Encryptor Combiners: A Unified Approach to Multiparty NIKE, (H)IBE, and Broadcast Encryption

Fermi Ma Mark Zhandry

Princeton University
{fermim,mzhandry}@princeton.edu

February 16, 2017

Abstract

We define the concept of an encryptor combiner. Roughly, such a combiner takes as input n public keys for a public key encryption scheme, and produces a new *combined* public key. Anyone knowing a secret key for *one* of the input public keys can learn the secret key for the combined public key, but an outsider who just knows the input public keys (who can therefore compute the combined public key for himself) cannot decrypt ciphertexts from the combined public key. We actually think of public keys more generally as encryption procedures, which can correspond to, say, encrypting to a particular identity under an IBE scheme or encrypting to a set of attributes under an ABE scheme.

We show that encryptor combiners satisfying certain natural properties can give natural constructions of multi-party non-interactive key exchange, low-overhead broadcast encryption, and hierarchical identity-based encryption. We then show how to construct two different encryptor combiners. Our first is built from universal samplers (which can in turn be built from indistinguishability obfuscation) and is sufficient for each application above, in some cases improving on existing obfuscation-based constructions. Our second is built from lattices, and is sufficient for hierarchical identity-based encryption. Thus, encryptor combiners serve as a new abstraction that (1) is a useful tool for designing cryptosystems, (2) unifies constructing hierarchical IBE from vastly different assumptions, and (3) provides a target for instantiating obfuscation applications from better tools.

1 Introduction

Cryptography has always benefited greatly from abstractions such as one-way functions, collision resistant hashing, pseudorandom generators, pseudorandom functions, encryption, digital signatures, hard-core bits, trapdoor permutations, etc. The celebrated result that one-way functions implies digital signatures is a perfect example of this: first, a simple *one-time signature* is built from one-way functions [Lam79]. The message size of the one-time signature needs to be expanded using a *target collision resistant hash function*, which in turn can be built from any one-way function [Rom90]. Then a large-message one-time signature can be converted into a many-time signature using a *pseudorandom function*, which can be built from a *pseudorandom generator* [GGM86], which in turn can be built from any one-way function [HILL99]. Without the abstractions of pseudorandom generators, pseudorandom functions, and target collision resistance, constructing signatures from one-way functions would have been much harder, and perhaps would never have happened.

The last few years have seen an explosion in new cryptographic capabilities, including deniable encryption [SW14], multiparty non-interactive key agreement [GGH13a, BZ14], polynomially many hardcore bits for any one-way function [BST14, Zha16], and much, much more. Many of these new capabilities are the result of two new developments: new candidates for cryptographic multilinear maps [GGH13a, CLT13, GGH15], which in turn have been used to build very powerful program obfuscation [GGH$^+$13b].

We believe that in "Obfustopia" — this new landscape of cutting-edge cryptographic abilities based on obfuscation — abstractions are as important as ever. A new abstraction could offer several advantages:

- **Platform for further applications.** A new tool built from obfuscation can serve as the starting point for further applications. Even if the application was previously known, such an abstraction may illuminate a path that eliminates other building blocks.

- **Conceptual simplification.** Many applications of obfuscation can be quite complicated and have involved security analyses: it is not uncommon for schemes to take multiple pages to describe, and for the security analyses to span dozens of pages. Such complicated schemes and analyses can make it difficult to adapt existing techniques to new applications. The reason for this complication usually stems from the difficulty of proving security using *indistinguishability* obfuscation, the widely-accepted security notion for obfuscation. To make security proofs go through, the programs being obfuscated often have to contain extra computation paths, or additional complicated cryptographic primitives. Then, the security proofs themselves often consist of many steps, where in each step a small change is made to the program.

  Instead, if a new cryptographic tool based on obfuscation can abstract away these complicated techniques, it will provide a simple object that can be easily re-used without reproving everything from scratch.

- **Target for instantiation from other mathematical objects.** The dream of Obfustopia is to construct obfuscation or multilinear maps from more well-understood mathematical tools such as worst-case lattice problems, or problems on elliptic curves. This would yield all applications in Obfustopia secure under well-studied assumptions, as opposed to the current situation where obfuscation and its applications are built on new and tenuous assumptions on multilinear maps. A construction from traditional cryptographic tools would probably also come with efficiency gains.

  We are unfortunately currently very far from this dream goal, as even basing some very basic tasks implied by obfuscation — such as order revealing encryption, broadcast encryption with non-trivial ciphertext size, and multiparty non-interactive key agreement — on worst-case hard lattice problems remains elusive. Instead, progress towards this goal will probably be gradual, developing techniques for certain applications and then generalizing to others.

  A new tool that is simpler and weaker than obfuscation could be an ideal target for instantiation from lattices or other traditional cryptographic objects. Such an instantiation will have better understood security, and could have greatly improved efficiency. Then any application of this tool would inherit the benefits, yielding security improvements.

1.1 Our New Abstraction: Encryptor Combiners

An *encryptor* is a public (and typically randomized) encryption procedure $E(\cdot)$. An encryptor could be the encryption procedure for a public key encryption scheme with the public key hardwired: $E(m) = \mathsf{Enc}(\mathsf{pk}, m)$. However, more general encryptors are possible: E could be the encryption procedure for an *identity-based* encryption scheme for a particular identity ($E(m) = \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, m)$), the procedure to encrypt to a set of users in a broadcast scheme, the procedure to encrypt to a set of attributes in an attributed-based scheme, and so on. An encryptor is secure if $E(m_0)$ is indistinguishable from $E(m_1)$ for any two messages m_0, m_1. A decryptor is a secret (and usually deterministic) decryption procedure that corresponds to the inverse of an encryptor. For example, the decryptor for the public key encryptor $E(m) = \mathsf{Enc}(\mathsf{pk}, m)$ is $D(c) = \mathsf{Dec}(\mathsf{sk}, c)$, whereas the decryptor for the IBE encryptor $E(m) = \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, m)$ is $D(c) = \mathsf{Dec}(\mathsf{sk}_{\mathsf{id}}, c)$.

We define a new concept of an *encryptor combiner* which intuitively allows for combining several different encryptors $E_1, \ldots, E_n$ into a combined encryptor E^*, such that anyone holding a decryptor for *any one* of the E_i can also construct a decryptor for E^*. More specifically (but still informally), an *encryptor combiner* consists of two algorithms $(\mathsf{Combine}, \mathsf{Extract})$ where:

- $\mathsf{Combine}(E_1, \ldots, E_n)$ takes as input a sequence of encryptors, and produces a new encryptor E^*.
- $\mathsf{Extract}(E_1, \ldots, E_n, i, D_i)$ takes as input a sequence of encryptors, and a decryptor for any one of the encryptors. It outputs a combined decryptor D^* that allows for decrypting messages encrypted by E^*.

The security guarantee is that as long as each of the input encryptors $E_1, \ldots, E_n$ are separately secure, the combined encryptor E^* is also secure.
Depending on the application, we may impose specific requirements on encryptor combiners, which we overview here:

- **Common reference string.** As a relaxation, we will consider encryptor combiners in a common *reference* string (crs) model. Here, there is an algorithm Setup that outputs a set of public parameters, which are then additional inputs to $\mathsf{Combine}$ and $\mathsf{Extract}$.
- **Independence.** We may require that every possible valid input decryptor leads to the same combined decryptor D^*. That is, $\mathsf{Extract}(E_1, \ldots, E_n, i, D_i) = \mathsf{Extract}(E_1, \ldots, E_n, j, D_j)$ for any $i \neq j$. We refer to this as perfect independence. We can also consider a weakened version where $\mathsf{Extract}$ is potentially randomized, and while different input decryptors are not guaranteed to give an *identical* combined decrpytor, the distributions of $\mathsf{Extract}(E_1, \ldots, E_n, i, D_i)$ and $\mathsf{Extract}(E_1, \ldots, E_n, j, D_j)$ are the same, or at least statistically close.
- **Restricted encryptor types.** We may restrict the kinds of encryptors that the combiner works on. For example, we could put an a priori bound on the description size of the encryptor. Or we could ask that the input encryptors are of a specific form, say an ElGamal or Regev encryptor.
- **Bounded vs. Unbounded.** In the crs setting, we may allow for n to be an input to Setup, in which case $\mathsf{Combine}$ and $\mathsf{Extract}$ only work for up to n encryptors. We call this a bounded encryptor combiner. In an unbounded combiner, Setup does not take as input n, and $\mathsf{Combine}$ and $\mathsf{Extract}$ work for any number of users.
- **Ciphertext compactness.** We may also require that the ciphertexts produced by the combined decryptor do not grow with the number of input encryptors.

At this point, we note that there is a trivial encryptor combiner construction: the combined encryptor $E^*(m)$ simply outputs $(E_1(m), \ldots, E_n(m))$, and $\mathsf{Extract}(E_1, \ldots, E_n, i, D_i)$ simply outputs D_i^*, which breaks its ciphertext into n components $c_1, \ldots, c_n$, and then decrypts c_i using D_i. This trivial combiner does not require a common reference string or a restriction on the encryptor types, and it works for an unbounded number of input encryptors. However, it has no independence (since D_i^* is clearly dependent on i) and is not compact. For the following discussion, we will usually need either some form of independence or compactness; as such, this trivial combiner will be insufficient.

1.2 Applications

Encryptor combiners have several natural applications.

- **Multiparty non-interactive key exchange (NIKE).** In multiparty NIKE, n users wish to come together and establish a shared secret key, and they wish to do so using minimal rounds of interaction. Specifically, each user separately generates a public value and a secret value. Then, simultaneously, all users publish their public values to a public bulletin board. Next, after everyone publishes, each user reads off the public values from the bulletin board. This is the extent of the interaction: each user, knowing the public contents of the bulletin board and their individual secret value, is able to privately compute a common key K. Meanwhile, an eavesdropper who only sees the contents of the public bulletin board learns no information about K.

  An encryptor combiner, together with public key encryption, gives a simple solution to this task. Each user i generates a secret and public key pair $(\mathsf{sk}_i, \mathsf{pk}_i)$ for a public key encryption scheme, and

publishes the public key, keeping the secret key to themselves. Each public key pk_i gives rise to an encryptor E_i and each secret key sk_i gives a corresponding decryptor D_i in the natural way. Now, user i reads off the public keys $\{\mathsf{pk}_i\}_{i\in[n]}$ from the bulletin board and constructs the respective encryptors $\{E_i\}_{i\in[n]}$. Then, using the decryptor D_i built from sk_i, the user extracts the combined decryptor: $D^* \leftarrow \mathsf{Extract}(E_1, \ldots, E_n, i, D_i)$. The decryptor D^* is the shared secret key[1]. As long as the encryptor combiner satisfies perfect independence, each user will obtain the exact same decryptor D^*.

If the encryptor combiner requires a common reference string, we immediately get a common reference string NIKE protocol; however, we can move the NIKE protocol back to the standard model by designating one of the users to set up the reference string, analogous to the obfuscation-based construction of [BZ14]. If the encryptor combiner works for an unbounded number of users, then we get a NIKE protocol for an unbounded number of users.

- **Broadcast encryption.** In broadcast encryption, a content distributor wishes to broadcast to a set of subscribers. The distributor wishes that only the users currently subscribed can read the broadcast. The trivial solution is to encrypt the message independently to each subscriber. However, this means the ciphertext size grows with the number of users, which is clearly undesirable.

  Again, encryptor combiners offer a natural solution. Start with an identity-based encryption scheme. Each user in the system is assigned an identity id (say, their email address), and receives the secret key $\mathsf{sk}_{\mathsf{id}}$ associated with the identity. To broadcast to a set S of users, construct the encryptors $E_{\mathsf{id}}(\cdot) = \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, \cdot)$ corresponding to each user $\mathsf{id} \in S$. Then combine the encryptors into a combined encryptor $E^*_S \leftarrow \mathsf{Combine}(\{E_{\mathsf{id}}\}_{\mathsf{id}\in S})$. Finally, encrypt the content using E^*_S: $c = E^*_S(m)$. The ciphertext is just c. Each user in S can now decrypt by using their secret key $\mathsf{sk}_{\mathsf{id}}$ to build a decryptor D_{id}, and then use $\mathsf{Extract}$ to compute the combined decryptor D^*_S.

  As long as the encryptor combiner has ciphertext compactness, the broadcast size will be independent of the number of recipients. Moreover, the public and secret keys of the system are also independent of the number of users, since they are just identity-based encryption keys. Thus, we obtain a broadcast scheme with asymptotically optimal parameters.

- **(Hierarchical) Identity-based encryption ((H)IBE).** The above application required an identity-based encryption scheme. We can actually obtain identity-based encryption from encryptor combiners (and public key encryption).

  To set up the scheme for identities of length ℓ, generate 2ℓ public and secret keys for a public key encryption scheme: $(\mathsf{sk}_{i,b}, \mathsf{pk}_{i,b})$ for $i \in [\ell]$, $b \in \{0,1\}$. The master public key consists of all public keys, and the master secret key consists of the secret keys. To encrypt to an identity $\mathsf{id} \in \{0,1\}^\ell$, use the identity to select ℓ of the public keys: $\mathsf{pk}_{i,\mathsf{id}_i}$. These correspond to encryptors E_{i,id_i}. Then combine the ℓ encryptors into a combined encryptor E^*_{id}. Finally, encrypt the message m as $c \leftarrow E^*_{\mathsf{id}}(m)$.

  The secret key for a user id is just the corresponding combined decryptor D^*_{id}, which clearly allows for decrypting the ciphertext. The user secret key can be computed from the master secret key using the $\mathsf{Extract}$ operation. Security against arbitrary collusions can be proved as long as the encryptor combiner satisfies the weaker distributional version of independence.

  By further combining the encryptors E^*_{id}, we can even create a hierarchical identity-based encryption scheme.

1.3 Constructions

We next give two constructions of encryptor combiners. The first is built from universal samplers [HJK+14]. In turn, universal samplers can be built from indistinguishability obfuscation [HJK+14] or from functional encryption [GPSZ16]. As such, this construction currently lives squarely in the realm of Obfustopia. However,

[1] Technically, an eavesdropper can learn the combined encryptor E^*, and therefore learns something about the decryptor: namely, an encryptor that it works for. To remedy this issue, we extract pseudorandom bits using a hardcore predicate.

the construction satisfies most of the wanted properties of an encryptor combiner: perfect independence, unboundedness, unrestricted encryptor types (except for an a priori bound on description size), and ciphertext compactness. Our second construction is a simple combiner for the restricted class of Dual Regev encryptors. It does not satisfy compactness or perfect independence, but does satisfy the weaker notion of distributional independence and is unbounded.

Obfustopia construction. Imagine an oracle that accepts as input arbitrary sampling procedures P. The oracle runs the sampling procedure to obtain a sample s, and then responds with the sample. The key feature of this oracle is that it is actually deterministic: whenever two different users query on the same procedure P, they are guaranteed to get the same sample in return. A universal sampler is a standard-model (but common reference string) version of this concept [HJK+14]. There is a *public deterministic* procedure $\mathsf{Sampler}$ that takes as input a sampling procedure P (as well as the common reference string), and outputs a sample s. Anyone can run $\mathsf{Sampler}$ on P to obtain the sample s. The important property of the universal sampler is that the sample s is "as good as" a fresh random sample from P; for example, if P generates an RSA composite N, nobody should be able to factor N produced by $\mathsf{Sampler}(P)$, even though they generated N for themselves from the sampler.

Universal samplers readily give an encryptor combiner for two encryptors. Given encryptors E_1, E_2, let P_{E_1,E_2} be a sampling procedure that does the following. It generates a fresh secret and public key pair $(\mathsf{sk}, \mathsf{pk})$ for a public key encryption scheme. Let E^*, D^* be the encryptor/decryptor corresponding to these keys. Then P_{E_1,E_2} outputs the sample $\langle E^* \rangle, E_1(\langle D^* \rangle), E_2(\langle D^* \rangle)$. Here we use $\langle D^* \rangle$ and $\langle E^* \rangle$ to denote the descriptions of D^* and E^*. In other words, P_{E_1,E_2} outputs a fresh random encryptor, and then encrypts the corresponding decryptor under both E_1 and E_2.

To combine two encryptors E_1, E_2, simply run $E^*, c_1, c_2 \leftarrow \mathsf{Sampler}(P_{E_1,E_2})$. E^* is the combined encryptor. To obtain the combined decryptor given D_1 or D_2, simply decrypt the appropriate ciphertext c_1 or c_2. Security of this encryptor combiner can be proved under the "static one-time" security notion for universal samplers. Such universal samplers can be built from obfuscation or functional encryption.

The idea above can be readily extended to any number of users by constructing programs $P_{E_1,\ldots,E_n}$ that encrypt D^* under each of the encryptors. However, the number of users will be bounded, since universal samplers have an a priori bound on the size of the samples it can produce[2].

To get an unbounded combiner, we take our cue from the recent work of Garg et al. [GPSZ16]. We note that the encryptor E^* can be again combined with, say, the encryptor E_3, to obtain a new combined encryptor E^{**}. Then E^{**} can be combined with E_4, and so on until all encryptors have been combined. While intuitively this should work, making the proof go through is non-trivial. We follow [GPSZ16], and show how to prove security using a certain pebbling game. However, we depart from their construction and use a different pebbling strategy which uses a binary tree instead of a line. The result is encryptor combiner security based on a *static one-time* universal sampler. Such a universal sampler can be built from obfuscation or functional encryption, as shown in [GPSZ16]. This construction suffices to build all of the applications mentioned above. In addition to providing unified approaches to these tasks, we obtain several "firsts":

- **Multiparty NIKE for unbounded users from static one-time universal samplers.** Garg et al. [GPSZ16] show how to build multiparty NIKE for an unbounded number of users from *k-time* universal samplers *with interactive simulation*, a new notion they define. We instead obtain encryptor combiners, and hence multiparty NIKE, from just one-time static universal samplers, as defined by [HJK+14]. This improvement is due to our new pebbling game. One-time static samplers are a much simpler object, and are potentially more likely to be instantiable from simpler tools. We note, however, that all known approaches to building universal samplers (namely obfuscation and functional encryption) give the stronger variant used by [GPSZ16].

