Anonymous IBE, Leakage Resilience and Circular Security from New Assumptions

Zvika Brakerski[1(✉)], Alex Lombardi[2], Gil Segev[3], and Vinod Vaikuntanathan[2]

[1] Weizmann Institute of Science, Rehovot, Israel
zvika.brakerski@weizmann.ac.il
[2] MIT, Cambridge, USA
[3] Hebrew University of Jerusalem, Jerusalem, Israel

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.

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

The full version of this paper [BLSV17] is available on ePrint.

J. B. Nielsen and V. Rijmen (Eds.): EUROCRYPT 2018, LNCS 10820, pp. 535–564, 2018.
https://doi.org/10.1007/978-3-319-78381-9_20

msk can generate individual secret keys for users $\mathsf{sk}_{\mathsf{id}}$ according to their identity id, and encryption is performed using a master public key (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 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 Döttling and Garg [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 id does not expose 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 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 id using a key for id', the decryption process will succeed up to the node of divergence between sk and 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 id and (a known) 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 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, YZW+17]. 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 achieve 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 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 encryption 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 id different from either of the challenge identities id_0 and 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

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

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. For lack of space, we refer the reader to our full version [BLSV17] for the details on the leakage-resilience and KDM security constructions.

1.2 Concurrent Work

In concurrent and independent work, Döttling, Garg, Hajiabadi, and Masny [DGHM18] construct (non-anonymous) IBE from a subexponential assumption on constant-noise LPN (similar in spirit to our assumption). In another concurrent and independent work, Kitagawa and Tanaka [KT18] construct *KDM-secure IBE* from any IBE along with any KDM-secure secret key encryption scheme. Since we construct both IBE and KDM-secure PKE from Batch Encryption, combining [KT18] with our work yields KDM-secure IBE from Batch Encryption (and hence constructions from CDH/Factoring and from $\log^2(n)/n$-noise LPN).

1.3 Our Techniques

The rest of the paper is organized as follows. In Sect. 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 Sect. 4. Then, in Sect. 5, we show how to use (blind) batch encryption to construct a weakly compact (blind) IBE scheme. In Sect. 6, we bootstrap the weakly compact (blind) IBE scheme into a full-fledged (blind) IBE scheme. The applications to leakage resilience and KDM security, as well as many details in the following sections, are deferred to the full version of our paper [BLSV17].

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 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 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 id' hardcoded. The circuit $C[m, \mathsf{id}']$, on input an mpk, produces an encryption of m under mpk with identity 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 (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 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 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 (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 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 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 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 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 $\text{poly}(\lambda)$ 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 Sect. 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 λ. 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 λ. 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 mpk of the weakly compact IBE scheme, and indeed its length is λ, 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 id-th of them and uses it to encrypt the message. The IBE secret key for identity 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 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 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^λ *and an identity length* 1^t*. It returns public parameters* 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* pp *and returns a master public key* mpk *and master secret key* msk*.*
3. $\mathsf{Keygen}(\mathsf{pp}, \mathsf{msk}, \mathsf{id})$ *takes as input public parameters* pp *and the master secret key* msk*. It outputs a secret key* $\mathsf{sk}_{\mathsf{id}}$ *associated to* id*.*
4. $\mathsf{Enc}(\mathsf{pp}, \mathsf{mpk}, \mathsf{id}, m)$ *encrypts a message* m *to a specified identity* id*. It outputs a ciphertext* ct*.*
5. $\mathsf{Dec}(\mathsf{pp}, \mathsf{sk}, \mathsf{ct})$ *decrypts a ciphertext* ct *with secret key* 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* id^* *was never queried by* $\mathcal{A}$ *to its* Keygen *oracle.*

Definition 2 (Anonymous IBE). *An anonymous IBE scheme also has the syntax (*$\mathsf{Params}, \mathsf{Setup}, \mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec}$*) 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 ϵ-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 ϵ. We note that any $q \geq \mathsf{ord}(g) \cdot \lceil 1/\epsilon \rceil$ is an ϵ-randomizer, so it is sufficient to have an upper bound on the order of g in order to compute a randomizer for any ϵ.

