathsf{Gen}(\mathsf{crs}_i, h_{v||0}||h_{v||1}).$$

   Finally, $\mathsf{Gen}'$ outputs $h_\epsilon$, where $\epsilon$ denotes the empty string.

3. $\mathsf{Enc}'(\mathsf{crs}', h_\epsilon, \mathbf{M})$, where $\mathbf{M} \in \{0,1\}^{n\times 2}$, defines for each $v \in \{0,1\}^{\leq d-1}$ a circuit $C[\mathsf{crs}_i, \mathbf{A}, r^{(v)}]$ with the following hardwired parameters: $\mathsf{crs}_i$ is the common reference string with index $i = |v|$, $\mathbf{A}$ denotes an arbitrary (hardwired) matrix in $\Sigma^{2\lambda\times 2}$ (for some alphabet $\Sigma = \{0,1\}^{\mathrm{poly}(\lambda)}$), and $r^{(v)}$ denotes fresh encryption randomness. The circuit $C[\mathsf{crs}_i, \mathbf{A}, r^{(v)}](h)$ takes in a public key $h \in \{0,1\}^\lambda$ and outputs $E_2(\mathsf{crs}_i, h, \mathbf{A}; r^{(v)})$. Then, $\mathsf{Enc}'$ outputs

$$\left((\mathsf{ct}_v)_{v\in\{0,1\}^{\leq d-1}}, (\widehat{C}_v)_{v\in\{0,1\}^{\leq d-1}}, (\mathsf{lab}^{(\epsilon)}_{j,(h_\epsilon)_j})_{j\in[\lambda]}\right),$$

   where

   - For every $v$ such that $|v| = d-1$, $(\widehat{C}_v, \overline{\mathsf{lab}}^{(v)}) \leftarrow \mathsf{BGC.Garble}(1^\lambda, C[\mathsf{crs}_{d-1}, \mathbf{M}[v], r^{(v)}])$ for fresh encryption randomness $r^{(v)}$ and $\mathbf{M}[v]$ defined to be the $2\lambda \times 2$ submatrix of $\mathbf{M}$ consisting of all rows of $\mathbf{M}$ in the range $[2v\lambda, (2v+2)\lambda - 1]$
   - For every $v \in \{0,1\}^i$ and $i < d-1$, $(\widehat{C}_v, \overline{\mathsf{lab}}^{(v)}) \leftarrow \mathsf{BGC.Garble}(1^\lambda, C[\mathsf{crs}_i, \mathbf{A}^{(v)}, r^{(v)}])$, where $\mathbf{A}^{(v)} := \begin{bmatrix} \overline{\mathsf{lab}}^{(v||0)} \\ \overline{\mathsf{lab}}^{(v||1)} \end{bmatrix}$ denotes the $2\lambda \times 2$ matrix written as the concatenation of $\overline{\mathsf{lab}}^{(v||0)}$ and $\overline{\mathsf{lab}}^{(v||1)} \in (\{0,1\}^\lambda)^{(\lambda\times 2)}$.
   - For every $v \in \{0,1\}^{\leq d-1}$, $\mathsf{ct}_v = E_1(\mathsf{crs}_i; r^{(v)})$ for $i = |v|$.

4. $\mathsf{Dec}'(\mathsf{crs}', x, \mathsf{ct})$ parses $\mathsf{ct} = ((\mathsf{ct}_v)_{v\in\{0,1\}^{\leq d-1}}, (\widehat{C}_v)_{v\in\{0,1\}^{\leq d-1}}, \widehat{L}^{(\epsilon)})$ and does the following:

   - For each $v \in \{0,1\}^{<d-1}$ (in order from shortest to longest string), compute $\mathsf{ct}'_v = \mathsf{BGC.Eval}(\widehat{C}_v, \widehat{L}^{(v)})$. Then, compute $(\widehat{L}^{(v||0)}, \widehat{L}^{(v||1)}) = \mathsf{Dec}(\mathsf{crs}_i, h_{v||0}||h_{v||1}, \mathsf{ct}_v||\mathsf{ct}'_v)$, where $i = |v|$.