- **Adaptively secure broadcast encryption from FE where all parameters are small.** Previously, the only construction from functional encryption used the multiparty NIKE of [GPSZ16] together

[2]This is not just a limitation of current techniques. The "static one-time" security notion can be shown to require such a limitation

with the NIKE-to-Broadcast conversion of [BZ14]. However, the resulting scheme has a public key that grows with the number of users; this is not the case in our scheme. Moreover, the scheme obtained by [BZ14] is only statically secure, whereas ours is adaptively secure. One trade-off we make is that key generation is no longer distributed: in [BZ14], each user generates their own secret key for themselves, and then contributes a component to the master public key.

- **Adaptively secure broadcast encryption from obfuscation and one-way functions.** By instantiating the universal sampler and public key encryption in the construction with obfuscation and one-way functions, we obtain broadcast encryption from obfuscation and one-way functions where all parameters are small. The only previous small-parameter scheme from obfuscation was due to Zhandry [Zha14]. That scheme is also adaptively secure, but unfortunately requires the extra ingredient of a *somewhere statistically binding hash function*. Such hash functions can be built from number-theoretic primitives, but cannot be built generically from obfuscation and one-way functions.

Dual Regev construction. Next, we show an encryptor combiner for the special class of Dual Regev encryptors. Recall that the public key in Dual Regev encryption is just a random wide matrix A modulo some integer q, and the secret key is a "trapdoor" for this matrix: a "short" full rank (over the integers) matrix T that is orthogonal to A modulo q.

Combining is trivial: to combine the encryptors corresponding to matrices $A_1, \ldots, A_n$, simply concatenate them together as $A^* = [A_1|\cdots|A_n]$; the combined encryptor is just the Dual Regev encryptor corresponding to A^*. Extraction is more interesting: given a trapdoor T_i for A_i, it is possible to construct a trapdoor T^* for A^* using techniques in [CHKP10]. Moreover, it is possible to construct this trapdoor in a randomized way so that the distribution on T^* is independent of i and T_i. As such, our combiner satisfies the weaker distributional independence requirement.

Therefore, this combiner is sufficient for the application to (hierarchical) identity-based encryption ((H)IBE), and security will follow from the Learning With Errors (LWE) assumption. Obtaining (H)IBE from LWE is not new and has been accomplished by prior works [CHKP10, ABB10a, ABB10b], and our construction is even reminiscent of [CHKP10]. The advantage of our approach is in its conceptual unification, bringing constructions from very different tools — lattices and obfuscation — under the common framework of encryptor combiners.

Unfortunately, Dual Regev encryption produces ciphertexts that grow with the width of the matrix. Therefore, our encryptor combiner does not have compact ciphertexts, as the width of A^* grows with the number of encryptors being combined. Our combiner also does not satisfy perfect independence since the generation of T^* is randomized. Therefore our encryptor combiner is not sufficient to build multiparty NIKE or for broadcast encryption from LWE, two major open problems in lattice-based cryptography. However, our encryptor combiner framework can be used to identify the specific features of lattice-based encryption that would be necessary to achieve these applications. If it is possible to modify the Dual Regev encryption to have ciphertexts much shorter than the width of the public key, we would immediately obtain a broadcast scheme with short ciphertexts. Moreover, if it were possible to modify the encryptor combiner to have perfect independence, then we would immediately obtain a multiparty NIKE protocol.

2 Preliminaries

2.1 Security Parameter

Throughout this paper, we hide the security parameter λ, as it will remain constant throughout every proof. It can be assumed that any Setup or KeyGen algorithm takes a security parameter as input, although we will write these algorithms as not taking any input. Additionally, we write *negl* to refer to a function that is negligible in the security parameter.

2.2 Encryptors and Decryptors

We abstract the process of encryption of messages and decryption of ciphertexts into functions that handle these processes.

Definition 1. *An encryptor E is any probabilistic polynomial time (PPT) function that takes a plaintext message as input and outputs a ciphertext.*

Definition 2. *A decryptor D is a PPT function that takes a ciphertext as input and outputs a plaintext message. D is valid for an encryptor E if for all messages m, $Pr[D(E(m))] = m \geq 1 - negl$ for some negligible function $negl$.*

We postpone discussion of what it means for an encryptor to be secure until Section 3.1.

2.3 Public Key Encryption

A public key encryption scheme PK consists of a tuple of PPT algorithms $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ that work as follows:

- $\mathsf{KeyGen}()$ outputs a public/secret key pair $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{Enc}(\mathsf{pk}, m)$ takes as input a public key pk and a message m, and outputs a ciphertext c.
- $\mathsf{Dec}(\mathsf{sk}, c)$ takes as input a secret key sk and a ciphertext c, and outputs a message m.

The scheme is correct if

$$\Pr[\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m)) = m] = 1 \ \forall m, \ \forall (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}().$$

We can easily produce an encryptor/decryptor pair given a public encryption scheme. We let $E(m) = \mathsf{Enc}(\mathsf{pk}, m)$ and $D(c) = \mathsf{Dec}(\mathsf{sk}, c)$ for $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}$. If the public key encryption scheme satisfies correctness, D is clearly a valid decryptor for E.

We define semantic security for public key encryption. For $b = 0, 1$ denote by $\texttt{PK-EXP}(b)$ the following experiment involving an algorithm $\mathcal{A}$:

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}$
$m_0, m_1 \leftarrow \mathcal{A}(\mathsf{pk})$
$c \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)$
$b' \leftarrow \mathcal{A}(c)$

For $b = 0, 1$ let W_b be the event that $b' = 1$ in $\texttt{PK-EXP}(b)$ and define $\texttt{PK-EXP}^{\text{adv}} = |\Pr[W_0] - \Pr[W_1]|$.

Definition 3. *A public key encryption scheme is semantically secure if for all PPT adversaries $\mathcal{A}$, $\texttt{PK-EXP}^{adv}$ is negligible.*

2.4 Identity-Based Encryption

Definition 4. *An identity-based encryption scheme for identity space ID consists of the following PPT algorithms:*

- $\mathsf{Setup}(\mathsf{ID})$ *takes as input an identity space ID and outputs a master public/secret key pair $(\mathsf{mpk}, \mathsf{msk})$.*
- $\mathsf{Extract}(\mathsf{id}, \mathsf{msk})$ *takes as input an identity $\mathsf{id} \in \mathsf{ID}$ and the master secret key msk, and returns D_{id}, a decryptor for identity id.*
- $\mathsf{EncGen}(\mathsf{mpk}, \mathsf{id})$ *takes as input the master public key mpk, an identity $\mathsf{id} \in \mathsf{ID}$, and returns E_{id}, an encryptor for identity id.*

Note that this differs from the standard definitions of identity-based encryption, as this scheme produces encryptors and decryptors. Here, there is no separate decryption algorithm because the decryptor output by $\mathsf{Extract}$ performs decryption.

For correctness, we require that for any id, if $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(\mathsf{ID})$, $D_{\mathsf{id}} \leftarrow \mathsf{Extract}(\mathsf{id}, \mathsf{msk})$, and $E_{\mathsf{id}} \leftarrow \mathsf{EncGen}(\mathsf{mpk}, \mathsf{id})$, then $\Pr[D_{\mathsf{id}}(E_{\mathsf{id}}(m)) = m] = 1$.

We define IND-ID-CPA security for an identity-based encryption scheme via an interactive game. For $b = 0, 1$ denote by $\texttt{IBE-EXP}(b)$ the following experiment involving an adversary $\mathcal{A}$:

$\mathsf{id}^* \leftarrow \mathcal{A}$
$(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(\mathsf{ID})$
$(m_0, m_1) \leftarrow A^{\mathcal{O}_{IBE}}(\mathsf{mpk})$
$E_{\mathsf{id}^*} \leftarrow \mathsf{EncGen}(\mathsf{mpk}, \mathsf{id}^*)$
$b' \leftarrow A^{\mathcal{O}_{IBE}}(E_{\mathsf{id}^*}(m_b))$.

where $\mathcal{O}_{IBE}$ is a identity decryptor oracle which takes as input $\mathsf{id} \neq \mathsf{id}^*$, and returns $D_{\mathsf{id}} \leftarrow \mathsf{Extract}(\mathsf{msk}, \mathsf{id})$. $A^{\mathcal{O}_{IBE}}$ denotes an adversary A with oracle access to $\mathcal{O}_{IBE}$.

For $b = 0, 1$ let W_b be the event that $b' = 1$ in $\texttt{IBE-EXP}(b)$ and define $\texttt{IBE-EXP}^{\text{adv}} = |\Pr[W_0] - \Pr[W_1]|$.

Definition 5. *An identity-based encryption scheme is IND-ID-CPA secure if* $\texttt{IBE-EXP}^{adv}$ *is negligible for all PPT adversaries* $\mathcal{A}$.

We can easily extend this definition to cover adaptive security. We simply change the experiment so that the IND-ID-CPA adversary $\mathcal{A}$ selects id^* when committing to the messages m_0, m_1, instead of at the beginning.

3 Encryptor Combiners

3.1 Public Encryptor Combiners

A public encryptor combiner takes encryptors as input and outputs a combined encryptor. Any user with access to a decryptor for one of the input users can compute a decryptor for the combined encryptor. We will frequently drop the word "public" and just refer to these as encryptor combiners.

Definition 6. *An encryptor combiner consists of the following three PPT algorithms:*

- $\mathsf{Setup}()$ *outputs public parameters* params.
- $\mathsf{Combine}(\mathsf{params}, \{E_i\}_{i\in[n]})$ *takes as input* params *and a set of* n *encryptors* $\{E_i\}_{i\in[n]}$ *and outputs a combined encryptor* E.
- $\mathsf{Extract}(\mathsf{params}, \{E_i\}_{i\in[n]}, j, D_j)$ *takes as input* params*, a set of* n *encryptors* $\{E_i\}_{i\in[n]}$*, an index* $j \in [n]$*, and the decryptor* D_j*. It outputs a combined decryptor* D.

For correctness, we require that $D \leftarrow \mathsf{Extract}(\mathsf{params}, \{E_i\}_{i\in[n]}, j, D_j)$ is a valid decryptor for E with probability $1 - negl$ for some negligible function $negl$.

For most applications, we will require encryptor combiners to satisfy some notion of independence, which informally means that it should not matter which D_j is used to compute the combined encryptor. We define two different notions of independence.

Definition 7. *An encryptor combiner satisfies perfect independence if* $\forall j, k$*, and all valid decryptors* D_j, D_k *for* E_j, E_k*,*

$$Pr[\mathsf{Extract}(\mathsf{params}, \{E_i\}_{i\in[n]}, j, D_j) = \mathsf{Extract}(\mathsf{params}, \{E_i\}_{i\in[n]}, k, D_k)] \geq 1 - negl$$

for some negligible function $negl$.

Definition 8. *Let* Dist_{D_j} *be the distribution on* $D^* \leftarrow \mathsf{Extract}(\mathsf{params}, \{E_i\}_{i\in[n]}, j, D_j)$*. An encryptor combiner satisfies distributional independence if* $\forall j, k$*, and all valid decryptors* D_j, D_k *for* E_j, E_k

$$\mathsf{Dist}_{D_j} \approx_S \mathsf{Dist}_{D_k},$$

where $\approx_S$ *denotes that the statistical difference between two distributions is negligible.*

Note that perfect independence trivially implies distributional independence.

Definition 9. *We say that an encryptor combiner satisfies compactness if the size of the ciphertext produced by the combined encryptor is independent of the number* n *of encryptors.*

Finally, we note that the above notion allows one params to be used for any (polynomial) number of input encryptors n. We can also consider a variant where Setup takes n as input. Now, Setup may run in time polynomial in n, the size of params may depend polynomially on n, and $\mathsf{Combine}$ only accepts up to n encryptors. We call this variant a *bounded* encryptor combiner. When distriguishing from a bounded combiner, we call the notion above without a restriction on n an *unbounded* combiner.

Security. Now we consider what it means for an encryptor to be secure. Let Dist be a distribution on triples $(E, \mathsf{aux}, \mathcal{O})$ where E is an encryptor, aux is some auxiliary information, and $\mathcal{O}$ is an oracle. More precisely, $\mathcal{O}$ is a potentially randomized interactive and stateful Turing machine. $\mathcal{O}$ may have some embedded secrets, and can answer certain queries for the adversary that will depend on the application. We stress that the adversary does not get the code for $\mathcal{O}$, but can only interact with it via queries. We will often hide the oracle $\mathcal{O}$ when it is unnecessary. For $b = 0, 1$ denote by $\texttt{EC-EXP}(b)$ the following experiment:

$(E, \mathsf{aux}, \mathcal{O}) \leftarrow \mathsf{Dist}$
$m_0, m_1 \leftarrow \mathcal{A}^{\mathcal{O}}(E, \mathsf{aux})$
$b' \leftarrow \mathcal{A}^{\mathcal{O}}(E(m_b), E, \mathsf{aux})$

Here $\mathcal{A}^{\mathcal{O}}$ denotes that $\mathcal{A}$ may make polynomially many adaptive calls to $\mathcal{O}$.
For $b = 0, 1$ let W_b be the event that $b' = 1$ in $\texttt{EC-EXP}(b)$ and define $\texttt{EC-EXP}^{\mathrm{adv}} = |\Pr[W_0] - \Pr[W_1]|$.

Definition 10. Dist *is a secure distribution on encryptors if for all PPT adversaries* $\mathcal{A}$, $\texttt{EC-EXP}^{adv}$ *is negligible.*

Now consider a distribution $\mathsf{Dist} = (\{E_i\}_{i\in[n]}, \mathsf{aux}, \mathcal{O})$ on n separate encryptors, auxiliary information aux, and an oracle $\mathcal{O}$. For $i \in [n]$, Dist_i will be a distribution derived from Dist, that outputs samples of the form

$$(E_i, \mathsf{aux}_i = \{E_1, \ldots, E_{i-1}, E_{i+1}, \ldots, E_n, \mathsf{aux}\}, \mathcal{O}) \leftarrow \mathsf{Dist}_i.$$

Let Dist^* be the distribution that outputs samples

$$(E^*, \mathsf{aux}^* = \{E_1, \ldots, E_n, \mathsf{aux}, \mathsf{params}\}, \mathcal{O}) \leftarrow \mathsf{Dist}^*,$$

where $\mathsf{params} \leftarrow \mathsf{Setup}()$ and $E^* \leftarrow \mathsf{Combine}(\mathsf{params}, E_1, \ldots, E_n)$.

Definition 11. *An encryptor combiner is statically secure if* Dist^* *is a secure distribution for any* Dist *where* Dist_i *are secure distributions for all* $i \in [n]$*.*

We also consider a stronger, adaptive notion of security for encryptor combiners. For $b = 0, 1$ denote by $\texttt{ADAPT-EC-EXP}(b)$ the following experiment:

$(E_1, \ldots, E_n, \mathsf{aux}, \mathcal{O}) \leftarrow \mathsf{Dist}$
$\mathsf{params} \leftarrow \mathsf{Setup}$
$S, m_0, m_1 \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{params}, E_1, \ldots, E_n, \mathsf{aux})$ where $S \subseteq [n]$
$E^* \leftarrow \mathsf{Combine}(\mathsf{params}, \{E_i\}_{i\in S})$
$b' \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{params}, E^*_S(m_b), \mathsf{aux})$

For $b = 0, 1$ let W_b be the event that $b' = 1$ in $\texttt{ADAPT-EC-EXP}(b)$ and define $\texttt{ADAPT-EC-EXP}^{\text{adv}} = |\Pr[W_0] - \Pr[W_1]|$.

Definition 12. *A public encryptor combiner is adaptively secure if for all PPT adversaries $\mathcal{A}$, $\texttt{ADAPT-EC-EXP}^{adv}$ is negligible.*

3.2 Private Encryptor Combiners

A private encryptor combiner takes encryptors as input, but does not output a combined encryptor. Instead, any user with a decryptor for an input encryptor can use the private combiner to generate a shared key. Note that private encryptor combiners will always be referred to as such, to avoid confusion with regular (public) encryptor combiners.

Definition 13. *A private encryptor combiner consists of the following two PPT algorithms:*

- $\mathsf{Setup}()$ *outputs public parameters* params.
- $\mathsf{Extract}(\mathsf{params}, \{E_i\}_{i\in[n]}, j, D_j)$ *takes as input* params *and a set of n encryptors $\{E_i\}_{i\in[n]}$ and outputs a shared key $K \leftarrow \mathsf{Extract}(\mathsf{params}, \{E_i\}_{i\in[n]}, j, D_j)$.*

The correctness requirement is that each user should generate the same shared key. For all $j, k \in [n]$,

$$Pr[\mathsf{Extract}(\mathsf{params}, \{E_i\}_{i\in[n]}, j, D_j) = \mathsf{Extract}(\mathsf{params}, \{E_i\}_{i\in[n]}, k, D_k)] \geq 1 - negl,$$

for $\mathsf{params} \leftarrow \mathsf{Setup}()$ and for some negligible function $negl$.

Security. We first define static security. Let Dist be a distribution that produces samples of the form $(\{E_i\}_{i\in[n]}, \mathsf{aux}, \mathcal{O})$ for n separate encryptors E_i and auxiliary information aux, and an oracle $\mathcal{O}$. For $i \in [n]$, Dist_i is derived from Dist, and outputs samples of the form

$$(E_i, \mathsf{aux}_i = \{E_1, \ldots, E_{i-1}, E_{i+1}, \ldots, E_n, \mathsf{aux}\}) \leftarrow \mathsf{Dist}_i.$$

Now consider the following experiment $\texttt{PEC-EXP}(b)$ for $b = 0, 1$:

$(E_1, \ldots, E_n, \mathsf{aux}, \mathcal{O}) \leftarrow \mathsf{Dist}$
$\mathsf{params} \leftarrow \mathsf{Setup}()$
$m_0 \xleftarrow{R} \mathcal{K}, m_1 \leftarrow \mathsf{Extract}(\mathsf{params}, \{E_i\}_{i\in[n]}, j, D_j)$
$b' \leftarrow \mathcal{A}^{\mathcal{O}}(\{E_i\}_{i\in[n]}, \mathsf{aux}, \mathsf{params}, m_b)$

For $b = 0, 1$ let W_b be the event that $b' = 1$ in $\texttt{PEC-EXP}(b)$ and define $\texttt{PEC-EXP}^{\text{adv}} = |\Pr[W_0] - \Pr[W_1]|$.