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 with respect to $\mathcal{G}$, 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 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 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 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 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, μ 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 algorithm $\mathcal{A}$, $\Pr_{(x,z)\leftarrow(X,Z),\mathcal{A}}[\mathcal{A}(1^\lambda, z) = x] = \mathrm{negl}(\lambda)$, then $(z, \mathsf{gl\text{-}enc}(x, \mu))$ is computationally indistinguishable from $(z, U_{\ell+1})$ for any μ (possibly dependent on z). We furthermore note that if μ 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 λ and key length n, and outputs a common reference string 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 crs, the public key h, and a matrix $\mathbf{M} \in \{0,1\}^{n \times B}$ and outputs a ciphertext ct. For the purpose of defining the blinding property below, the ciphertext 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 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 crs, the public key h, an index $i \in [n]$, and a message $\mathbf{m} \in \{0,1\}^B$ and outputs a ciphertext ct. As above, the ciphertext 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 crs, the secret key x, an index $i \in [n]$, and a ciphertext 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 ith 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 4 (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* Enc*.*

Definition 5 (δ-Pointwise-Correctness for $\mathsf{SingleEnc}$). *Letting* $\mathsf{crs} = \mathsf{Setup}(1^\lambda, 1^n)$*, then for all* $x, i, \mathbf{m}$*, 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 6. *A batch encryption scheme is* α*-succinct if for* $\mathsf{crs} = \mathsf{Setup}(1^\lambda, 1^n)$ *and* $h = \mathsf{Gen}(\mathsf{crs}, x)$ *for some* $x \in [B]^n$*, it holds that* $|h| \leq \alpha n \log B$*.*

Definition 7. *A batch encryption scheme is fully succinct if for* $\mathsf{crs} = \mathsf{Setup}(1^\lambda, 1^n)$ *and* $h = \mathsf{Gen}(\mathsf{crs}, x)$ *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 8 (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^λ *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* 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* ct *to the adversary.*
5. *The adversary outputs a bit* β' *and wins if* $\beta' = \beta$*.*

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

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

Definition 9 ($\mathsf{SingleEnc}$ Security). *We say that a batch encryption scheme satisfies* $\mathsf{SingleEnc}$*-security if no polynomial time adversary can win the following game with probability* $\geq 1/2 + 1/poly(\lambda)$*:*

1. *The adversary takes* 1^λ *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* 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* ct *to the adversary.*
5. *The adversary outputs a bit* β' *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 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 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 10 (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* $\geq \frac{1}{2} + 1/poly(\lambda)$*.*

1. *The adversary takes* 1^λ *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* ct *as follows.*
   - *If* $\beta = 0$ *then* $\mathsf{ct} = (\mathsf{subct}_1, \mathsf{subct}_2)$*.*
   - *If* $\beta = 1$ *then sample a random bit string* subct_2' *of the same length as* 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* β' *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 Enc and adopt the notation that outputs of E_1 are denoted by subct_1 and outputs of E_2 are denoted by 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 the full version [BLSV17].

Lemma 1. *Suppose that there is an* α*-succinct (blind) batch encryption scheme with block size* B*. Then, there is an* α*-succinct (blind) batch encryption scheme with block size* 2*.*

From α-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 Sect. 4), similar to its use in Sects. 5 and 6. We state the lemma below and provide the proof in the full version [BLSV17].

Lemma 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 Sect. 2.2. Recall the Goldreich-Levin encoding/decoding procedure as per Sect. 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
$$\mathsf{ct} = \left(\mathsf{subct}_1 = \{\hat{g}_{j,b}\}_{j\neq i, b\in\{0,1\}}, \mathsf{subct}_2 = \{\mu_{i,b}\}_{b\in\{0,1\}}\right).$$
4. $\mathsf{CDH\text{-}BE.SingleDec}(\mathsf{crs}, x, i, \mathsf{ct})$. Given $\mathsf{ct} = \left(\{\hat{g}_{j,b}\}_{j\neq i, b\in\{0,1\}}, \{\mu_{i,b}\}_{b\in\{0,1\}}\right)$. 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, we note that this scheme is *fully succinct* (see Definition 7; note that $h \in \mathbb{G}$ has a fixed $\mathrm{poly}(\lambda)$ size representation by assumption).