- For each $v \in \{0,1\}^{d-1}$, compute $\mathsf{ct}'_v = \mathsf{BGC.Eval}(\widehat{C}_v, \widehat{L}^{(v)})$. Then, compute $\mathbf{m}'^{(v)} = \mathsf{Dec}(\mathsf{crs}_{d-1}, h_{v||0}||h_{v||1}, \mathsf{ct}_v||\mathsf{ct}'_v)$.
- Finally, output $(\mathbf{m}'^{(v)})_{v\in\{0,1\}^{d-1}}$.

We want to prove the following two results: (1) $\mathcal{BBENC}'$ is a secure, fully succinct batch encryption scheme with block size 2, and (2) if $\mathcal{BBENC}$ is blind, then $\mathcal{BBENC}'$ is blind with decomposition $E'_1(\mathsf{crs}', \mathbf{r}) = (\mathsf{ct}_v)_{v\in\{0,1\}^{(\leq d-1)}}$ and $E'_2(\mathsf{crs}', h_\epsilon, \mathbf{M}, \mathbf{r}) = (\widehat{C}_v)_{v\in\{0,1\}^{\leq d-1}}||\widehat{L}^{(\epsilon)}$.

**Proof of Correctness, Efficiency, and Succinctness.** We prove correctness by an inductive argument: if we are given $\mathsf{ct} \leftarrow \mathsf{Enc}'(\mathsf{crs}', h_\epsilon, \mathbf{M})$ for some $\mathbf{M}$, then for each step of the decryption process corresponding to $v \in \{0,1\}^{<d-1}$ (when decrypting $\mathsf{ct}$),

$$\mathsf{ct}'_v = E_2\left(\mathsf{crs}_i, h_v, \begin{bmatrix}\overline{\mathsf{lab}}^{(v||0)} \\ \overline{\mathsf{lab}}^{(v||1)}\end{bmatrix}; r^{(v)}\right)$$

by the correctness of $\mathcal{BGBL}$ (for $i = |v|$), so

$$L_j^{(v||0)} = \mathsf{lab}^{(v||0)}_{j,(h_{v||0})_j} \text{ and } L_j^{(v||1)} = \mathsf{lab}^{(v||1)}_{j,(h_{v||1})_j}$$

for each $j$ by the correctness of $\mathcal{BBENC}$. Thus, we also have that for each $v \in \{0,1\}^{d-1}$,

$$\mathsf{ct}'_v = E_2(\mathsf{crs}_{d-1}, h_v, \mathbf{M}[v]; r^{(v)})$$

by the correctness of $\mathcal{BGBL}$, so that $(\mathbf{m}'^{(v)})_j = (\mathbf{M}[v])_{j,(y_v)_j}$ for every $j$, as desired.

Efficiency of encryption follows from the fact that we execute $D-1 = 2^d-1$ garbling operations on circuits of some bounded $\mathrm{poly}(\lambda)$ size, while we execute $(D-1)$ $E_1$ operations on matrices with at most $\lambda$-bit entries. Thus, the entire encryption operation takes $\mathrm{poly}(D, \lambda) = \mathrm{poly}(n, \lambda)$ time. Efficiency of decryption follows similarly.

Full succinctness follows from the fact that a public key $h_\epsilon$ assigned to $x$ under $\mathsf{crs}'$ is a single $\mathcal{BBENC}$ public key with size $\lambda$.

**Proof of Batch Encryption Security.** As stated before, this part of the proof only requires that $\mathcal{BBENC}$ is a secure batch encryption scheme and $\mathcal{BGBL}$ is a secure garbling scheme (i.e. no blindness is required).