Definition 14. *A private encryptor combiner is statically secure if for all distributions* Dist *where* Dist_i *is secure for all $i \in [n]$, $\texttt{PEC-EXP}^{adv}$ is negligible for all PPT adversaries $\mathcal{A}$.*

Adaptive security can be defined by modifying the experiment so that the adversary may pick a subset $S \subseteq [n]$ to be challenged on after seeing the private encryptor combiner parameters. Formally, let $\texttt{ADAPT-PEC-EXP}(b)$ for $b = 0, 1$ be the following:

$(E_1, \ldots, E_n, \mathsf{aux}, \mathcal{O}) \leftarrow \mathsf{Dist}$
$\mathsf{params} \leftarrow \mathsf{Setup}()$
$S \leftarrow \mathcal{A}^{\mathcal{O}}(\{E_i\}_{i\in[n]}, \mathsf{aux}, \mathsf{params})$, where $S \subseteq [n]$
$m_0 \xleftarrow{R} \mathcal{K}, m_1 \leftarrow \mathsf{Extract}(\mathsf{params}, \{E_i\}_{i\in S}, j \in S, D_j)$
$b' \leftarrow \mathcal{A}^{\mathcal{O}}(\{E_i\}_{i\in[n]}, \mathsf{aux}, \mathsf{params}, m_b)$

For $b = 0, 1$ let W_b be the event that $b' = 1$ in $\texttt{ADAPT-PEC-EXP}(b)$ and define $\texttt{ADAPT-PEC-EXP}^{\text{adv}} = |\Pr[W_0] - \Pr[W_1]|$.

Definition 15. *A private encryptor combiner is adaptively secure if for all distributions* Dist *where* Dist_i *is secure for all $i \in [n]$, $\texttt{ADAPT-PEC-EXP}^{adv}$ is negligible for all PPT adversaries $\mathcal{A}$.*

3.3 Public Encryptor Combiners from Universal Samplers

3.3.1 Universal Samplers

We review the definition of a universal sampler, introduced by Hofheinz et al. [HJK+14]. A universal sampler is an object that takes as input a circuit C (i.e. a description of C), and outputs a fresh-looking sample from C. The sampler is public and will always generate the same output for a given input.

Definition 16. *A universal sampler consists of the following two PPT algorithms:*

- Setup *outputs parameters* params.
- $\mathsf{Sampler}(\mathsf{params}, C)$ *takes as input sampler parameters* params *and a circuit description* C*, and outputs* p_C*, a sample from the circuit.*

Definition 17. *A universal sampler* $(\mathsf{Setup}, \mathsf{Sampler})$ *is one-time statically secure if there exists an efficient algorithm* Sim *such that*

- *There exists a negligible function negl such that for all circuits* C *of length* ℓ *taking* m *bits of input and outputting* k *bits, and for all strings* $p_C \in \{0,1\}^k$*,*

$$\Pr[\mathsf{Sampler}(\mathsf{Sim}(C, p_C), C) = p_C] \geq 1 - negl$$

- *For any PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, define the following two experiments.* $\texttt{US-EXP}(0)$ *is defined as:*

  $(C^*, \mathsf{state}) \leftarrow \mathcal{A}_1$
  Output $b' = \mathcal{A}_2(\mathsf{Setup}(), \mathsf{state})$.

  and $\texttt{US-EXP}(1)$ *is defined as:*

  $(C^*, \mathsf{state}) \leftarrow \mathcal{A}_1$
  Choose r *uniformly from* $\{0,1\}^m$
  Let $p_C = C^*(r)$*. Output* $b' = \mathcal{A}_2(\mathsf{Sim}(C^*, p_C), \mathsf{state})$

  For $b = 0, 1$ *let* W_b *be the event that* $b' = 1$ *in experiment* $\texttt{US-EXP}(b)$ *and define* $\texttt{PK-EXP}^{adv} = |Pr[W_0] - Pr[W_1]|$*. Then we require that*

$$\texttt{PK-EXP}^{adv} \leq negl$$

Intuitively, this security definition states that the parameters generated by the simulator Sim should be indistinguishable from the parameters of an honestly generated sampler, where the simulated sampler, on input C, outputs $p_C = C^*(r)$ for true randomness r.

3.3.2 Encryptor Combiner Construction

We construct public encryptor combiners from one-time statically secure universal samplers and public key encryption. Intuitively, we imagine a binary tree where each leaf node (level 0) corresponds to an input encryptor E_i. At each internal node of the tree with two children nodes E_1 and E_2, we use a universal sampler to output $(E_{12}, E_1(D_{12}), E_2(D_{12}))$. Encryptor E_{12} becomes the encryptor associated with this internal node, and we use the process to recursively define every internal node. At each node at level l of the tree that is the left child of its parent, we use universal sampler parameters params^l_{left}, and each node that is the right child of its parent, we use $\mathsf{params}^l_{right}$.

We give the following construction for public encryptor combiners using one-time statically secure universal samplers and public key encryption.

- $\mathsf{Setup}()$ first runs the universal sampler setup algorithm 2λ times to generate $\{\mathsf{params}^l_{left}, \mathsf{params}^l_{right}\}_{0<l\leq\lambda}$, and outputs all of the sampler parameters: $\mathsf{params} = \{\mathsf{params}^l_{left}, \mathsf{params}^l_{right}\}_{0<l\leq\lambda}$.

- $\mathsf{Combine}(\mathsf{params}, \{E_i\}_{i\in[n]})$ takes $\mathsf{params} = \{\mathsf{params}^l_{left}, \mathsf{params}^l_{right}\}_{0<l\leq\lambda}$ and a set of n encryptors $\{E_i\}_{i\in[n]}$, where $n \leq 2^\lambda$. Relabel these encryptors as $\{\mathsf{E}^0_i\}_{i\in[n]}$ to signify level 0. Define $C_{E,E'}$ to be the circuit that generates a new encryptor/decryptor pair E'', D'' according to $\mathsf{PK.Gen}$, and outputs $E'', E(D''), E'(D'')$. For each level $0 < l \leq \lceil \log_2 n \rceil$, it sets $(E^l_i, E^{l-1}_{2i-1}(D^l_i), E^{l-1}_{2i}(D^l_i)) = \mathsf{Sampler}(\mathsf{params}^l_{left}, C_{E^{l-1}_{2i-1}, E^{l-1}_{2i}})$ if i is odd and sets $(E^l_i, E^{l-1}_{2i-1}(D^l_i), E^{l-1}_{2i}(D^l_i)) = \mathsf{Sampler}(\mathsf{params}^l_{right}, C_{E^{l-1}_{2i-1}, E^{l-1}_{2i}})$ if i is even. If E^{l-1}_{2i} does not exist, we repeat E^{l-1}_{2i-1} to hold its place. Output $E^{\lceil \log_2 n \rceil}_1$.

- $\mathsf{Extract}(\mathsf{params}, \{E_i\}_{i\in[n]}, l, D - l)$ takes as input params, a set of n encryptors $\{E_i\}_{i\in[n]}$, a user index l, and a decryptor D_l. Note that given D_l, which we relabel as D^0_l, we can compute the "ancestor" decryptor $D^1_{\lceil l/2 \rceil}$. We repeat this process until we have $D^{\lceil \log_2 n \rceil}_1$, which we output.

Theorem 1. *If* Setup *gives a one-time statically secure universal sampler, the above construction gives a secure unbounded encryptor combiner with perfect independence*

Proof. The fact that the combiner has perfect independence is straightforward. Syntactically, the combiner works for any $n \leq 2^\lambda$; taking λ to grow with the security parameter, we can handle any polynomial n.

We now prove security. Let Dist be a distribution on $(\{E_i\}_{i\in[n]}, \mathsf{aux}, \mathcal{O})$ for a polynomial n, Dist_i be the derived distributions $(E_i, \mathsf{aux}_i = (\mathsf{aux}, \{E_j\}_{j\neq i}), \mathcal{O})$, and let Dist^* be

$$(E^*, \mathsf{aux}^* = \{E_1, \ldots, E_n, \mathsf{aux}, \mathsf{params}\}, \mathcal{O}) \leftarrow \mathsf{Dist}^*,$$

where $\mathsf{params} \leftarrow \mathsf{Setup}()$ and $E^* \leftarrow \mathsf{Combine}(\mathsf{params}, E_1, \ldots, E_n)$. Suppose Dist_i is a secure distribution on encryptors for each i. Our goal is to prove that Dist^* is a secure distribution on encryptor E^*.

Let $\mathcal{A}$ be a hypothetical adversary for Dist^*. We bound the advantage of $\mathcal{A}$ using a collection of hybrids. We say that a set T is *legal* if it consists of index/level pairs (i, l) such that for any level l, there is at most one element (i, l) where i is odd, and at most one element (i, l) where i is even. Any legal T corresponds to the constraint that at each level l, we can program params^l_{left} and $\mathsf{params}^l_{right}$ at at most one point each.

Let **Hybrid** T be as follows. Sample $(\{E_i\}_{i\in[n]}, \mathsf{aux}, \mathcal{O})$ from Dist. Define $E^0_i = E_i$ for $i = 1, n$. Then, for $l = 1, \ldots, \lceil \log_2 n \rceil$ do the following. If there is an $(i, l) \in T$ with i odd, then choose a fresh encryptor/decryptor pair E^l_i, D^l_i and set $\mathsf{params}^l_{left} \leftarrow \mathsf{Sim}(C_{E^{l-1}_{2i-1}, E^{l-1}_{2i}}, (E^l_i, E^{l-1}_{2i-1}(0), E^{l-1}_{2i}(0)))$. Otherwise, set $\mathsf{params}^l_{left} \leftarrow \mathsf{Setup}()$. We similarly handle even i: if there is an $(i, l) \in T$ with i even, choose a fresh encryptor/decryptor pair E^l_i, D^l_i and set $\mathsf{params}^l_{right} \leftarrow \mathsf{Sim}(C_{E^{l-1}_{2i-1}, E^{l-1}_{2i}}, (E^l_i, E^{l-1}_{2i-1}(0), E^{l-1}_{2i}(0)))$. Otherwise, set $\mathsf{params}^l_{right} \leftarrow \mathsf{Setup}()$. For all i for which there does not exist an $(i, l) \in T$, set E^l_i by running $(E^l_i, c_0, c_1) \leftarrow \mathsf{Sampler}(C_{E^{l-1}_{2i-1}, E^{l-1}_{2i}})$. If at some point E^{l-1}_{2i} does not exist, we repeat E^{l-1}_{2i-1} to hold its place. Notice that E^* is defined to be $E^{\lceil \log_2 n \rceil}_1$.

For $l = \lceil \log_2 n \rceil + 1, \ldots, \lambda$, set $\mathsf{params}^l_{left} \leftarrow \mathsf{Setup}()$ and $\mathsf{params}^l_{right} \leftarrow \mathsf{Setup}()$.

If $T = \emptyset$, then params is generated honestly and this situation is equivalent to the honest encryptor combiner construction. When T contains $(1, \lceil \log_2 n \rceil)$, all information about $D^* = D^{\lceil \log_2 n \rceil}_1$ is information theoretically independent of the adversary's view, since it is replaced by encodings of 0. Thus if T contains $(1, \lceil \log_2 n \rceil)$, public key encryption security guarantees security of the combined encryptor. Our goal is to get from $T = \emptyset$ to $T \ni (1, \lceil \log_2 n \rceil)$. We use the following lemma to accomplish this:

Lemma 1. *Hybrid* T *and Hybrid* T' *are indistinguishable if* $T' = T \cup \{(i, l)\}$ *where* $(2i-1, l-1), (2i, l-1) \in T$, *or* $l = 1$.

We can think of T as containing a set of pebbles on a binary tree with n leaves. All the leaves always have pebbles, corresponding to the leaf encryptors being secure. A pebble at an internal node x corresponds to programming the universal sampler so that the encryptor E_x at that node is freshly random, and the

params is independent of the decryptor. Lemma 1 says that, for any node x in the tree, if we currently have pebbles on both of x's children (so both children's encryptors are fresh and no information is revealed about their decryptors), then we can add or remove a pebble from x. The fact that T, T' are legal means that for every level in the tree (except the leaves), there is at most one pebble on a right child, and at most one pebble on a left child.

Given this lemma, it remains to show that there exists a sequence of hybrids for sets $T_0, \ldots, T_t$ such that each T_i is legal, $T_0 = \emptyset$, $T_t = \{(1, \lceil \log_2 n \rceil)\}$, and T_j and T_{j+1} differ at a single element (i_j, l_j) such that either $(2i_j - 1, l_j - 1), (2i_j, l_j - 1) \in T_j \cap T_{j+1}$ or $l_j - 1 = 0$. We prove this via induction on the total number of levels. Suppose that the total number of levels is 1. In this case, we let $T_0 = \emptyset$ and $T_t = (1, 1)$. For the inductive step, suppose there is a sequence $T_0, \ldots, T_{t_k}$ that works for k levels. We apply these steps to the first $2^{\lceil \log_2 n \rceil - 1}$ level 0 encryptors to reach the set $\{1, k\}$, and then to the remaining level 0 encryptors to reach $\{(1, k), (2, k)\}$. From here, we can reach $\{(1, k+1), (1, k), (2, k)\}$, and then reverse the procedure to get to $\{(1, k+1)\}$.

Put another way, to get a pebble on a node x, we inductively get a pebble on its left child. While during the inductive step there may be pebbles on every level below x, after the inductive step is completed there is exactly one pebble on the level below x (on its left child), and no pebbles on subsequent levels except for the leaves. Therefore, we can inductively also get a pebble on the right child. At this point, we can place a pebble on x. We are not quite finished, as we need to remove the pebbles from the children of x. This is easily done via two more recursive calls on the children of x. In the induction, 4 calls are made to a problem of half the size, so solving the recurrence gives a total number of steps $O(n^2)$, a polynomial.

To complete the proof, we prove Lemma 1.

WLOG, suppose i is odd. We claim that if $(2i-1, l-1), (2i, l-1) \in T$ or $l = 1$, then it does not matter whether or not params^l_{left} is generated by $\mathsf{Sim}(C_{E^{l-1}_{2i-1}, E^{l-1}_{2i}}, (E^l_i, E^{l-1}_{2i-1}(0), E^{l-1}_{2i}(0)))$ or by Setup.

In **Hybrid** T, params^l_{left} is generated by Setup. Consider an intermediate **Hybrid** T^1 that is identical to T except that params^l_{left} is now generated by $\mathsf{Sim}(C_{E^{l-1}_{2i-1}, E^{l-1}_{2i}}, (E^l_i, E^{l-1}_{2i-1}(D^l_i), E^{l-1}_{2i}(D^l_i)))$. If there exists an adversary that can distinguish **Hybrid** T^1 from **Hybrid** T, we can use it to break the one-time static security of the universal sampler.

We introduce another intermediate **Hybrid** T^2 where params^l_{left} is generated by $\mathsf{Sim}(C_{E^{l-1}_{2i-1}, E^{l-1}_{2i}}, (E^l_i, E^{l-1}_{2i-1}(0), E^{l-1}_{2i}(D^l_i)))$. If an adversary can distinguish **Hybrid** T^1 from **Hybrid** T^2, then the adversary can break the security of E^{l-1}_{2i-1}, since D^{l-1}_{2i-1} is not needed to simulate either hybrid.

Finally, **Hybrid** T^2 is indistinguishable from **Hybrid** T', where params^l_{left} is generated by $\mathsf{Sim}(C_{E^{l-1}_{2i-1}, E^{l-1}_{2i}}, (E^l_i, E^{l-1}_{2i-1}(0), E^{l-1}_{2i}(0)))$ since an adversary that can distinguish the two can break the security of E^{l-1}_{2i} (since D^{l-1}_{2i} is not needed to simulate either hybrid). □

3.4 Private Encryptor Combiners from Universal Samplers

The construction in Section 3.3 can be modified to give a private encryptor combiner. We define $D_{E,E'}$ to be a circuit that generates an encryptor/decryptor pair E'', D'' according to PK, and outputs $0, E(D''), E'(D'')$. $\mathsf{Combine}$ is modified so that when $j = n$, $D_{E,E'}$ is used in place of $C_{E,E'}$, and thus nothing is output.

If a direct construction of a private encryptor combiner is desired, this construction may be the simplest. Section 3.6 will focus on how to construct a private encryptor combiner starting from a public encryptor combiner.

3.5 Trapdoored Encryptor Combiner Variants

For some applications, it will be convenient to have an encryptor combiner with a trapdoor. We define a trapdoored encryptor combiner to be an encryptor combiner with the following modifications: 1) Setup outputs a trapdoor alongside params. 2) There is an additional PPT algorithm, $\mathsf{TrapdoorExtract}$, which produces the same output as $\mathsf{Extract}$ but takes the trapdoor as input instead of an individual user's valid decryptor.

We define correctness analogous to independence. For perfect correctness, let D be the trapdoor generated by Setup. We require that for all $j \in [n]$,

$$Pr[\mathsf{TrapdoorExtract}(\mathsf{params}, \{E_i\}_{i\in[n]}, D) = \mathsf{Extract}(\mathsf{params}, \{E_i\}_{i\in[n]}, j, D_j)] \geq 1 - negl,$$

for $D, \mathsf{params} \leftarrow \mathsf{Setup}()$ and for some negligible function $negl$. Distributional correctness is defined so that the distribution of $\mathsf{TrapdoorExtract}$ is statistically close to the distribution of $\mathsf{Extract}$. The same security definitions for regular encryptor combiners apply here.

We construct a trapdoored public encryptor combiner consisting of $(\mathsf{Setup}', \mathsf{Combine}', \mathsf{Extract}', \mathsf{TrapdoorExtract})$, from a public encryptor combiner consisting of $(\mathsf{Setup}, \mathsf{Combine}, \mathsf{Extract})$ as follows.