Lemma 3. *The scheme* CDH-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^λ 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 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 Sect. 2.4), this implies the security of the scheme as per Definition 9.

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

Proof. Consider the game in Definition 10. We first of all note that in our scheme, subct_1 is independent of $h, \mathbf{m}$ and therefore the marginal distribution of subct_1 is identical regardless of the value of β. From the properties of gl-enc (see Sect. 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 δ-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 Sect. 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 δ point-wise correctness can be amplified, however this amplification does not preserve the blindness property. Therefore, even though our δ-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 $\mathbf{A}$.

[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.

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 5. *The scheme* $\mathsf{LPN\text{-}BE}$ *is* $1/poly(n)$*-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 ϵ, 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 6. *The scheme* $\mathsf{LPN\text{-}BE}$ *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 $\mathsf{SingleEnc}$ security game in Definition 9 (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.

Definition 11 (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* λ *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^λ*) 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:*

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

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 12 (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.)*

Using a construction essentially identical to the point-and-permute garbled circuits of [BMR90, Rog91], we prove the following result.

Lemma 7. *Assuming the existence of one-way functions, there exists a blind garbling scheme.*

We refer the reader to the full version [BLSV17] for details.

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 10), we say that an IBE scheme is blind if, when encrypting (under some identity 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 13 (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* Enc.

Lemma 8. *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 (id_0, m_0) and (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 Sect. 6, we show that full (blind) IBE can be bootstrapped from weakly compact (blind) IBE.

Definition 14 (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:*

- 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* $\mathrm{poly}(\lambda, T)$*. /See Footnote 3/*

Definition 15. *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 16 (Blind Public Key Encryption). *An blind public key encryption scheme (with public parameters) is a public key encryption scheme*

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

$(\mathsf{Params}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *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*

$$\left\{\mathsf{pp} \leftarrow \mathsf{Params}(1^\lambda), (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(\mathsf{pp}), m \xleftarrow{\$} \{0,1\}^n : (\mathsf{pp}, \mathsf{pk}, \mathsf{sk}, \mathsf{Enc}(\mathsf{pp}, \mathsf{pk}, m))\right\}$$

is computationally indistinguishable from the distribution

$$\left\{\mathsf{pp} \leftarrow \mathsf{Params}(1^\lambda), (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(\mathsf{pp}), m \xleftarrow{\$} \mathcal{M}, L = |E_2(\mathsf{pp}, \mathsf{pk}, m; r)|,\right.$$

$$\left.\mathsf{subct}_2 \xleftarrow{\$} \{0,1\}^L : (\mathsf{pp}, \mathsf{pk}, \mathsf{sk}, E_1(\mathsf{pp}; r)||\mathsf{subct}_2)\right\}.$$

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.*

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 $\mathsf{BPKE}.E_1||\mathsf{BPKE}.E_2$ as in Definition 16;
- 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 $\mathsf{Batch}.E_1||\mathsf{Batch}.E_2$ as in Definition 10. Moreover, we assume that $\mathcal{BBENC}$ is fully succinct.

The construction works as follows.

1. $\mathsf{WBIBE.Params}(1^T)$: Given a bound T on the number of identities, the parameter generation algorithm 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, a master public key, an identity 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 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 ℓ is defined to be the output 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 show that this scheme is a weakly compact blind IBE scheme.

Theorem 1. *Suppose $\mathcal{BPKE}$ 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 defer the reader to the full version [BLSV17] for details.

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 Sect. 6.1 and analyzed in the full version of our paper [BLSV17]. Recall that a high level overview was provided in the introduction (Sect. 1.3).