Suppose that $\mathcal{A}$ is an efficient adversary playing the batch encryption security game (as defined in Definition 3.5), and let $H_0 = H_{\epsilon,0}$ denote the probability space underlying the batch encryption security game. We now define the hybrids $H_{v,1}$ and $H_{v_{\text{next}},0}$ for all $v \in \{0,1\}^{\leq d-1}$, where $v_{\text{next}} \in \{0,1\}^{\leq d}$ is defined to be $v+1$ if $v \neq 1^i$ for some $i$, and $v_{\text{next}} := 0^{i+1}$ if $v = 1^i$. These hybrids denote probability spaces underlying altered security games in which the challenge ciphertext is modified in various ways.

- For all $v \in \{0,1\}^{<d-1}$ with $i = |v|$, hybrid $H_{v,1}$ is obtained by modifying how $(\widehat{C}_v, \widehat{L}^{(v)})$ is computed as compared to $H_{v,0}$. Instead of honestly computing $(\widehat{C}_v, \overline{\mathsf{lab}}^{(v)}) \leftarrow \mathsf{BGC.Garble}(1^\lambda, C[\mathsf{crs}_i, \mathbf{A}^{(v)}; r^{(v)}])$ and setting $L_j^{(v)} = \mathsf{lab}^{(v)}_{j,(h_v)_j}$, we compute $\mathsf{ct}'_v = E_2(\mathsf{crs}_i, h_v, \mathbf{A}^{(v)}; r^{(v)}))$, and then compute (using the simulator) $(\widetilde{C}_v, \widetilde{L}^{(v)}) \leftarrow \mathsf{BGC.Sim}(1^\lambda, 1^{|C[\mathsf{crs}_i, \mathbf{A}^{(v)}, r^{(v)}]|}, 1^\lambda, \mathsf{ct}'_v)$. By induction, we maintain the invariant that in $H_{v,0}$ only $\widehat{L}^{(v)}$ (that is, the collection of labels $\overline{\mathsf{lab}}^{(v)}_{j,(h_v)_j}$) is used in any futher computation.

- For all $v \in \{0,1\}^{<d-1}$, hybrid $H_{v_{\text{next}},0}$ is obtained by modifying the matrix $\mathbf{A}^{(v)}$ as compared to $H_{v,1}$. Instead of setting $\mathbf{A}^{(v)} = \begin{bmatrix} \overline{\mathsf{lab}}^{(v||0)} \\ \overline{\mathsf{lab}}^{(v||1)} \end{bmatrix}$ as in $H_{v,1}$, we construct the matrix $\mathbf{A}^{(v)}$ by defining $\mathbf{A}^{(v)} = \begin{bmatrix} \tilde{\mathbf{A}}^{(v||0)} \\ \tilde{\mathbf{A}}^{(v||1)} \end{bmatrix}$, where $\tilde{\mathbf{A}}^{(w)}_{j,h_{wj}} = \mathsf{lab}^{(w)}_{j,(h_w)_j}$ and $\tilde{\mathbf{A}}^{(v)}_{j,1-(h_w)_j} = R^{(v)}_{j+a\lambda,1-(h_w)_j}$ for all $j \in [\lambda], b \in \{0,1\}, a \in \{0,1\}$, $w = v||a$. Here, each $R^{(v)}_{j+a\lambda,b}$ denotes an independent uniformly random string.
- For all $v \in \{0,1\}^{d-1}$, hybrid $H_{v,1}$ is obtained by modifying how $(\widehat{C}_v, \widehat{L}^{(v)})$ is computed as compared to $H_{v,0}$. Instead of honestly computing $(\widehat{C}_v, \overline{\mathsf{lab}}^{(v)}) \leftarrow \mathsf{BGC.Garble}(1^\lambda, C[\mathsf{crs}_{d-1}, \mathbf{M}^{(\beta)}[v], r^{(v)}])$ (where $\beta$ is the challenger's random bit) and setting $L^{(v)}_j = \mathsf{lab}^{(v)}_{j,(h_v)_j}$, we compute the ciphertext $\mathsf{ct}'_v = E_2(\mathsf{crs}_{d-1}, h_v, \mathbf{M}^{(\beta)}[v]; r^{(v)})$, and then compute (using the simulator) $(\widetilde{C}_v, \widetilde{L}^{(v)}) \leftarrow \mathsf{BGC.Sim}(1^\lambda, 1^{|C[\mathsf{crs}_{d-1}, \mathbf{M}^{(\beta)}[v], r^{(v)}]|}, 1^\lambda, \mathsf{ct}'_v)$.
- For all $v \in \{0,1\}^{d-1}$, hybrid $H_{v_{\text{next}},0}$ is obtained from $H_{v,1}$ by replacing $\mathbf{M}^{(\beta)}[v]$ by $\mathbf{M}^{(c)}[v]$ for an independent uniformly random bit $c \xleftarrow{\$} \{0,1\}$.