- $\mathsf{Setup}'()$ outputs $(\ D, \mathsf{params}' = (\mathsf{params}, E)\ )$, where $\mathsf{params} \leftarrow \mathsf{Setup}()$, and E, D correspond to $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PK.KeyGen}()$. The trapdoor is D.
- $\mathsf{Combine}'(\mathsf{params}', \{E_i\}_{i\in[n]})$ takes as input $\mathsf{params}' = (\mathsf{params}, E)$ and a set of n encryptors $\{E_i\}_{i\in[n]}$ and outputs $\mathsf{Combine}(\mathsf{params}, \{E_i\}_{i\in[n]} \cup \{E\})$.
- $\mathsf{Extract}'(\mathsf{params}', \{E_i\}_{i\in[n]}, i, D_i)$ takes as input $\mathsf{params}' = (\mathsf{params}, E)$, a set of n encryptors $\{E_i\}_{i\in[n]}$, a user index $i \in [n]$, decryptor D_i, and outputs $\mathsf{Extract}(\mathsf{params}, \{E_i\}_{i\in[n]} \cup \{E\}, i, D_i)$.
- $\mathsf{TrapdoorExtract}(\mathsf{params}', \{E_i\}_{i\in[n]}, D)$ takes as input $\mathsf{params}' = (\mathsf{params}, E)$, a set of n encryptors $\{E_i\}_{i\in[n]}$, a trapdoor D. It then runs and outputs $\mathsf{Extract}(\mathsf{params}, \{E_i\}_{i\in[n]} \cup \{E\}, n+1, D)$.

The same technique works for adding a trapdoor to a private encryptor combiner, so we skip an explicit construction.

3.6 Private Encryptor Combiners from Public Encryptor Combiners

Suppose we have a public encryptor combiner consisting of $(\mathsf{Setup}, \mathsf{Combine}, \mathsf{Extract})$. As long as this combiner satisfies perfect independence, We use this to construct a private encryptor combiner with one-bit keys as follows.

- $\mathsf{Setup}^{(1)}()$ outputs $\mathsf{params}^{(1)} \leftarrow (\mathsf{params}, r)$, where $\mathsf{params} \leftarrow \mathsf{Setup}()$ and r is a random vector in $\mathbb{F}_2^n$.
- $\mathsf{Extract}^{(1)}(\mathsf{params}^{(1)}, \{E_i\}_{i\in[n]}, j, D_j)$ takes as input $\mathsf{params}^{(1)}$, a set of n encryptors $\{E_i\}_{i\in[n]}$, a user index $j \in [n]$, and decryptor D_j. Set $D \leftarrow \mathsf{Extract}(\mathsf{params}, \{E_i\}_{i\in[n]}, j, D_j)$. Interpret D as a vector in $\mathbb{F}_2^n$, and return $b \leftarrow \langle D, r\rangle$.

Theorem 2. *The above construction gives a statically secure private encryptor combiner with one-bit keys if the public combiner has perfect independence.*

Proof Sketch. b is a Goldreich-Levin hardcore bit that is pseudorandom for any unpredictable source. The security of the public encryptor combiner implies that the source, D, is unpredictable. □

To get a multi-bit private encryptor combiner, we have to work a little harder. It might be tempting to just run multiple one-bit instances in parallel, but unfortunately it is not clear how to prove security. The problem is that the proof would use several intermediate hybrids, where in each hybrid we need to give the adversary some of the correct combined key bits. However, the reduction will not have the ability to generate those bits. Instead, we can add a trapdoor to the one-bit combiner as described in Section 3.5. Let the trapdoored scheme consist of $\mathsf{Setup}^{(2)}, \mathsf{Extract}^{(2)}$ and $\mathsf{TrapdoorExtract}^{(2)}$. Run m instances of the trapdoored scheme in parallel to construct a private encryptor combiner with m-bit keys.

- $\mathsf{Setup}^{(3)}()$ outputs $\mathsf{params}^{(3)} = \{\mathsf{params}_l\}_{l\in[m]}$ where $\mathsf{params}_l \leftarrow \mathsf{Setup}_{\mathsf{l}}^{(2)}()$ for $l \in [m]$.
- $\mathsf{Extract}^{(3)}(\mathsf{params}^{(3)}, \{E_i\}_{i\in[n]}, j, D_j))$ takes as input $\mathsf{params}^{(3)}$, a set of n encryptors $\{E_i\}_{i\in[n]}$, a user index $j \in [n]$, and decryptor D_j. It outputs the concatenation of $\{\mathsf{Extract}^{(2)}(\mathsf{params}_k, \{E_i\}_{i\in[n]}, j, D_j)\}_{k\in[m]}$.

Theorem 3. *The above construction gives a statically secure private encryptor combiner with m-bit keys.*

Proof. We prove this with a sequence of hybrids. For $l \in \{0, 1, \ldots, m\}$, define $\mathbf{Hybrid}_l$ as follows.

$\mathsf{params}^{(3)} \leftarrow \mathsf{Setup}^{(3)}()$
$(E_i, \mathsf{aux}) \leftarrow \mathsf{Dist}_\mathsf{i} \forall i \in [n]$
key_l has the first l bits set to $\{\mathsf{Extract}^{(2)}(\mathsf{params}_k, \{E_i\}_{i\in[n]}, j, D_j)\}_{k\in[l]}$
and the remaining $m - l$ bits randomly generated.
$b' \leftarrow \mathcal{A}(\mathsf{params}, \mathsf{key}_l, \{E_i\}_{i\in[n]}, \mathsf{aux})$

$\mathbf{Hybrid}_0$ and $\mathbf{Hybrid}_m$ are identical to $\texttt{PEC-EXP}(0)$ and $\texttt{PEC-EXP}(1)$, respectively. It remains to show that no adversary $\mathcal{A}$ can distinguish between $\mathbf{Hybrid}_{k-1}$ and $\mathbf{Hybrid}_k$ for any $k \in [m]$.

Suppose there exists an adversary $\mathcal{A}$ that can distinguish between the two for some k. We construct an adversary $\mathcal{B}$ that breaks the security of the one-bit trapdoor private encryptor combiner. $\mathcal{B}$ receives $(\mathsf{params}_k, \{E_i\}_{i\in[n]}, \mathsf{aux})$ from the challenger, and then runs $\mathcal{A}$, playing the role of its challenger. First, $\mathcal{B}$ generates $\mathsf{params}^{(3)}$ by generating params_l for all $l \neq k$ for one-bit combiners on his own using $\mathsf{Setup}()$. Then $\mathcal{B}$ constructs a key K_b by concatenating $k-1$ random bits, the random bit received from the challenger, and the remaining $m - k$ honest bits computed from $\mathsf{TrapdoorExtract}^{(2)}$. $\mathcal{B}$ sends $\mathsf{params}^{(3)}, K_b, \{E_i\}_{i\in[n]}, \mathsf{aux}$ to $\mathcal{A}$. $\mathcal{A}$ guesses $b = 0$ or 1, corresponding to $\mathbf{Hybrid}_k$ and $\mathbf{Hybrid}_{k-1}$, respectively. Thus, $\mathcal{B}$ forwards $\mathcal{A}'s$ response to the challenger, and attains a non-negligible advantage in distinguishing the hybrids.

□

4 Hierarchical Identity-Based Encryption

4.1 Identity-Based Encryption

We give a construction for identity-based encryption via public encryptor combiners and admissible/collision-resistant hash functions. In Section 4.2, we show how to extend this to hierarchical IBE. We first review the definitions of these hash function families.

Definition 18. *A hash function family* H *is defined with two PPT algorithms* $\mathsf{KeyGen}, \mathsf{Eval}$ *that work as follows*

- $\mathsf{KeyGen}()$ *outputs a hash key* hk.
- $\mathsf{Eval}(\mathsf{hk}, x)$ *takes as input a hash key* hk *and a value* x*, and outputs a hash* y.

Definition 19. *We say that a hash function family* H *is collision resistant if for any PPT algorithm* $\mathcal{A}$ *we have*

$$Pr[\mathsf{H.Eval}(\mathsf{hk}, x) = \mathsf{H.Eval}(\mathsf{hk}, y) \text{ and } x \neq y : hk \leftarrow \mathsf{H.KeyGen}(); (x, y) \xleftarrow{R} \mathcal{A}(k)] < negl$$

for some negligible function $negl$.

To define admissible hash functions, we closely follow the definition of Cash et al. [CHKP10]. We note that admissible hash functions can be built from collision-resistant hash functions [BB04].

We assume that $\mathsf{H.Eval}(\mathsf{hk}, x)$ maps $\{0,1\}^n \to \{0,1\}^q$. For a given $\mathsf{hk} \leftarrow \mathsf{H.KeyGen}()$, and $K \in \{0, 1, \perp\}^q$, define

$$F_{K,\mathsf{hk}}(x) = \begin{cases} 0 & \text{if } \exists i \in [q] : \mathsf{H.Eval}(\mathsf{hk}, x)|_i = K|_i \\ 1 & \text{if } \forall i \in [q] : \mathsf{H.Eval}(\mathsf{hk}, x)|_i \neq K|_i \end{cases}$$

For $m \in [q]$, let $\mathcal{K}_m$ be the uniform distribution on all $K \in \{0, 1, \perp\}^q$ with exactly m non-$\perp$ digits.

Definition 20. H *is* f*-admissible for a function* $f : \mathbb{N}^2 \to \mathbb{R}$ *if for every polynomially bounded function* $Q(n)$*, there exists an efficiently computable function* $\mu(n)$ *and a set* $S_\mathsf{hk} \subseteq \{\{0,1\}^n\}^*$ *such that:*

- *The advantage of any PPT algorithm $\mathcal{A}$ that outputs a vector $\boldsymbol{x} \in \{\{0,1\}^n\}^{Q+1}$ in the following experiment is negligible in n*

  $\mathsf{hk} \leftarrow \mathsf{H.KeyGen}()$
  $\boldsymbol{x} \leftarrow \mathcal{A}(\mathsf{hk})$
  $\mathcal{A}$ wins if $\boldsymbol{x} \in S_{\mathsf{hk}}$.

- *For every $\mathsf{hk} \leftarrow \mathsf{H.KeyGen}()$ and every $\boldsymbol{x} = (x_0, \ldots, x_Q) \in \{\{0,1\}^n\}^{Q+1} \setminus S_{\mathsf{hk}}$,*

$$\Pr[F_{k,\mathsf{hk}}(x_0) = 1 \text{ and } F_{k,\mathsf{hk}}(x_1) = \cdots = F_{k,\mathsf{hk}}(x_Q) = 0] \geq f(n, Q)$$

Definition 21. H *is admissible if* H *is f-admissible for some function $f(n, Q)$ which is non-negligible for every polynomial $Q(n)$.*

The following IBE construction uses public key encryption, encryptor combiners, and a hash function family H. This construction gives a statically secure IBE scheme if H is collision-resistant (or injective), and an adaptively secure scheme if H is admissible and the encryptor combiner is adaptively secure.

We use a hash function family where the output of $\mathsf{H.Eval}(\mathsf{hk}, \cdot)$ is in $\{0,1\}^n$. Informally, this scheme is initialized by generating a $2 \times n$ "grid" of encryptors, encryptor combiner parameters, and a hash key. The encryptors and the combiner parameters are set as the master public key, and the $2n$ corresponding decryptors make up the master secret key. The encryptor for an identity id is generated by hashing id, reading off the bits of the hash to pick one encryptor from each column of the grid, and then combining those n encryptors. A formal construction of the scheme is as follows:

- $\mathsf{ID.Setup}()$ runs $\mathsf{EC.params} \leftarrow \mathsf{EC.Setup}()$, $\mathsf{hk} \leftarrow \mathsf{H.KeyGen}()$, and for each $i \in [n], j \in \{0,1\}$, runs $\mathsf{PK.KeyGen}() \rightarrow (\mathsf{pk}_{i,j}, \mathsf{sk}_{i,j})$. Let $E_{i,j}(\cdot) = \mathsf{PK.Enc}(\mathsf{pk}_{i,j}, \cdot)$, and let $D_{i,j}(\cdot) = \mathsf{PK.Dec}(\mathsf{sk}_{i,j}, \cdot)$.

  Output $(\mathsf{mpk}, \mathsf{msk}) = (\{\mathsf{EC.params}, \mathsf{hk}, \{E_{i,j}\}_{i\in[n],j\in\{0,1\}}, \{D_{i,j}\}_{i\in[n],j\in\{0,1\}})$

- $\mathsf{ID.Extract}(\mathsf{id}, \mathsf{msk})$ takes as input a user id and a master secret key msk. Compute $\mathsf{H.Eval}(\mathsf{hk}, \mathsf{id}) \rightarrow \sigma_1, \ldots, \sigma_n$. Output the combined decryptor $D^* \leftarrow \mathsf{EC.Extract}(\mathsf{EC.params}, \{E_{i,\sigma_i}\}_{i\in[n]}, k, D_{k,\sigma_k})$ for an arbitrary $k \in [n]$.

- $\mathsf{ID.EncGen}(\mathsf{id}, \mathsf{mpk})$ takes as input a user id and a master public key mpk. Compute $\mathsf{H.Eval}(\mathsf{hk}, \mathsf{id}) \rightarrow \sigma_1, \ldots, \sigma_n$. Output the combined encryptor $E^* \leftarrow \mathsf{EC.Combine}(\mathsf{EC.params}, \{E_{i,\sigma_i}\}_{i\in[n]})$.

Theorem 4. *If the underlying public encryptor combiner is statically secure and satisfies distributional independence,* H *is a collision-resistant hash function family (or* H *is injective), and public key encryption exists, then this IBE scheme is statically secure.*

Proof. Suppose an adversary $\mathcal{A}$ attains a non-negligible advantage in the static IBE security experiment. We construct an adversary $\mathcal{B}$ that runs $\mathcal{A}$ as a subroutine and breaks the static security of the public encryptor combiner.

$\mathcal{B}$ first obtains an identity id from $\mathcal{A}$. $\mathcal{B}$ generates $\mathsf{hk} \leftarrow \mathsf{H.KeyGen}$, and computes $\mathsf{H.Eval}(\mathsf{hk}, \mathsf{id}) \rightarrow \sigma_1, \ldots, \sigma_n$. Let Dist be a distribution on $(\{E_i\}_{i\in[n]}, \mathsf{aux})$ such that Dist_i, a distribution on $(E_i, \mathsf{aux}_i = \{E_1, \ldots, E_{i-1}, E_{i+1}, \ldots, E_n, \mathsf{aux}\})$, is secure for all $i \in [n]$. For each i, we relabel E_i to be the encryptor E_{i,σ_i} in the IBE construction. Then $\mathcal{B}$ uses a public key encryption scheme to freshly generate the remaining IBE encryptors $(E_{i,j}, D_{i,j})$ for all $i \in [n], j \in \{0,1\}, j \neq \sigma_i$.

$\mathcal{B}$ receives $(\mathsf{EC.params}, E^*, \mathsf{aux}^* = \{E_1, \ldots, E_n, \mathsf{aux}\})$ from the encryptor combiner challenger.

$\mathcal{B}$ forwards $\mathsf{mpk} = \{\mathsf{EC.params}, \mathsf{hk}, \{E_{i,j}\}_{i\in[n],j\in\{0,1\}}\}$ to $\mathcal{A}$. $\mathcal{B}$ then answers queries to simulate $\mathcal{O}_{IBE}$ for $\mathcal{A}$. $\mathcal{A}$ queries on identities $\mathsf{id}' \neq \mathsf{id}$. To answer these queries, $\mathcal{B}$ computes $\mathsf{H.Eval}(\mathsf{hk}, \mathsf{id}') \rightarrow \sigma'_1, \ldots, \sigma'_n$ and looks for the first k where σ'_k differs from σ_k. Here, collision-resistance or injectivity guarantees that such a k exists except with negligible probability. $\mathcal{B}$ computes

$$D \leftarrow \mathsf{EC.Extract}(\mathsf{EC.params}, \{E_{i,\sigma_i}\}_{i\in[n]}, k, D_{k,\sigma'_k})$$

and gives this to $\mathcal{A}$. $\mathcal{A}$ submits two messages m_0, m_1, and $\mathcal{B}$ forwards this to the challenger. The challenger sends back $E^*(m_b)$, which $\mathcal{B}$ forwards to $\mathcal{A}$. $\mathcal{A}$ will make additional identity queries, and $\mathcal{B}$ will answer these in the same fashion. $\mathcal{A}$ makes a guess b' for $\mathcal{B}$, and $\mathcal{B}$ commits to the same guess b'. This allows $\mathcal{B}$ to break the static security of the public encryptor combiner, which is a contradiction. □

We note that, combining Theorem 4 (using H as an arbitrary injective function) with the IBE-to-CCA secure PKE conversion of [CHK04], we see that encryptor combiners are a bridge between CPA- and CCA-secure public key encryption:

Corollary 1. *If (potentially bounded, non-compact) encryptor combiners exist that have distributional independence, and public key encryption exists, then CCA-secure public key encryption exists.*

Theorem 5. *If the underlying public encryptor combiner is adaptively secure and satisfies distributional independence, $H(\cdot)$ is an admissible hash function, and public key encryption exists, then our IBE scheme is adaptively secure.*

Proof. Suppose an adversary $\mathcal{A}$ attains a non-negligible advantage ϵ in the adaptive IBE security experiment. Suppose $\mathcal{A}$ makes d extract queries. We prove security using a sequence of hybrid experiments.