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 $T = \mathrm{poly}(\lambda)$ 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 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 ϵ 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 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

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

on input $\mathsf{wbibe.mpk}$, 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

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

on input $\mathsf{wbibe.mpk}$. 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 $(\mathsf{lab}^{(0)}_{j,b_{0,j}})_{j\in[S]}$.

– $\mathsf{BIBE.Dec}(\mathsf{bibe.pp}, \mathsf{bibe.sk_{id}}, \mathsf{bibe.ct})$: 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 $1 \le i \le n$, 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 $L^{(i)}_j \leftarrow \mathsf{WBIBE.Dec}(\mathsf{wbibe.pp}, \mathsf{sk}_{i,j}, \mathsf{ct}_{i,j,b_{i,j}} || \mathsf{ct}'_{i,j,b_{i,j}})$ for all $j \in [S]$ and $b_{i,j} = \mathsf{wbibe.mpk}^{(\le i)}[j]$. Let $\widehat{L}^{(i)}$ denote the collection of all $L^{(i)}_j$.

  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 2. *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.*

We refer the reader to the full version [BLSV17] for details.

Acknowledgments. The first author was supported by the Israel Science Foundation (Grant No. 468/14), Binational Science Foundation (Grants No. 2016726, 2014276), ERC Project 756482 REACT and European Union PROMETHEUS Project (Horizon 2020 Research and Innovation Program, Grant 780701). The third author was 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. The second and fourth authors were supported 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 and 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. The second author was in addition supported by an Akamai Presidential Fellowship.

References

[ABB10] Agrawal, S., Boneh, D., Boyen, X.: Efficient lattice (H)IBE in the standard model. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 553–572. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_28

[ABC+08] Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., Malone-Lee, J., Neven, G., Paillier, P., Shi, H.: Searchable encryption revisited: consistency properties, relation to anonymous IBE, and extensions. J. Cryptol. **21**(3), 350–391 (2008)

[ACPS09] Applebaum, B., Cash, D., Peikert, C., Sahai, A.: Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 595–618. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03356-8_35

[AGV09] Akavia, A., Goldwasser, S., Vaikuntanathan, V.: Simultaneous hardcore bits and cryptography against memory attacks. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 474–495. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00457-5_28

[Ale11] Alekhnovich, M.: More on average case vs approximation complexity. Comput. Complex. **20**(4), 755–786 (2011)

[App11] Applebaum, B.: Key-dependent message security: generic amplification and completeness. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 527–546. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20465-4_29

[BBR99] Biham, E., Boneh, D., Reingold, O.: Breaking generalized Diffie-Hellmann modulo a composite is no easier than factoring. Inf. Process. Lett. **70**(2), 83–87 (1999)

[BCHK07] Boneh, D., Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. SIAM J. Comput. **36**(5), 1301–1328 (2007)

[BCOP04] Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_30

[BF03] Boneh, D., Franklin, M.K.: Identity-based encryption from the weil pairing. SIAM J. Comput. **32**(3), 586–615 (2003)

[BG10] Brakerski, Z., Goldwasser, S.: Circular and leakage resilient public-key encryption under subgroup indistinguishability. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 1–20. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14623-7_1

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

[BHHI10] Barak, B., Haitner, I., Hofheinz, D., Ishai, Y.: Bounded key-dependent message security. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 423–444. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_22

[BHHO08] Boneh, D., Halevi, S., Hamburg, M., Ostrovsky, R.: Circular-secure encryption from decision Diffie-Hellman. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 108–125. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85174-5_7

[BLSV17] Brakerski, Z., Lombardi, A., Segev, G., Vaikuntanathan, V.: Anonymous IBE, leakage resilience and circular security from new assumptions. Cryptology ePrint Archive, Report 2017/967 (2017). https://eprint.iacr.org/2017/967

[BLVW17] Brakerski, Z., Lyubashevsky, V., Vaikuntanathan, V., Wichs, D.: Cryptographic hashing and worst-case hardness for LPN via code smoothing. Personal Communication (2017)

[BMR90] Beaver, D., Micali, S., Rogaway, P.: The round complexity of secure protocols (extended abstract). In: Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, Baltimore, Maryland, USA, 13–17 May 1990, pp. 503–513 (1990)