Clearly $\Pr_{\tilde{H}}[\mathcal{A} \text{ wins}] = \frac{1}{2}$ for $\tilde{H} := H_{0^d,0}$, as in this hybrid the challenge ciphertext received by the adversary is independent of the bit $\beta$. Thus, it suffices to prove that $\mathcal{A}$'s views in $H_{v,0}$, $H_{v,1}$, and $H_{v_{\text{next}},0}$ are computationally indistinguishable for every $v \in \{0,1\}^{\leq d-1}$, which we denote by the shorthand $H_{v,0} \approx_c H_{v,1} \approx_c H_{v_{\text{next}},0}$.

**Claim A.1.** *For every $v \in \{0,1\}^{\leq d-1}$, $H_{v,0} \approx_c H_{v,1}$.*

This follows directly from the simulation security $\mathcal{BGBL}$, crucially using the fact that only labels $\mathsf{lab}^{(v)}_{j,(h_v)_j}$ are used in $H_{v,0}$.

**Claim A.2.** *For every $v \in \{0,1\}^{<d-1}$, $H_{v,1} \approx_c H_{v_{next},0}$.*

This follows from the security of $\mathcal{BBENC}$, crucially using the fact $\mathbf{A}^{(v)}_{j+a\lambda,(h_{v||a})_j} = \tilde{\mathbf{A}}^{(v||a)}_{j,(h_{v||a})_j}$ for each $j \in [\lambda], a \in \{0,1\}$.

**Claim A.3.** *For every $v \in \{0,1\}^{d-1}$, $H_{v,1} \approx_c H_{v_{next},0}$.*

This also follows from the security of $\mathcal{BBENC}$, crucially using the fact that $\mathbf{M}^{(0)}[x] = \mathbf{M}^{(1)}[x]$.

**Proof of Blindness.** Next, we show that $\mathcal{BBENC}'$ is blind assuming that $\mathcal{BBENC}$ and $\mathcal{BGBL}$ are blind. Note that syntatically, $E'_1$ does not depend on $h_\epsilon$ or $\mathbf{M}$, as desired.

Suppose that $\mathcal{A}$ is an efficient adversary playing the blind batch encryption security game (as described in Definition 3.7), and let $H_0 = H_{\epsilon,0}$ denote the probability space underling the blind batch encryption security game. We define the hybrids $H_{v,1}, H_{v_{\text{next}},0}$ (for all $v \in \{0,1\}^{<d-1}$) analogously as in the batch encryption security proof, so that $H_{\epsilon,0} \approx_c H_{\epsilon,1} \approx_c \ldots \approx_c H_{1^{d-2},1} \approx_c H_{0^{d-1},0}$ (by the same proof). We additionally define hybrids $H_{v,0}$ for each $v \in \{0,1\}^{d-1}$ similarly to the previous security proof, although we skip the step that changes $\mathbf{M}$ to a random matrix (in the blind batch encryption security game, $\mathbf{M}$ is already random). That is, now $H_{v_{\text{next}},0}$ is just obtained

from $H_{v,0}$ by replacing $(\widehat{C}_v, \widehat{L}^{(v)})$ with its simulated version. We then have that $H_{v,0} \approx_c H_{v_{\text{next}},0}$ for all $v \in \{0,1\}^{d-1}$ (by the same reduction to garbled circuit security).