Hybrid 0 will correspond to the adaptive IBE experiment. In **Hybrid 1**, we add an abort condition. At the beginning of the experiment, choose a random $k \in \mathcal{K}_q$ for some q to be chosen later. At the end of the experiment, after recording all of extract queries $\{\mathsf{id}_i\}$ and the challenge query id^*, test that $F_{k,\mathsf{hk}}(\mathsf{id}^*) = 1$ and $F_{k,\mathsf{hk}}(\mathsf{id}_i) = 0$ for all i. If not, output a random bit and abort. We also need an artificial abort: If the test passes, then with probability $\gamma(\mathsf{id}^*, \{\mathsf{id}_i\})$ output a random bit and abort. Finally, if no aborts happen, output the output of $\mathcal{A}$. Here, γ is chosen so that the abort happens with probability (statistically) independent of the queries made by the adversary, and can be chosen using standard techniques. The result is that, in **Hybrid 1**, the adversary still has advantage $\epsilon/f(n,d)$, which is non-negligible. Note that we can think of the initial abort check as happening on the fly: if we ever see an extract query on id for which $F_{k,\mathsf{hk}}(\mathsf{id}_i) \neq 0$, we immediately abort. Similarly, if the challenge query is for id^* with $F_{k,\mathsf{hk}}(\mathsf{id}^*) \neq 1$, we immediately abort.

Hybrid 2 is identical to **Hybrid 1**, except we now change how extraction queries are answered. To answer a decryption query on an identity id, run $\mathsf{H.Eval}(\mathsf{hk}, \mathsf{id}) \to \sigma_1, \ldots, \sigma_n$. Use the D_{i,σ_i} for the minimal i such that $k_i = \sigma_i$. Since we already checked the abort condition, there is guaranteed to be some i. By distributional independence of the encryptor combiner, **Hybrid 2** and **Hybrid 1** are statistically indistinguishable, so $\mathcal{A}$ has non-negligible advantage in **Hybrid 2**.

We now construct an adversary $\mathcal{B}$ that runs $\mathcal{A}$ as a subroutine and breaks the adaptive security of the public encryptor combiner. Let Dist be the distribution on $(\{E_i\}_{i\in[q]})$ where each E_i is chosen freshly using the public key encryption scheme. Notice that Dist_i is secure for all $i \in [q]$. The value of q will be set later. $\mathcal{B}$ will a random vector $k \in \mathcal{K}_q$. For every pair (i,j) such that $j = k_i \neq \perp$ (of which there are q), $\mathcal{B}$ will set $E_{i,j}$ to be one of the E_i. For each of the remaining $2n - q$ index pairs (i,j), $\mathcal{B}$ freshly generates $(E_{i,j}, D_{i,j})$ using a public key encryption scheme. $\mathcal{B}$ also generates $\mathsf{hk} \leftarrow \mathsf{H.KeyGen}()$.

$\mathcal{B}$ forwards $\mathsf{mpk} = \{\mathsf{EC.params}, \mathsf{hk}, \{E_{i,j}\}_{i\in[n],j\in\{0,1\}}\}$ to $\mathcal{A}$. When $\mathcal{A}$ queries on an identity id, $\mathcal{B}$ computes $\mathsf{H.Eval}(\mathsf{hk}, \mathsf{id}) = \sigma_1, \ldots, \sigma_n$. As long as $\mathcal{B}$ possesses $D_{i,\sigma(i)}$ for some $i \in [n]$, $\mathcal{B}$ uses the first such i to compute the combined decryptor D_{id} and answer the query.

Eventually, $\mathcal{A}$ commits to an identity id^* as well as messages m_0, m_1. If $F_{k,\mathsf{hk}}(\mathsf{id}^*) \neq 1$, $\mathcal{B}$ aborts. Notice that if $F_{k,\mathsf{hk}}(\mathsf{id}^*) = 1$, then each encryptor $E_{i,\sigma(\mathsf{id}^*)}$ corresponds to an encryptor in the set $\{E_i\}_{i\in[q]}$. In this case, $\mathcal{B}$ submits the subset of encryptions corresponding to $E_{i,\sigma(\mathsf{id}^*)}$ as the challenge set S, and forwards along m_0, m_1. $\mathcal{B}$ obtains $E^*(m_b)$ from the challenger, and forwards this back to $\mathcal{A}$. $\mathcal{A}$ makes additional queries, and $\mathcal{B}$ answers them in the same fashion. Finally, $\mathcal{A}$ makes a guess b'. $\mathcal{B}$ will possibly choose to artificially abort as in **Hybrid 2**. Otherwise, $\mathcal{B}$ uses b' as his own guess.

$\mathcal{B}$ perfectly simulates the view of $\mathcal{A}$ in **Hybrid 2**. Therefore, the advantage of $\mathcal{B}$ is non-negligible. This breaks the adaptive security of the public encryptor combiner, which is a contradiction. □

4.2 Hierarchical Identity-Based Encryption

In this section, we construct an IND-sID-CPA secure hierarchical identity-based encryption scheme (HIBE) from encryptor combiners and public key encryption. This can be directly transformed into an IND-sID-CCA secure scheme using the techniques of Canetti et al. [CHK04].

HIBE generalizes IBE so that instead of there being a single central authority, there is a tree structure where an identity at a level k can act as an authority and issue private keys to descendants at level $k+1$. This identity cannot, however, decrypt messages intended for any identities outside of its set of descendants. In keeping with the style of this paper, we work with a slightly modified definition of HIBE that uses encryptors and decryptors instead of explicit keys.

We let id_0 denote the identity at the root (defined to be at depth 0), and write any identity id at depth k as a $k+1$-dimensional vector $(\mathsf{id}_0, \ldots, \mathsf{id}_k)$. Reading off this vector gives a path from the root to the node at depth k. Thus, any prefix of the identity is an ancestor in the tree (e.g. its parent identity is the k-dimensional vector $(\mathsf{id}_0, \ldots, \mathsf{id}_{k-1})$). For a given id, we denote by $\mathsf{id}|k'$ the depth k' ancestor of id, $(\mathsf{id}_0, \ldots, \mathsf{id}_{k'})$.

Formally, a HIBE scheme consists of the following PPT algorithms.

- $\mathsf{Setup}()$ outputs a master public/secret key pair $(\mathsf{mpk}, \mathsf{msk})$.
- $\mathsf{Extract}(D_{\mathsf{id}|k'}, \mathsf{id})$ takes as input an identity $\mathsf{id} = (\mathsf{id}_0, \ldots, \mathsf{id}_k)$ at depth k and the decryptor $D_{\mathsf{id}|k'}$ for ancestor identity $\mathsf{id}|k'$ where $k' < k$, and outputs the decryptor D_{id} for identity id.
- $\mathsf{EncGen}(\mathsf{mpk}, \mathsf{id})$ takes as input parameters mpk and an identity id, and outputs an encryptor E_{id}.

The master secret key msk will contain D_{id_0}.
Let id be a depth k identity. We say that D_{id} is *properly generated* if $\mathsf{id} = \mathsf{id}_0$, or if

- $D_{\mathsf{id}} \leftarrow \mathsf{Extract}(D_{id|k'}, \mathsf{id})$ for some $k' < k$, and
- $D_{\mathsf{id}|k'}$ is properly generated.

For correctness, we require that any properly generated D_{id} is a valid decryptor for E_{id}.

To define security, let $\texttt{HIBE-EXP}(b)$ for $b = 0, 1$ denote the following experiment involving an adversary $\mathcal{A}$:

$id^* \leftarrow \mathcal{A}$
$(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}()$
$(m_0, m_1) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{mpk})$
$E \leftarrow \mathsf{EncGen}(\mathsf{mpk}, \mathsf{id}^*)$
$b' \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{mpk}, E(m_b))$

Here, $\mathcal{A}$ can make polynomially many adaptive decryptor queries to the oracle $\mathcal{O}$, where $\mathcal{A}$ submits an identity id and $\mathcal{O}$ returns $D_{\mathsf{id}} \leftarrow \mathsf{Extract}(D_{\mathsf{id}|0}, \mathsf{id})$. The only restriction is that id may not be the challenge identity id^*, or any prefix of it.

For $b = 0, 1$ let W_b be the event that $b' = 1$ in $\texttt{HIBE-EXP}(b)$ and define $\texttt{HIBE-EXP}^{\text{adv}} = |\Pr[W_0] - \Pr[W_1]|$.

Definition 22. *A HIBE scheme is IND-sID-CPA secure if for all PPT adversaries $\mathcal{A}$, $\texttt{HIBE-EXP}^{adv}$ is negligible.*

We construct a k-level IND-sID-CPA secure HIBE scheme using statically secure encryptor combiners with distributional independence and public key encryption. Informally, our construction assigns a $2 \times n$ grid of encryptors to each identity, where n is set to be the number of bits in the string id_i. The grid of encryptors for the root identity is generated through a public key encryption scheme. The top $k-1$ levels each have a $2 \times n$ auxiliary grid of encryptors as well as public encryptor combiner parameters. The decryptor for an identity consists of the $2 \times n$ grid of decryptors corresponding to the encryptors for that

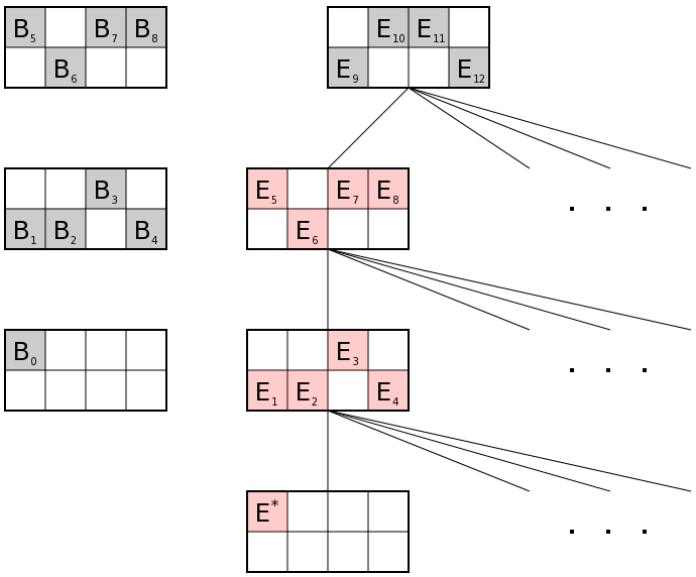

Figure 1: In this proof illustration, the shaded encryptors $B_0, B_1, \ldots, B_8$ and $E_9, \ldots, E_{12}$ are the ones generated from independent secure distributions (note that this illustration uses slightly different notation from the proof for simplicity). The adversary will challenge the identity corresponding to the grid that contains E^*. Thus, the distribution on E^* must not be secure, and encryptor combiner security implies that the distribution on one of $B_0, E_1, \ldots, E_4$ is not secure. By assumption, B_0 is secure, so one of $E_1, \ldots, E_4$ is not secure. This argument continues until one of $E_9, \ldots, E_{12}$ must not be secure, but this contradicts the starting assumption.

identity, and decryption is done by feeding the ciphertext into one of these decryptors (for consistency, we pick the top left decryptor). We generate the $2 \times n$ grid of encryptors for identity $\mathsf{id} = (\mathsf{id}_0, \ldots, \mathsf{id}_k)$ at level k as follows. Select n encryptors, one from each column, from the grid of encryptors for identity $\mathsf{id}|k-1$ by reading off the bits of id_k. 0 corresponds to choosing the encryptor in the top row, and 1 corresponds to choosing the encryptor in the bottom row. To fill out the encryptor at position (i, j), we select the encryptor (i, j) position in the depth $k-1$ auxiliary grid. These $n+1$ encryptors are combined using the encryptor combiner parameters for level $k-1$.

Formally, our construction consists of the following PPT algorithms.

- $\mathsf{Setup}()$. For each $l \in [k-1]$, generate $\mathsf{params}_l \leftarrow \mathsf{EC.Setup}$, as well as an auxiliary grid of $2 \times n$ encryptors $\{E^l_{i,j}\}_{i \in \{0,1\}, j \in [n]}$. For the root node (with identity id_1), generate $\{E^{\mathsf{id}_1}_{i,j}\}_{i \in \{0,1\}, j \in [n]}$, a grid of $2 \times n$ encryptors, as well as $\{D^{\mathsf{id}_1}_{i,j}\}_{i \in \{0,1\}, j \in [n]}$, the corresponding $2 \times n$ decryptors using $\mathsf{PK.KeyGen}()$. Output $\mathsf{mpk} = \{\{\mathsf{params}_l, \{E^l_{i,j}\}_{i \in \{0,1\}, j \in [n]}\}_{l \in [k-1]}, \{E^{id_1}_{i,j}\}_{i \in \{0,1\}, j \in [n]}\}$, and $\mathsf{msk} = \{D^{\mathsf{id}_1}_{i,j}\}_{i \in \{0,1\}, j \in [n]}$.

- $\mathsf{Extract}(D_{\mathsf{id}|k'}, \mathsf{id})$. The decryptor $D_{\mathsf{id}|k'}$ consists of the $2 \times n$ decryptors $\{D^{id|k'}_{i,j}\}_{i \in \{0,1\}, j \in [n]}$ at node $id|k'$. Write $\mathsf{id} = (\mathsf{id}_1, \mathsf{id}_2, \ldots, \mathsf{id}_k)$, where $\mathsf{id}_k \in \{0,1\}^n$. Let $[\mathsf{id}_k]_j$ denote the jth bit of id_k. First, compute $\{E^{\mathsf{id}|k-1}_{[id_k]_j, j}\}_{j \in [n]}$ by combining encryptors starting at the root node. For each $i \in \{0,1\}, j \in [n]$, we compute $D^{\mathsf{id}}_{i,j} \leftarrow \mathsf{EC.Extract}(\mathsf{params}_k, \{E_{[\mathsf{id}_k]_j, j}\}_{j \in [n]} \cup \{E^k_{i,j}\}, n+1, D^k_{i,j})$ (and we follow this same procedure to compute $D^k_{i,j}$ from the previous level, going up until $D^{k'}_{i,j}$) and thus return $\{D^{\mathsf{id}}_{i,j}\}_{i \in \{0,1\}, j \in [n]}$. This decryptor performs decryption on ciphertext c by outputting $D^{\mathsf{id}}_{0,1}$.

- $\mathsf{EncGen}(\mathsf{params}, \mathsf{id}, m)$. We compute $E^{\mathsf{id}}_{0,1}$ by combining encryptors starting at the root node. Output $E^{id}_{0,1}(m)$.

Theorem 6. *If the underlying encryptor combiner is statically secure and satisfies distributional independence, the above construction gives an IND-sID-CPA secure k-level HIBE scheme.*

Proof. We give a simplified illustration of this proof in Figure 1.

Suppose for the sake of contradiction that an adversary $\mathcal{A}$ breaks the IND-sID-CPA security of this HIBE scheme.

$\mathcal{A}$ first commits to a level k identity $\mathsf{id}^* = (\mathsf{id}_0, \ldots, \mathsf{id}_k)$. In this proof, we assume that $\mathcal{A}$ only attacks identities at the lowest level of the HIBE tree.

Now we describe a specific instantiation of our HIBE construction where, if $\mathcal{A}$ can break IND-sID-CPA security, encryptor combiner security must also be compromised.

As usual, the encryptor combiner parameters params_l are independently generated for each level l. However, the encryptors in the root node and $k-1$ buffers are determined based on $\mathcal{A}$'s choice of id^*.

The following $n(k-1)+1$ encryptors will all be generated from independent secure distributions:

- $\{E^{id_1}_{[id_2]_j,j}\}_{j\in[n]}$
- $\{E^{h}_{[\mathsf{id}_{h+1}]_j,j}\}_{j\in[n]}$ for $h \in [k-2]$
- $E^{k-1}_{1,1}$

The remaining $2nk - (n(k-1)+1) = nk+n-1$ encryptors will be independently generated along with their corresponding decryptors.

Define aux_h to be the set consisting of

- $\{E^{\mathsf{id}_1}_{i,j}\}_{i\in\{0,1\},j\in[n]}$
- $\{E^{h}_{i,j}\}_{i\in\{0,1\},j\in[n]}$ for $h \in [k-1]$
- The set of all $nk+n-1$ generated decryptors
- params'_h for $h' \in [h-1]$.

We know that $\mathcal{A}$ can break the security of

$$\mathsf{Dist}_1 = (E^k_{1,1}, \mathsf{aux} = \{E^{k-1}_{1,1}, \{E^{\mathsf{id}_{k-1}}_{[\mathsf{id}_k]_j,j}\}_{j\in[n]}, \mathsf{aux}_{k-1}, \mathsf{params}_{k-1}\}),$$

since aux_{k-1} contains enough information for $\mathcal{A}$ to answer all oracle decryption queries.

For readability, we relabel this distribution as Dist^*, ($E^k_{1,1}$ as E^*, $E^{k-1}_{1,1}$ as E_0, and $\{E^{\mathsf{id}_{k-1}}_{[\mathsf{id}_k]_j,j}\}_{j\in[n]}$ sequentially as $E_1, \ldots, E_n$. Since $\mathcal{A}$ breaks the security of

$$\mathsf{Dist}^* = (E_*, \mathsf{aux} = \{E_0, E_1, \ldots, E_n\}, \mathsf{aux}_{k-1}, \mathsf{params}_{k-1}\}),$$

encryptor combiner security implies at least one distribution of the form

$$\mathsf{Dist}_i = (E_i, \mathsf{aux} = \{E_0, \ldots, E_{i-1}, E_{i+1}, \ldots, E_n, \mathsf{aux}_{k-1}\})$$

must not be secure.

However, we know that Dist_0 is secure since $E_0 = E^{k-1}_{1,1}$ was independently securely generated. Thus, Dist_i is not secure for some $i \in [n]$. Consider such an i. We note that Dist_i can be rewritten as the combined distribution of another set of encryptors, but from one level higher. In other words, the distributions

$$\mathsf{Dist}_i = (E^{\mathsf{id}_{k-1}}_{[\mathsf{id}_k]_i,i}, \mathsf{aux} = \{E^{k-1}_{1,1}, \{E^{\mathsf{id}_{k-1}}_{[\mathsf{id}_k]_j,j}\}_{j\in[n],j\neq i}, \mathsf{aux}_{k-1}, \mathsf{params}_{k-1}\}),$$

and

$$\mathsf{Dist}'_i = (E^{\mathsf{id}_{k-1}}_{[\mathsf{id}_k]_i,i}, \mathsf{aux} = \{E^{k-2}_{[\mathsf{id}_k]_i,i}, \{E^{\mathsf{id}_{k-2}}_{[\mathsf{id}_{k-1}]_j,j}\}_{j\in[n]}, \mathsf{aux}_{k-2}, \mathsf{params}_{k-2}\}),$$

contain the same amount of information, since the auxiliary information aux_{k-2} can be used to generate the encryptors not in common.