[BRS02] Black, J., Rogaway, P., Shrimpton, T.: Encryption-scheme security in the presence of key-dependent messages. In: Nyberg, K., Heys, H. (eds.) SAC 2002. LNCS, vol. 2595, pp. 62–75. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36492-7_6

[BW06] Boyen, X., Waters, B.: Anonymous hierarchical identity-based encryption (without random oracles). In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 290–307. Springer, Heidelberg (2006). https://doi.org/10.1007/11818175_17

[CDG+17] Cho, C., Döttling, N., Garg, S., Gupta, D., Miao, P., Polychroniadou, A.: Laconic oblivious transfer and its applications. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10402, pp. 33–65. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63715-0_2

[CHKP12] Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai trees, or how to delegate a lattice basis. J. Cryptol. **25**(4), 601–639 (2012)

[Coc01] Cocks, C.: An identity based encryption scheme based on quadratic residues. In: Proceedings of the 8th IMA International Conference on Cryptography and Coding (2001)

[DG17a] Döttling, N., Garg, S.: Identity-based encryption from the Diffie-Hellman assumption. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10401, pp. 537–569. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_18

[DG17b] Döttling, N., Garg, S.: From selective IBE to full IBE and selective HIBE. In: Theory of Cryptography Conference (2017, to appear)

[DGHM18] Döttling, N., Garg, S., Hajiabadi, M., Masny, D.: New constructions of identity-based and key-dependent message secure encryption schemes. In: IACR International Workshop on Public Key Cryptography. Springer (2018). https://eprint.iacr.org/2017/978

[DGK+10] Dodis, Y., Goldwasser, S., Tauman Kalai, Y., Peikert, C., Vaikuntanathan, V.: Public-key encryption schemes with auxiliary inputs. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 361–381. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-11799-2_22

[DKXY02] Dodis, Y., Katz, J., Xu, S., Yung, M.: Key-insulated public key cryptosystems. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 65–82. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-46035-7_5

[Gen06] Gentry, C.: Practical identity-based encryption without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 445–464. Springer, Heidelberg (2006). https://doi.org/10.1007/11761679_27

[GKW16] Goyal, R., Koppula, V., Waters, B.: Semi-adaptive security and bundling functionalities made generic and easy. In: Hirt, M., Smith, A. (eds.) TCC 2016. LNCS, vol. 9986, pp. 361–388. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53644-5_14

[GL89] Goldreich, O., Levin, L.A.: A hard-core predicate for all one-way functions. In: STOC, pp. 25–32. ACM (1989)

[GPSW06] Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, 30 October–3 November 2006, pp. 89–98 (2006)

[GPV08] Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, 17–20 May 2008, pp. 197–206 (2008)

[HLWW16] Hazay, C., López-Alt, A., Wee, H., Wichs, D.: Leakage-resilient cryptography from minimal assumptions. J. Cryptol. **29**(3), 514–551 (2016)

[KSW08] Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 146–162. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78967-3_9

[KT18] Kitagawa, F., Tanaka, K.: Key dependent message security and receiver selective opening security for identity-based encryption. In: IACR International Workshop on Public Key Cryptography. Springer (2018). https://eprint.iacr.org/2017/987

[McC88] McCurley, K.S.: A key distribution system equivalent to factoring. J. Cryptol. **1**(2), 95–105 (1988)

[NS12] Naor, M., Segev, G.: Public-key cryptosystems resilient to key leakage. SIAM J. Comput. **41**(4), 772–814 (2012)

[Rog91] Rogaway, P.: The round-complexity of secure protocols. Ph.D. thesis, MIT (1991)

[Sha84] Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985). https://doi.org/10.1007/3-540-39568-7_5

[Shm85] Shmuely, Z.: 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] Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005). https://doi.org/10.1007/11426639_27

[YZW+17] Yu, Y., Zhang, J., Weng, J., Guo, C., Li, X.: Learning parity with noise implies collision resistant hashing. https://eprint.iacr.org/2017/1260.pdf