Next, we additionally define hybrids $H_{v,2}$ and $H_{v,3}$ for all $v \in \{0,1\}^{\leq d-1}$; the hybrids are defined in *reverse order* from before.

- Hybrid $H_{1^{d-1},2} = H_{1^{d-1},0}$ by definition.
- For every $v \in \{0,1\}^{d-1}$, hybrid $H_{v,3}$ is obtained by modifying how $\mathsf{ct}'_v$ is computed as compared to $H_{v,2}$. Instead of setting $\mathsf{ct}'_v := E_2(\mathsf{crs}_{d-1}, h_v, \mathbf{M}[v]; r^{(v)})$, we compute $e_v = E_2(\mathsf{crs}_{d-1}, h_v, \mathbf{M}[v], r^{(v)})$ and set $\mathsf{ct}'_v$ to be a random ciphertext length $|e_v|$.
- For every $v \in \{0,1\}^{\leq d-1}$, hybrid $H_{v_{\text{prev}},2}$ is obtained by modifying how $(\widetilde{C}_v, \widetilde{L}^{(v)})$ is computed as compared to $H_{v,3}$. The simulator output $(\widetilde{C}_v, \widetilde{L}^{(v)}) \leftarrow \mathsf{BGC.Sim}(1^\lambda, 1^{|C[\mathsf{crs}_i, \mathbf{A}^{(v)}, r^{(v)}]|}, 1^\lambda, \mathsf{ct}'_v)$ (as in $H_{v,3}$) is replaced with a random pair $(\widetilde{C}_v, \widetilde{L}^{(v)})$ of length $|\mathsf{BGC.Sim}(1^\lambda, 1^{|C[\mathsf{crs}_i, \mathbf{A}^{(v)}, r^{(v)}]|}, 1^\lambda, \mathsf{ct}'_v)|$.
- For every $v \in \{0,1\}^{<d-1}$, hybrid $H_{v,3}$ is obtained by modifying how $\mathsf{ct}'_v$ is computed as compared to $H_{v,2}$. Instead of setting $\mathsf{ct}'_v := E_2(\mathsf{crs}_i, h_v, \mathbf{A}^{(v)}; r^{(v)})$, we compute $e_v = E_2(\mathsf{crs}_i, h_v, \mathbf{A}^{(v)}; r^{(v)})$ and set $\mathsf{ct}'_v$ to be a uniformly random ciphertext of length $|e_v|$.

It is clear that for $\tilde{H} = H_{\epsilon,3}$, $\Pr_{\tilde{H}}[\mathcal{A} \text{ wins}] = \frac{1}{2}$, as in $\tilde{H}$ the challenge ciphertext recieved by $\mathcal{A}$ no longer depends on the value of $\beta$ (in fact, it is always the original challenge ciphertext in $\beta = 1$ mode). Thus, it suffices to show that all adjacent hybrids defined above are computationally indistinguishable.

**Claim A.4.** *$H_{v,2} \approx_c H_{v,3}$ for all $v \in \{0,1\}^{d-1}$.*

This follows from the blindness of $\mathcal{BBENC}$, crucially using the fact that $\mathbf{M}$ is uniformly random and independent of the rest of the $\mathcal{BBENC}'$ security experiment.

**Claim A.5.** *$H_{v,2} \approx_c H_{v,3}$ for all $v \in \{0,1\}^{<d-1}$*

This also follows from the blindness of $\mathcal{BBENC}$, crucially using the fact that in $H_{v,3}$, the matrix $\mathbf{A}^{(v)}$ is uniformly random and independent of the rest of the $\mathcal{BBENC}'$ experiment.

**Claim A.6.** *$H_{v,3} \approx_c H_{v_{prev},2}$ for all $v \in \{0,1\}^{\leq d-1}$*

This follows from the blindness of $\mathcal{BGBL}$, crucially using the fact that the ciphertext $\mathsf{ct}'_v$ plugged into $\mathsf{BGC.Sim}$ in hybrid $H_{v,3}$ is uniformly random.

This completes the proof of Lemma 3.2.