We can repeat this process until the combined distribution consists only of encryptors that were independently securely generated. But then encryptor combiner security implies that one of those encryptor distributions is not secure, which is a contradiction.

□

The transformation of [CHK04], can be used to turn any CCA-secure HIBE scheme into a CPA-secure HIBE at the cost of losing one level. Thus, we can immediately strengthen the security of this construction.

Theorem 7. *An IND-sID-CCA secure HIBE scheme can be built from public key encryption and a statically secure encryptor combiner with distributional independence.*

5 Broadcast Encryption

A broadcast encryption scheme lets a user broadcast a message to a subset of recipients. The scheme is collusion resistant if nobody outside the intended set of recipients can learn anything about the message.

In this section, we use encryptor combiners and identity-based encryption to build broadcast encryption schemes for identity spaces of various sizes.

5.1 Broadcast Encryption Definition

Definition 23. *A broadcast encryption scheme for identity space* ID *is a set of PPT algorithms* $(\mathsf{Setup}, \mathsf{EncGen}, \mathsf{DecGen}, \mathsf{Extract}$ *with the following properties.*

- $\mathsf{Setup}(\mathsf{ID})$ *takes the identity space* ID *as input and outputs the master public /secret key pair* $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(\mathsf{ID})$.
- $\mathsf{EncGen}(\mathsf{mpk}, S)$ *takes as input the master public key* mpk *and a set* $S \subseteq \mathsf{ID}$ *of user identities, and outputs an encryptor* $E \leftarrow \mathsf{EncGen}(\mathsf{mpk}, S)$ *for broadcasting to the set* S.
- $\mathsf{DecGen}(\mathsf{mpk}, \mathsf{id}, D_{\mathsf{id}}, S)$ *takes as input the master public key* mpk*, a user identity* $\mathsf{id} \in \mathsf{ID}$*, user* id*'s decryptor* D_{id}*, and a set* $S \subseteq \mathsf{ID}$ *of user identities, and outputs a decryptor* $D \leftarrow \mathsf{DecGen}(\mathsf{mpk}, \mathsf{id}, D_{\mathsf{id}}, S)$ *to decrypt ciphertexts intended for users in* S.
- $\mathsf{Extract}(\mathsf{id}, \mathsf{msk})$ *takes as input a user identity* $\mathsf{id} \in \mathsf{ID}$ *and the master secret key* msk*, and outputs the user's decryptor* $D_{\mathsf{id}} \leftarrow \mathsf{Extract}(\mathsf{id}, \mathsf{msk})$.

We will consider three cases for ID. The case where $\mathsf{ID} = \{1, \ldots, poly\}$ and user identities are given out sequentially corresponds to the traditional notion of broadcast encryption where the number of users is bounded. The case where $\mathsf{ID} = \{1, \ldots, superpoly\}$ and user identities are given out sequentially corresponds to broadcast encryption with an unbounded number of users. Finally, when $\mathsf{ID} = \{1, \ldots, exponential\}$ and identities are arbitrary this is equivalent to identity-based broadcast encryption (IBBE).

For correctness, we require that for all subsets $S \subseteq \mathsf{ID}$ and for all identities $\mathsf{id} \in S$, if $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(\mathsf{ID})$, $D_{\mathsf{id}} \leftarrow \mathsf{Extract}(\mathsf{id}, \mathsf{msk})$, and $c \leftarrow E(m)$ for $E \leftarrow \mathsf{EncGen}(\mathsf{mpk}, S)$, then for $D \leftarrow \mathsf{DecGen}(\mathsf{mpk}, \mathsf{id}, D_{\mathsf{id}}, S)$, $D(c)$ outputs m.

We first define adaptive CPA-security. For $b = 0, 1$ denote by $\texttt{BE-EXP}(b)$ the following experiment:

$(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(\mathsf{ID})$
$(m_0, m_1, S^*) \leftarrow \mathcal{A}^{\mathsf{RD}(\cdot)}(\mathsf{mpk})$
$E \leftarrow \mathsf{EncGen}(\mathsf{mpk}, S^*)$
$c^* \leftarrow E(m_b)$
$b' \leftarrow \mathcal{A}^{\mathsf{RD}(\cdot)}(c^*)$

where $\mathsf{RD}(\mathsf{id})$ is an extract oracle that takes as input $\mathsf{id} \in \mathsf{ID}$ and returns $D_{\mathsf{id}} \leftarrow \mathsf{Extract}(\mathsf{id}, \mathsf{msk})$.

$\mathcal{A}^{\mathsf{RD}(\cdot)}$ denotes an adversary $\mathcal{A}$ that can adaptively query the recipient decryptor oracle a polynomial number of times.

For $b = 0, 1$ let W_b be the event that $b' = 1$ in $\texttt{BE-EXP}(b)$ and define $\texttt{BE-EXP}^{\text{adv}} = |\Pr[W_0] - \Pr[W_1]|$.

Definition 24. *A broadcast encryption scheme is adaptively CPA-secure if* $\texttt{BE-EXP}^{adv}$ *is negligible for all PPT adversaries* $\mathcal{A}$*.*

Static CPA-security can be defined similarly. The only change in the security experiment is that $\mathcal{A}$ must pick S^* before the setup step.

We can extend this definition to adaptive/static CCA-security by modifying RD to also answer CCA decryption queries. In these queries, $\mathcal{A}$ submits (S, id, c), consisting of a set of users S, an identity $\mathsf{id} \in S$, and a cyphertext c, and receives the decryption of c by user id: $D(c)$ where $D \leftarrow \mathsf{DecGen}(\mathsf{mpk}, \mathsf{id}, D_{\mathsf{id}}, S)$, $D_{\mathsf{id}} \leftarrow \mathsf{Extract}(\mathsf{id}, \mathsf{msk})$. $\mathcal{A}$ is allowed to submit any query of this form provided that $(S, c) \neq (S^*, c^*)$.

5.2 Broadcast Encryption Construction

We construct broadcast encryption for identity space ID given an identity based encryption scheme $IBE = (\mathsf{IBE.Setup}, \mathsf{IBE.Extract}, \mathsf{IBE.EncGen})$ and a public encryptor combiner $(\mathsf{EC.Setup}, \mathsf{EC.Combine}, \mathsf{EC.Extract})$. We assume the encryptor combiner satisfies compactness, which gives optimal-length ciphertexts in this scheme.

- $\mathsf{BE.Setup}(\mathsf{ID})$ outputs $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{IBE.Setup}()$.
- $\mathsf{BE.EncGen}(\mathsf{mpk}, S)$ takes as input a master public key mpk and a subset $S \subseteq \mathsf{ID}$. For each $\mathsf{id} \in S$, let $E_{\mathsf{mpk},\mathsf{id}} \leftarrow \mathsf{ID.EncGen}(\mathsf{mpk}, \mathsf{id})$. Let $\mathsf{EC.params} \leftarrow \mathsf{EC.Setup}()$. Set $E^* \leftarrow \mathsf{EC.Combine}(\mathsf{EC.params}, \{E_{\mathsf{mpk},\mathsf{id}}\}_{\mathsf{id} \in S})$. Define E to be the encryptor that, on input m, outputs $(\mathsf{EC.params}, E^*(m))$. Output E.
- $\mathsf{BE.DecGen}(\mathsf{mpk}, \mathsf{id}, D_{\mathsf{id}}, S)$ takes as input a master public key mpk, a user identity $\mathsf{id} \in \mathsf{ID}$, decryptor D_{id}, and a subset S. Define D to work as follows. On input $E(m) = (\mathsf{EC.params}, E^*(m))$, it computes

$$D^* \leftarrow \mathsf{EC.Extract}(\mathsf{EC.params}, \{E_{\mathsf{mpk},\mathsf{id}}\}_{\mathsf{id} \in S}, \mathsf{id}, D_{\mathsf{id}})$$

  and then outputs $D^*(E^*(m))$.
- $\mathsf{BE.Extract}(\mathsf{id}, \mathsf{msk})$ takes as input a user identity $\mathsf{id} \in \mathsf{ID}$ and the master secret key msk. Output $D_{\mathsf{id}} \leftarrow \mathsf{IBE.Extract}(\mathsf{id}, \mathsf{msk})$.

Theorem 8. *If the underlying IBE scheme is adaptively secure, the public encryptor combiner is statically secure and compact, this scheme is adaptively secure. The encryptor combiner does not need to satisfy independence.*

Note the encryptor combiner-based IBE from Section 4 was only proved to be statically secure. However, if ID is polynomial in size, then adaptive and static security are equivalent by guessing the challenge identity. Moreover, even if ID is superpolynomial, but identities are given out sequentially, then adaptive and static security are equivalent. As such, we obtain the following corollary:

Corollary 2. *If compact statically secure public encryptor combiners exist and public key encryption exists, then adaptively-secure (non-identity-based) broadcast encryption for an unbounded number of users exists.*

Proof of Theorem 8. Suppose there exists an adversary $\mathcal{A}$ that attains a non-negligible advantage in the identity-based broadcast encryption static security experiment (equivalent to the static broadcast encryption experiment). We separate the adversary $\mathcal{A}$ into adversaries $\mathcal{A}_0$ and $\mathcal{A}_1$ that run in sequence. $\mathcal{A}_0$ first receives mpk from the IBBE challenger. $\mathcal{A}_0$ submits a polynomial number of user identity queries, and receives those users' decryptors. $\mathcal{A}_0$ finishes by sending the IBBE challenger a set S of user identities (where S does not contain any users that $\mathcal{A}_0$ has queried on) as well as choices for messages m_0, m_1. $\mathcal{A}_0$ terminates and

outputs $(S, m_0, m_1, \mathsf{state})$ where state is its internal state. $\mathcal{A}_1$ starts by receiving $(S, m_0, m_1, \mathsf{state})$. The IBBE challenger forwards a ciphertext $c^* = E(m_b)$ for $E \leftarrow \mathsf{BE.EncGen}(\mathsf{mpk}, S)$ for a randomly chosen $b \in \{0, 1\}$. $\mathcal{A}_1$ is then free to make polynomially many identity queries on users outside of S, and receives those users' decryptors. Finally, $\mathcal{A}_1$ submits a guess b' for b.

We define Dist as follows. Generate $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{IBE.Setup}(\mathsf{ID})$. Dist plays the role of the identity-based broadcast encryption challenger for $\mathcal{A}_0$. Dist gives mpk to adversary $\mathcal{A}_0$, and uses msk to answer $\mathcal{A}_0$'s identity queries. $\mathcal{A}_0$ finishes and returns $(S, m_0, m_1, \mathsf{state})$. Dist outputs $(\{E_{\mathsf{mpk},\mathsf{id}}\}_{\mathsf{id} \in S}, \mathsf{aux} = \{\}, \mathcal{O})$, where $\mathcal{O}$ is an oracle that outputs $(S, m_0, m_1, \mathsf{state})$ when queried on $\perp$, and outputs $D_{\mathsf{id}'}$ when queried on $\mathsf{id}' \notin S$. Given Dist, we define Dist_i to output $(E_{i_S}, \mathsf{aux} = \{E_{\mathsf{mpk},\mathsf{id}}\}, \mathcal{O})$, where E_{i_S} denotes the i^{th} encryptor in S.

We claim that Dist_i is an oracle-secure distribution on encryptors. To prove this, suppose some adversary $\mathcal{C}_i$ breaks the security of this distribution. We use it to construct an adversary $\mathcal{F}_i$ that breaks the adaptive security of Identity-Based Encryption. $\mathcal{F}_i$ receives mpk from the IBE challenger. $\mathcal{F}_i$ runs $\mathcal{A}_0$ as a subroutine. $\mathcal{F}_i$ forwards mpk to $\mathcal{A}_0$. $\mathcal{F}_i$ is given $\mathcal{O}_{IBE}$, the identity decryption oracle, and uses it to honestly answer $\mathcal{A}_0$'s identity queries. At the end, $\mathcal{A}_0$ outputs $(S, m_0, m_1, \mathsf{state})$. $\mathcal{F}_i$ gives E_{i_S}, the i^{th} encryptor in S, to $\mathcal{C}_i$. Additionally, $\mathcal{F}_i$ simulates the oracle $\mathcal{O}$ for $\mathcal{C}_i$ with queries on $\mathcal{O}_{IBE}$. $\mathcal{C}_i$ then picks messages m'_0, m'_1 and forwards them to $\mathcal{F}_i$. Finally, $\mathcal{F}_i$ passes along m'_0, m'_1 to the IBE challenger, and receives a ciphertext $c = E_{i_S}(m'_b)$ in return, which it forwards to $\mathcal{C}_i$. Since $\mathcal{C}_i$ attains a non-negligible advantage in guessing b, $\mathcal{F}_i$ simply uses $\mathcal{C}_i$'s guessed bit to break the static security of the IBE scheme.

Given that Dist_i for all i is an oracle-secure distribution on encryptors, we construct an adversary $\mathcal{B}$ that breaks the security of the oracle public encryptor combiner. $\mathcal{B}$ samples $(E^*, \mathcal{O}^*)$ from Dist^*. $\mathcal{B}$ queries O^* on $\perp$ to obtain $(S, m_0, m_1, \mathsf{state})$, and then forwards m_0, m_1 to the oracle public encryptor combiner challenger. The challenger then sends $\mathcal{B}$ the ciphertext $E^*(m_b)$. $\mathcal{B}$ now runs $\mathcal{A}_1$ as a subroutine. It passes $(S, m_0, m_1, \mathsf{state})$ to $\mathcal{A}_1$. Next, $\mathcal{B}$ forwards the ciphertext $E^*(m_b)$ to $\mathcal{A}_1$. $\mathcal{A}_1$ makes identity queries on users outside of S, which $\mathcal{B}$ answers honestly by using $\mathcal{O}_{\mathsf{id}}$. $\mathcal{A}_1$ makes a guess b' for b, and $\mathcal{B}$ forwards along the bit as its own guess. Since $\mathcal{A}_1$ attains a non-negligible advantage in guessing b', $\mathcal{B}$ breaks the static oracle-security of the public encryptor combiner.

Since the public encryptor combiner is oracle-secure, there cannot exist $\mathcal{A}_0$ and $\mathcal{A}_1$ that break the security of the identity-based broadcast encryption scheme.

Theorem 9. *If the underlying IBE scheme is statically secure, the public encryptor combiner is statically secure and compact, this scheme is statically secure. The encryptor combiner does not need to satisfy independence.*

Proof. (informal) The proof of Theorem 8 can be easily modified so that the adversary $\mathcal{A}_0$ commits to a set S before receiving mpk from the IBBE challenger. The only other change is that $\mathcal{F}_i$ commits to an identity $\mathsf{id} = S_i$ before receiving mpk, which it can do since $\mathcal{A}_0$ can now output S before receiving mpk. □

Therefore, combining with our encryptor combiner-based IBE, we obtain

Corollary 3. *If compact statically secure public encryptor combiners exist and public key encryption exists, then statically-secure identity-based broadcast encryption exists*

5.3 CCA-Secure Broadcast Encryption

We now show how to make our broadcast scheme CCA secure. Before we give this construction, we very briefly review universal one-way hash function families. A hash function family H consists of PPT algorithms KeyGen, which outputs a hash key hk, and $\mathsf{Eval}(\mathsf{hk}, x)$, which takes the hash key hk and an input x, and outputs a hash y. The hash family is a universal one-way hash function family if no PPT adversary $\mathcal{A}$ can attain a non-negligible advantage in the following experiment: $\mathcal{A}$ commits to a value x_1, and the challenger generates $\mathsf{hk} \leftarrow \mathsf{KeyGen}$. Now given hk, $\mathcal{A}$ generates another value $x_2 \neq x_1$, and wins if $\mathsf{Eval}(\mathsf{hk}, x_1) = \mathsf{Eval}(\mathsf{hk}, x_2)$.

We construct a statically CCA-secure Broadcast Encryption scheme using a public encryptor combiner EC, a CPA-secure Broadcast Encryption scheme BE, a Identity Based Encryption scheme IBE, a strongly

one-time digital signature scheme DS, and a universal one-way hash function family H. Here, we assume the encryptor combiner satisfies independence.

The idea is to follow the conversion of IBE to CCA-secure encryption by Canetti et al. [CHK04]. To encrypt, choose a random verification and signing key $(\mathsf{vk}, \mathsf{sk})$ for a one-time strongly secure digital signature. Interpret vk as an identity for the IBE scheme, let E_{vk} be the encryptor corresponding to that identity. Let E_S be the encryptor for the set S using the CPA-secure broadcast scheme. Generate the encryptor E'_S as the result of combining E_S and E_{vk} using an encryptor combiner. The ciphertext is obtained by encrypting the message using E'_S, and then signing the result using sk, and outputting vk, the encrypted message, and the signature. More formally:

- $\mathsf{Setup}(\mathsf{ID})$ takes the identity space ID as input and generates $(\mathsf{mpk}_{\mathsf{BE}}, \mathsf{msk}_{\mathsf{BE}}) \leftarrow \mathsf{BE.Setup}(\mathsf{ID})$, $(\mathsf{mpk}_{\mathsf{IBE}}, \mathsf{msk}_{\mathsf{IBE}}) \leftarrow \mathsf{IBE.Setup}(\mathsf{ID})$, $\mathsf{EC.params} \leftarrow \mathsf{EC.Setup}()$. Set $\mathsf{mpk} = (\mathsf{mpk}_{\mathsf{BE}}, \mathsf{mpk}_{\mathsf{IBE}}, \mathsf{EC.params})$, $\mathsf{msk} = (\mathsf{msk}_{\mathsf{BE}}, \mathsf{msk}_{\mathsf{IBE}})$, and output $(\mathsf{mpk}, \mathsf{msk})$.
- $\mathsf{EncGen}(\mathsf{mpk}, S)$ takes as input $\mathsf{mpk} = (\mathsf{mpk}_{\mathsf{BE}}, \mathsf{mpk}_{\mathsf{IBE}}, \mathsf{EC.params})$ and a set of identities S. Output encryptor E, defined as follows.

  > Generate $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{DS.Gen}()$. Let $\mathsf{hk} \leftarrow \mathsf{H.KeyGen}()$ and set $\tau \leftarrow \mathsf{H.Eval}(\mathsf{hk}, S)$. Let $E_0 \leftarrow \mathsf{BE.EncGen}(\mathsf{mpk}_{\mathsf{BE}}, S)$, $E_1 \leftarrow \mathsf{IBE.EncGen}(\mathsf{mpk}_{\mathsf{IBE}}, \mathsf{vk})$. Set $E_2 \leftarrow \mathsf{EC.Combine}(\mathsf{EC.params}, E_0, E_1)$. Set $\sigma \leftarrow \mathsf{DS.Sig}(\mathsf{sk}, (E_2(m), \mathsf{hk}, \tau))$. Output $(E_2(m), \mathsf{hk}, \tau, \mathsf{vk}, \sigma)$.
- $\mathsf{DecGen}(\mathsf{mpk}, \mathsf{id}, D_{\mathsf{id}}, S)$ takes as input $\mathsf{mpk} = (\mathsf{mpk}_{\mathsf{BE}}, \mathsf{mpk}_{\mathsf{IBE}}, \mathsf{EC.params})$, an identity id, the corresponding decryptor D_{id}, and a set of identities S. On input of the form $(E_2(m), \mathsf{hk}, \tau, \mathsf{vk}, \sigma)$, check if $\mathsf{DS.Ver}(\mathsf{vk}, (E_2(m), \mathsf{hk}, \tau), \sigma)$ accepts, and if not, abort. Also check if $\mathsf{H.Eval}(\mathsf{hk}, S) = \tau$, and if not, abort. Now compute $D_0 \leftarrow \mathsf{BE.DecGen}(\mathsf{mpk}_{\mathsf{BE}}, \mathsf{id}, D_{\mathsf{id}}, S)$ and output $D \leftarrow \mathsf{EC.Extract}(\mathsf{EC.params}, \{E_0, E_1\}, 0, D_0)$.
- $\mathsf{Extract}(\mathsf{id}, \mathsf{msk})$ takes as input an identity id and the master secret key $\mathsf{msk} = (\mathsf{msk}_{\mathsf{BE}}, \mathsf{msk}_{\mathsf{IBE}})$. Output $D_{\mathsf{id}} \leftarrow \mathsf{BE.Extract}(\mathsf{msk}_{\mathsf{BE}}, \mathsf{id})$.

Theorem 10. *If the underlying broadcast encryption scheme is statically CPA-secure, the IBE scheme is statically secure, the public encryptor combiner is statically secure and satisfies distributional independence, the digital signature scheme is a strong one-time secure scheme, and the hash function used is a universal one-way hash function, then this scheme is statically CCA-secure.*

Proof. Assume towards a contradiction that an adversary $\mathcal{A}$ breaks the static CCA-security of this scheme. We break up $\mathcal{A}$ into two adversaries, $\mathcal{A}_0, \mathcal{A}_1$ that run in sequence.

$\mathcal{A}_0$ submits a set of users S^* and some state state to its challenger, and then terminates.

$\mathcal{A}_1$ receives $\mathsf{mpk}, S^*, \mathsf{state}$ from its challenger. $\mathcal{A}_1$ can then adaptively make polynomially many extract queries and CCA decryption queries. After this, $\mathcal{A}_1$ submits messages m_0, m_1 as a challenge query and receives the challenge ciphertext. Then $\mathcal{A}_1$ can make further extract and CCA decryption queries, and finishes by trying to guess the challenge bit b'.

We define the distribution Dist to work as follows. It generates $s^*, \mathsf{state} \leftarrow \mathcal{A}_0$, $(\mathsf{mpk}_{\mathsf{IBE}}, \mathsf{msk}_{\mathsf{IBE}}) \leftarrow \mathsf{IBE.Setup}$, $(\mathsf{mpk}_{\mathsf{BE}}, \mathsf{msk}_{\mathsf{BE}}) \leftarrow \mathsf{BE.Setup}$, $E_{S^*} \leftarrow \mathsf{BE.EncGen}(\mathsf{mpk}_{\mathsf{BE}}, S^*)$, $(\mathsf{sk}^*, \mathsf{vk}^*) \leftarrow \mathsf{DS.Setup}()$, $E_{\mathsf{vk}^*} \leftarrow \mathsf{IBE.EncGen}(\mathsf{mpk}_{\mathsf{IBE}}, \mathsf{vk}^*)$. It outputs $(E_{S^*}, E_{\mathsf{vk}^*}, \mathsf{aux}, \mathcal{O})$ with $\mathsf{aux} = (\mathsf{mpk}_{\mathsf{IBE}}, \mathsf{mpk}_{\mathsf{BE}}, \mathsf{sk}^*, S^*, \mathsf{state}, \mathsf{vk})$. Here, $\mathcal{O}$ can answer broadcast encryption extract queries for any user outside of S^*, and can also answer IBE extract queries on any identity except vk^*. On all other queries, $\mathcal{O}$ returns $\perp$.

Let $\mathsf{Dist}_{S^*} = (E_{S^*}, \mathsf{aux}_{S^*} = \{E_{\mathsf{vk}^*}, \mathsf{aux}\}, \mathcal{O})$ and $\mathsf{Dist}_{\mathsf{vk}^*} = (E_{\mathsf{vk}^*}, \mathsf{aux}_{\mathsf{vk}^*} = \{E_{S^*}, \mathsf{aux}\}, \mathcal{O})$. We claim that both of these distributions are secure. Suppose Dist_{S^*} is not secure. An adversary that breaks the security of this distribution can be used to break the static security of broadcast encryption, since the auxiliary information and the oracle $\mathcal{O}$ can be simulated by another adversary interacting with the broadcast encryption challenger. A similar argument based on static security of IBE shows that $\mathsf{Dist}_{\mathsf{vk}^*}$ is secure.

Thus, the combined distribution $\mathsf{Dist}^* = (E^*, \mathsf{aux}^* = \{E_{S^*}, E_{\mathsf{vk}^*}, \mathsf{aux}, \mathsf{EC.params}\}, \mathcal{O})$ where $\mathsf{EC.params} \leftarrow \mathsf{EC.Setup}()$ is secure by the static security of the public encryptor combiner.

Define **Hybrid 0** to be the following game in which an adversary $\mathcal{B}$ interacts with the encryptor combiner challenger. The challenger sends $\mathcal{B}$ a sample $(E^*, \mathsf{aux}^* = \{E_{S^*}, E_{\mathsf{vk}^*}, \mathsf{aux} = (\mathsf{mpk}_{\mathsf{IBE}}, \mathsf{mpk}_{\mathsf{BE}}, \mathsf{sk}^*, S^*, \mathsf{state}, \mathsf{vk}^*), \mathsf{EC.params}\}, \mathcal{O})$ from Dist^*. $\mathcal{B}$ simulates $\mathcal{A}_1$ and plays the role of the CCA BE challenger. It first sends $\mathsf{mpk}, S^*, \mathsf{state}$ to $\mathcal{A}_1$. $\mathcal{A}_1$ will make extract queries on users outside of S^*, and $\mathcal{B}$ will respond by querying $\mathcal{O}$ and forwarding the response. $\mathcal{A}_1$ makes CCA decryption queries (S, id, c) where S is a set of users, id is a user in S, and c is a ciphertext of the form $c = (E(m), \mathsf{hk}, \tau, \mathsf{vk}, \sigma)$. $\mathcal{B}$ first checks if $\mathsf{DS.Ver}(\mathsf{vk}, (E(m), \mathsf{hk}, \tau), \sigma)$ accepts, and returns $\perp$ to $\mathcal{A}_1$ if it fails. Additionally, if $\mathsf{vk} = \mathsf{vk}^*$, $\mathcal{B}$ returns $\perp$ to $\mathcal{A}_1$. Otherwise, $\mathcal{B}$ queries $\mathcal{O}$ to obtain D_{vk}. It also computes $E_S \leftarrow \mathsf{BE.EncGen}(\mathsf{mpk}_{\mathsf{BE}}, S)$ and $E_{\mathsf{vk}} \leftarrow \mathsf{IBE.EncGen}(\mathsf{mpk}_{\mathsf{IBE}}, \mathsf{vk})$. $\mathcal{B}$ then computes $D \leftarrow \mathsf{EC.Extract}, (E_S, E_{\mathsf{vk}}, 1, D_{\mathsf{vk}})$, and returns $D(E(m))$ to $\mathcal{A}_1$. On the challenge ciphertext, $\mathcal{A}_1$ submits two messages m_0, m_1. $\mathcal{B}$ forwards this the encryptor combiner challenger, who sends back $E^*(m_b)$. $\mathcal{B}$ generates $\mathsf{hk}^* \leftarrow \mathsf{H.KeyGen}()$, and computes $\tau^* \leftarrow \mathsf{H.Eval}(\mathsf{hk}^*, S^*)$ and $\sigma^* \leftarrow \mathsf{DS.Sig}(\mathsf{sk}^*, E^*(m_b))$. $\mathcal{B}$ sends $(E^*(m_b), \mathsf{hk}^*, \tau^*, \mathsf{vk}^*, \sigma^*)$ to $\mathcal{A}_1$. $\mathcal{A}_1$ guesses a bit b', and $\mathcal{B}$ commits to the same bit.

Define **Hybrid 1** to be the same as Hybrid 0, except now when $\mathcal{A}_1$ makes a CCA decryption query on (S, id, c) where $c = (E(m), \mathsf{hk}, \tau, \mathsf{vk}, \sigma)$ and all checks have passed, $\mathcal{B}$ answers by using the $\mathcal{O}$ to obtain D'_{id}, the decryptor for identity id in the CPA-BE scheme. Note that the same decryptor $D_{\mathsf{id}} = D'_{\mathsf{id}}$ is used in the CCA-BE construction, so $\mathcal{B}$ computes $D_S = \mathsf{BE.DecGen}(\mathsf{mpk}_{\mathsf{BE}}, \mathsf{id}, D_{\mathsf{id}}, S)$. Then $\mathcal{B}$ computes the combined decryptor $D \leftarrow \mathsf{EC.Extract}, (E_S, E_{\mathsf{vk}}, 1, D_{\mathsf{vk}})$, and returns $D(E(m))$ to $\mathcal{A}_1$. Hybrid 1 is indistinguishable from Hybrid 0 due to encryptor combiner correctness, which guarantees that as long as a valid decryptor is used to compute the combined decryptor, the combined decryptor is the same.

Define **Hybrid 2** to be the same as Hybrid 1, except we remove the check where $\mathcal{B}$ verifies $\mathsf{vk} \neq \mathsf{vk}^*$. We claim Hybrid 2 is indistinguishable from Hybrid 1. It suffices to consider cases where $\mathsf{vk} = \mathsf{vk}^*$. The first case to consider is if $\mathcal{A}_1$ submits a ciphertext $c = (E(m), \mathsf{hk}, \tau, \mathsf{vk}, \sigma)$ equal to the challenge ciphertext $c^* = (E^*(m), \mathsf{hk}^*, \tau^*, \mathsf{vk}^*, \sigma^*)$, which is allowable as long as $\mathcal{A}_1$ submits a set $S \neq S^*$. In this case, the checks will fail since UOWHF security guarantees that the probability $\mathsf{H.Eval}(\mathsf{hk}^*, S) = \mathsf{H.Eval}(\mathsf{hk}^*, S^*) = \tau^*$ for any $S \neq S^*$ that $\mathcal{A}_1$ picks is negligible. Thus, the ciphertext $\mathcal{A}_1$ submits cannot be equal to the challenge ciphertext. So either one of $(E(m), \mathsf{hk}, \tau) \neq (E^*(m), \mathsf{hk}^*, \tau^*)$ or $\sigma \neq \sigma^*$ must be the case. However, strong one-time security of the digital signature scheme guarantees that $\mathcal{A}_1$ cannot produce such a message/signature pair except with negligible probability.

The view of $\mathcal{A}_1$ in Hybrid 2 is exactly the view of $\mathcal{A}_1$ in the CCA broadcast game. Thus, our starting assumption implies that $\mathcal{A}_1$ obtains a non-negligible advantage in this game, and thus a non-negligible advantage in Hybrid 0. $\mathcal{B}$ then obtains the same advantage and breaks encryptor combiner security.

□

The above construction and proof result in a statically secure scheme, even if the CPA broadcast was adaptively secure. The difficulty lies in the fact that our encryptor combiner is static, and the parameters are generated during setup. This means the adversary's queries can potentially depend on the combiner parameters, making a reduction to static security impossible.

We can extend this construction to adaptive CCA-security as follows. Just like in the adaptive CPA broadcast construction, we have the encryptor combiner parameters generated during encryption time and send as part of the ciphertext. The problem now is that the adversary can potentially make decryption queries on ciphertexts containing malformed encryptor parameters. For such malformed parameters, independence may not hold, which breaks the proof above. We therefore apply a NIZK proof of well-formedness to the encryptor combiner parameters to guarantee that the combiner satisfies the necessary independence requirements. Proving security with these modifications is straightforward and we omit the details.

6 Non-Interactive Key Exchange

6.1 Multiparty NIKE from Private Encryptor Combiners

A Multiparty Non-interactive Key Exchange scheme consists of the following PPT algorithms.

- $\mathsf{Setup}(G, N)$: This algorithm takes as input G, the maximum number of users that can derive a shared key, and N, the maximum number of users in the scheme. It outputs public parameters params.
- $\mathsf{Publish}(\mathsf{params}, i)$: Each user $i \in [N]$ runs this algorithm, which takes as input the public parameters params and the user's index i. It outputs the user's secret key sk_i, and a public value pv_i, which the user publishes.
- $\mathsf{KeyGen}(\mathsf{params}, i, \mathsf{sk}_i, S, \{\mathsf{pv}_j\}_{j \in S})$: Each user i trying to derive a shared key runs this algorithm, which takes as input public parameters params, the user's index, the set $S \subseteq [N]$ of size at most G, and the set of public values $\{pv_j\}_{j \in S}$ of the users in S. It outputs a shared key k_S.

For correctness, we require that each user derives the same shared key. This means that for all $S \subseteq N$, $|S| \leq G$, $i, i' \in S$,

$$\mathsf{KeyGen}(\mathsf{params}, i, \mathsf{sk}_i, S, \{pv_j\}_{j \in S}) = \mathsf{KeyGen}(\mathsf{params}, i', \mathsf{sk}_{i'}, S, \{pv_j\}_{j \in S}).$$

We first define semi-static security using the definition of Boneh et al [BZ14]. For $b = 0, 1$, denote by $\texttt{SS-NIKE-EXP}(b)$ the following experiment.

$\hat{S} \leftarrow \mathcal{A}$
$\mathsf{params} \leftarrow \mathsf{Setup}(G, N)$
$b' \leftarrow \mathcal{A}^{\mathsf{Reg}(\cdot),\mathsf{RegCor}(\cdot),\mathsf{Ext}(\cdot),\mathsf{Rev}(\cdots),\mathsf{Test}(\cdots)}(G, N, \mathsf{params})$

- $\mathsf{Reg}(i \in [N])$ takes as input an index i and registers an honest user labelled i. It runs $(\mathsf{sk}_i, \mathsf{pv}_i) \leftarrow \mathsf{Publish}(\mathsf{params}, i)$, sends pv_i to $\mathcal{A}$, and records $(i, \mathsf{sk}_i, \mathsf{pv}_i, honest)$.
- $\mathsf{RegCor}(i \in [N], \mathsf{pv}_i)$ takes as input an index i and a public key pv_i, and records $(i, \perp, \mathsf{pv}_i, corrupt)$. The adversary may submit the same identity i more than once, in which case the challenger only saves the most recent tuple. Each query to RegCor and Ext must be on a user i outside the set $\hat{S}$.
- $\mathsf{Ext}(i)$ takes as input an index i and extracts the secret key for the honest user i. The challenger looks up the tuple $(i, \mathsf{sk}_i, \mathsf{pv}_i, honest)$ and returns sk_i to $\mathcal{A}$.
- $\mathsf{Rev}(S, i)$ reveals the shared secret for a group $S \subseteq [N] \leq G$ of users as calculated by the ith user, where $i \in S$. i must be honest. The challenger uses sk_i to derive the shared secret key k_S. Each query to Rev must have $S \not\subseteq \hat{S}$.
- $\mathsf{Test}(S)$ takes as input a set $S \subseteq N, |S| \leq G$ consisting solely of honest users. If $b = 0$, the challenger arbitrarily chooses a user to generate a shared secret key and gives this to the adversary. If $b = 1$, the challenger generates a random key for the adversary. Each query to Test must be on a subset S of $\hat{S}$.

For $b = 0, 1$, let W_b be the event that $b' = 1$ in $\texttt{SS-NIKE-EXP}(b)$. Define $\texttt{SS-NIKE-EXP}^{\text{adv}} = |Pr[W_0] - Pr[W_1]|$.

Definition 25. *A multiparty NIKE protocol is semi-statically secure if for all PPT adversaries A,* $\texttt{SS-NIKE-EXP}^{adv}$ *is negligible.*

To define static security, we remove the $\mathsf{Reg}, \mathsf{RegCor}, \mathsf{Ext}$, and Rev queries. $\mathcal{A}$ commits to S^* before seeing the public parameters, and is only allowed one query on Test, which must be on S^*.

We give the following multiparty NIKE construction using a private encryptor combiner consisting of $(\mathsf{EC.Setup}, \mathsf{EC.Extract})$ and a public key encryption scheme consisting of $(\mathsf{PK.KeyGen}, \mathsf{PK.Enc}, \mathsf{PK.Dec})$.

- $\mathsf{Setup}(G, N)$ takes as input the maximum number of users N and the maximum number of users that can derive a shared key, G, and runs $\mathsf{params} \leftarrow \mathsf{EC.Setup}$ and outputs encryptor combiner parameters params.
- $\mathsf{Publish}(\mathsf{params}, i)$. User i runs $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PK.KeyGen}()$, and publishes pk_i.

- $\mathsf{KeyGen}(\mathsf{params}, i, \mathsf{sk}_i, S, \{\mathsf{pv}_j\}_{j\in S})$. For each $j \in S$, let $E_j(\cdot) = \mathsf{Enc}(\mathsf{pv}_j, \cdot)$, and let $D_j(\cdot) = \mathsf{Dec}(\mathsf{sk}_j, \cdot)$. Set $k_S \leftarrow \mathsf{EC.Extract}(\mathsf{params}, \{E_j\}_{j\in S}, i, D_i)$.

Theorem 11. *If the private encryptor combiner is statically secure, then this multiparty NIKE protocol is statically secure.*

Proof. This proof is straightforward from private encryptor combiner security. Suppose an adversary $\mathcal{A}$ breaks the static security of this scheme. We construct an adversary $\mathcal{B}$ that runs $\mathcal{A}$ and breaks the static security of the private encryptor combiner. $\mathcal{B}$ simulates $\mathcal{A}$, who commits to a set S^*. Let $|S^*| = k$. Let Dist be a distribution on $(\{E_i\}_{i\in[k]}, \mathsf{aux})$ such that $\mathsf{Dist}_i = (E_i, \mathsf{aux}_i = \{E_1, \ldots, E_{i-1}, E_{i+1}, \ldots, E_k, \mathsf{aux})$ is secure for all $i \in [k]$. $\mathcal{B}$ should not be able to attain a non-neglible advantage in the private encryptor combiner static security experiment involving Dist. In this experiment, $\mathcal{B}$ receives $\mathsf{EC.params}, m_b, \{E_i\}_{i\in[k]}, \mathsf{aux}$ from the private encryptor combiner challenger. $\mathcal{B}$ sets NIKE $\mathsf{params} = \mathsf{EC.params}$, and forwards G, N, params to $\mathcal{A}$. When $\mathcal{A}$ makes the single Test query on S^*, $\mathcal{B}$ returns m_b to $\mathcal{A}$. Finally, $\mathcal{A}$ makes a guess b' for b. $\mathcal{B}$ forwards $\mathcal{A}$'s guess and attains a non-negligible advantage in the static private encryptor combiner experiment, which is a contradiction. □

Theorem 12. *If the private encryptor combiner is adaptively secure, then this multiparty NIKE protocol is semi-statically secure.*

Proof. First, we assume that only one query to Test is made. To justify this assumption, we can use a simple hybrid argument: an adversary $\mathcal{A}$ that makes q Test queries and obtains advantage ϵ can be used to produce an adversary $\mathcal{A}'$ that makes 1 test query and obtains advantage $\frac{\epsilon}{q}$.

Suppose an adversary $\mathcal{A}$ breaks the semi-static security of this scheme. We construct an adversary $\mathcal{B}$ that runs $\mathcal{A}$ as a subroutine and breaks the adaptive security of the private encryptor combiner. $\mathcal{B}$ simulates $\mathcal{A}$, who commits to a set $\hat{S}$. Let $|\hat{S}| = k$. Let Dist be a distribution on $(\{E_i\}_{i\in[k]}, \mathsf{aux})$ such that $\mathsf{Dist}_i = (E_i, \mathsf{aux}_i = \{E_1, \ldots, E_{i-1}, E_{i+1}, \ldots, E_{\hat{S}}, \mathsf{aux})$ is secure for all $i \in [k]$. These encryptors will correspond to the encryptors for users in $\hat{S}$.

$\mathcal{B}$ should not be able to attain a non-neglible advantage in the private encryptor combiner adaptive security experiment involving Dist. However, $\mathcal{B}$ can do so as follows. For the remaining users $i \notin \hat{S}$, $\mathcal{B}$ generates $(\mathsf{pv}_i, \mathsf{sk}_i) \leftarrow \mathsf{PK.KeyGen}()$. In the private encryptor combiner experiment, $\mathcal{B}$ receives $\mathsf{EC.params}, m_b, \{E_i\}_{i\in[k]}, \mathsf{aux}$ from the challenger. $\mathcal{B}$ sets NIKE $\mathsf{params} = \mathsf{EC.params}$, and then sends G, N, params to $\mathcal{A}$. It is straightforward to verify that $\mathcal{B}$ can answer any query $\mathcal{A}$ makes to $\mathsf{Reg}(\cdot), \mathsf{RegCor}(\cdot), \mathsf{Ext}(\cdot), \mathsf{Rev}(\cdots)$, given access to all secret keys outside of $\hat{S}$. When $\mathcal{A}$ makes the single Test query on $S \subseteq \hat{S}$, $\mathcal{B}$ forwards S to the private encryptor combiner challenger. $\mathcal{B}$ receives m_b in return, and forwards it to $\mathcal{A}$. Finally, $\mathcal{A}$ makes a guess b' for b. $\mathcal{B}$ forwards $\mathcal{A}$'s guess and breaks the adaptive security of the private encryptor combiner, which is a contradiction. □

7 Lattice-Based Encryptor Combiners

7.1 A Relaxed Notion of Encryptor Combiners

Here, we describe a relaxed notion of encryptor combiners. Let $\mathcal{E}$ be a class of encryptors, $\mathcal{D}$ a class of decryptors, and $\mathcal{R} : \mathcal{E} \times \mathcal{D} \to \{0, 1\}$ a relation on encryptors and decryptors. Let $\mathcal{P}$ be a class of probability distributions over $\mathcal{E}^n \times \mathcal{A} \times \mathcal{O}$, where $\mathcal{A}$ is a class of auxiliary information and $\mathcal{O}$ is a class of oracles.

Definition 26. *A $(\mathcal{E}, \mathcal{D}, \mathcal{R}, \mathcal{P})$ public encryptor combiner is an encryptor combiner where correctness only holds for decryptors in $\mathcal{D}$ satisfying the relation $\mathcal{R}$, and security only holds with respect to distributions $\mathsf{Dist} \in \mathcal{D}$.*

Our notion of encryptor combiners presented in Section 3 corresponds to $\mathcal{E}$ consisting of any encryptors of a priori bounded size, $\mathcal{D}$ being the set of all functions, $\mathcal{R}$ being the relation specifying valid encryptor/decryptor pairs, and $\mathcal{P}$ be the class of distributions Dist for which every encryptor is individually secure. Our relaxed notion above allows for placing restrictions on the types of encryptors, decryptors, relations, and distributions over which the encryptor combiner operates.

7.2 Background on Lattices

We defer a comprehensive discussion of lattices to prior works such as [GPV08], and here only discuss the basics needed to give our ideas.

Let $q \geq 2$ an integer, n a security parameter, and $m = poly(n)$ be an integer. Let χ_σ be the "discrete Gaussian" over $\mathbb{Z}$ of width σ.

Definition 27 (Learning With Errors)**.** *For a distribution χ over $\mathbb{Z}$, the* LWE *distribution* LWE_χ *is the distribution over $\mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$ defined as follows:*

- *Choose a random matrix $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$.*
- *Choose a random error vector $\mathbf{e} \leftarrow \chi^m$*
- *Output $(\mathbf{A}, \mathbf{b}^T = \mathbf{s}^T\mathbf{A} + \mathbf{e}^T \mod q)$*

The LWE *problem is then to distinguish between* LWE_χ *from the uniform distribution over $\mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$. The* LWE *problem is hard if, for all efficient adversaries A, the probability A distinguishes these two cases is negligible in n.*

We will consider LWE for the case where $\chi = \chi_\sigma$.

We will also consider a relaxed version of Learning with Errors where the distribution of $\mathbf{A}$ is perhaps non-uniform, and moreover there may be additional auxiliary informaiton that an adversary can use to try to solve the LWE instance.

Definition 28 (Generalized Learning With Errors)**.** *Fix a distribution* $\mathsf{Dist}(n)$ *over matrices $\mathbf{A} \in \mathbb{Z}_q^{n,m}$, auxiliary information* aux*, and oracles $\mathcal{O}$. For a distribution χ over $\mathbb{Z}$, the* LWE *distribution* $\mathsf{LWE}_{\mathsf{Dist},\chi}$ *is the distribution over $\mathbf{A}, \mathsf{aux}, \mathcal{O}, \mathbf{b}$ defined as follows:*

- *Sample $\mathbf{A}, \mathsf{aux}, \mathcal{O} \leftarrow \mathsf{Dist}(n)$*
- *Choose a random error vector $\mathbf{e} \leftarrow \chi^m$*
- *Output $(\mathbf{A}, \mathsf{aux}, \mathcal{O}, \mathbf{b}^T = \mathbf{s}^T\mathbf{A} + \mathbf{e}^T \mod q)$*

The generalized LWE *problem is then to distinguish between* $\mathsf{LWE}_{\mathsf{Dist},\chi}$ *from $(\mathbf{A}, \mathsf{aux}, \mathcal{O}, \mathbf{b}^T \mod q)$ where $\mathbf{b}$ is samples uniformly from $\mathbb{Z}_q^m$. The generalized* LWE *problem is hard if, for all efficient adversaries $\mathcal{A}$, the probability $\mathcal{A}$ distinguishes these two cases is negligible in n.*

7.3 Dual Regev Encryptor

Here, we present a variant of the Dual Regev encryption scheme [GPV08]. Let χ be some distribution over $\mathbb{Z}$ that is guaranteed to be bounded by B (at least with overwhelming probability in n). For example, if $\chi = \chi_\sigma$, the discrete Gaussian of width σ, then except with negligible probability, a sample from χ_σ is bounded by $B = \sigma\omega(\sqrt{n})$.

The encryptor is specified by a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$. To encrypt a bit $b \in \{0, 1\}$, do the following. If $b = 1$, output a uniform random vector $\mathbf{c} \leftarrow \mathbb{Z}_q^m$. If $b = 0$, instead choose a uniform random vector $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ and "short" vector $\mathbf{e} \leftarrow \chi^m$. Output $\mathbf{c}^T = \mathbf{s}^T\mathbf{A} + \mathbf{e}^T \mod q$. Notice that for a random $\mathbf{A}$, $b = 0$ corresponds to a sample from the LWE distribution LWE_χ, whereas $b = 1$ is a uniform sample. The LWE assumption implies that these distributions are hard to distinguish, meaning the encryptor is secure.

A decryptor is specified by a "short" full-rank (over $\mathbb{Q}$) matrix $\mathbf{T} \in \mathbb{Z}^{m \times m}$ such that $\mathbf{A} \cdot \mathbf{T} \mod q = 0$. Actually, for reasons that will become apparent later, we want a slightly stronger requirement that the columns of $\mathbf{T}$ to be a basis for the lattice of vectors $\mathbf{s} \in \mathbb{Z}^m$ where $\mathbf{A} \cdot \mathbf{s} \mod q = 0$.

If $\mathbf{c}$ is an encyption of 1 (meaning it is uniformly random in $\mathbb{Z}_q^m$), then $\mathbf{c}^T \cdot \mathbf{T} \mod q$ will also be uniformly random in $\mathbb{Z}_q^m$, and hence it will not be short, except with exponentially small probability. If $\mathbf{c}$

is an encryption of 0, so that $\mathbf{c}^T = \mathbf{s}^T\mathbf{A} + \mathbf{e}^T \mod q$, then $\mathbf{c} \cdot \mathbf{T} \mod q = \mathbf{e}^T \cdot \mathbf{T} \mod q$. Notice that, since $\mathbf{T}$ and $\mathbf{e}$ are short, so is there product, and so $\mathbf{e}^T \cdot \mathbf{T}$ does not wrap around mod q. Therefore, $\mathbf{c} \cdot \mathbf{T} \mod q = \mathbf{e}^T \cdot \mathbf{T}$, where the right hand side is *not* reduced mod q. Therefore, we can multiply by $\mathbf{T}^{-1}$ over $\mathbb{Q}$ (which is possible since $\mathbf{T}$ has full rank over $\mathbb{Q}$) to recover $\mathbf{e}$. Then we can simply test if $\mathbf{e}$ is short.

Therefore, the decryptor does the following. Compute $(\mathbf{c}^T \cdot \mathbf{T} \mod q) \cdot \mathbf{T}^{-1}$, and output 0 if any only if *all* components are bounded by B. By the analysis above, with overwhelming probability this decryptor outputs the correct answer.

It is possible to generate a (statistically close to) random Dual Regev encryptor $\mathbf{A}$ together with a decryptor $\mathbf{T}$ using [Ajt96].

7.4 Encryptor Combiners for Dual Regev Encryption

Let $\mathcal{E}$ be the set of dual Regev encryptors specified by matrices $\mathbf{A}$. Let $\mathcal{D}$ be the set of matrices, and $\mathcal{R}$ specifies that $\mathcal{D}$ is of dimension $m \times m$ and is "short", full-rank (over $\mathbb{Q}$), and satisfies $\mathbf{A} \cdot \mathbf{T} \mod q = 0$. Let $\mathcal{P}_\ell$ be the class of distributions Dist_ℓ for matrices $\mathbf{B} \in \mathbb{Z}_q^{n \times (m\ell)}$, auxiliary information aux and oracles $\mathcal{O}$ such that the generalized LWE problem is hard. We can think of $\mathbf{B}$ as consisting of ℓ separate $n \times m$ matrices $\mathbf{A}_1, \ldots, \mathbf{A}_\ell$.

Here, we show how to combine Dual Regev encryptors. There is no Gen procedure.

Combine. Given a set of encryptors $A_1, \ldots, A_\ell \in \mathbb{Z}_q^{n \times m}$, the combined encryptor A is simply the concatenation of each of the A_i:

$$\mathbf{A}^* = [\mathbf{A}_1 | \mathbf{A}_2 | \cdots | \mathbf{A}_n]$$

Extract. To extract, we are given a short full-rank matrix $\mathbf{T}_i$ such that $\mathbf{A}_i \cdot \mathbf{T}_i \mod q = 0$, and we wish to devise a short full-rank matrix $\mathbf{T}^*$ such that $\mathbf{A}^* \cdot \mathbf{T}^* \mod q = 0$. This can be done using the basis extension techniques of Cash et al. [CHKP10] and Agrawal et al.[ABB10a]. We note that the distribution on $\mathbf{T}^*$ has the property that it is (statistically) independent of the index i and the matrix $\mathbf{T}_i$ used to generate it.

Security. We now show the *adaptive* security of this encryptor combiner for Dual Regev Encryptors. Suppose towards contradiction there is an adversary $\mathcal{A}$ that can wins the following game. Encryptors $\mathbf{A}_1, \ldots, \mathbf{A}_\ell$ are sampled from Dist_ℓ, along with $\mathsf{aux}, \mathcal{O}$, and given to $\mathcal{A}$. $\mathcal{A}$ then chooses an arbitrary set $S \subseteq [\ell]$. Let

$$\mathbf{A}_S^* = [\mathbf{A}_i]_{i \in S}$$

be the concatenation of the matrices in S. Then $\mathcal{A}$ breaks the security of the dual Regev Encryptor specified by $\mathbf{A}_S^*$. Specifically, $\mathcal{A}$ can distinguish $\mathbf{c}^T = \mathbf{s}^T\mathbf{A}_S^* + \mathbf{e}^T \mod q$ for a "short" vector $\mathbf{e}$ from a uniformly random $\mathbf{c}$, potentially using aux and $\mathcal{O}$.

We now show how to use such an $\mathcal{A}$ to break the generalized LWE assumption for the distribution $\mathcal{P}_\ell$. We are given $\mathbf{B}, \mathsf{aux}, \mathcal{O}, \mathbf{b}$. Write $\mathbf{b} = [\mathbf{c}_1, \ldots, \mathbf{c}_\ell]$. Then either $\mathbf{c}_i = \mathbf{s}\mathbf{A}_i + \mathbf{e}_i$ for random $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ and $\mathbf{e}_i \leftarrow \xi^m$, or each of the $\mathbf{c}$ are uniformly random, and our goal is to distinguish the two cases. To that end, forward $\mathbf{B}, \mathsf{aux}, \mathcal{O}$ to $\mathcal{A}$. Then $\mathcal{A}$ responds with S, let $\mathbf{c}^T = [\mathbf{c}_i^T]_{i \in S}$. If $\mathbf{b}$ is an LWE samplem then $\mathbf{c}$ is an encryption of 0, and if $\mathbf{b}$ is uniformly random, the $\mathbf{c}$ is an encryption of 1. Therefore, since $\mathcal{A}$ distinguishes these two cases, we can use the output of $\mathcal{A}$ to break the generalize LWE problem.

We therefore have the following theorem:

Theorem 13. *The Dual Regev encryptor combiner above is a $(\mathcal{E}, \mathcal{D}, \mathcal{P}_\ell)$ encryptor combiner for any polynomial ℓ. It is unbounded and adaptively secure, and satisfies distributional independence.*

7.5 Application: Hierarchical Identity-Based Encryption

We can use the Dual Regev combiner together with Dual Regev encryption and the construction in Section 4 to get an identity-based encryption scheme. Since the combiner is adaptively secure, the resulting IBE is also adaptively secure using Theorem 5. The concatenation of all encryptors is always a uniform random matrix; as such, security relies on the standard LWE assumption.

Notice that the combined encryptor itself is a Dual Regev encryptor. As such, we can plug the combiner into the hierarchical identity-based encryption scheme, which again is adaptively secure assuming hardness of LWE, as long as the number of levels in the hierarchy is constant.

We note that many prior works construct lattice (H)IBE in the standard model [CHKP10, ABB10a, ABB10b]. Our scheme is not superior to these schemes; in fact, our scheme is similar in spirit to the scheme of [CHKP10]. However, the advantage of our approach is in its conceptual unification, bringing constructions from very different tools — lattices and obfuscation — under the common framework of encryptor combiners.

References

[ABB10a] Shweta Agrawal, Dan Boneh, and Xavier Boyen. *Efficient Lattice (H)IBE in the Standard Model*, pages 553–572. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[ABB10b] Shweta Agrawal, Dan Boneh, and Xavier Boyen. *Lattice Basis Delegation in Fixed Dimension and Shorter-Ciphertext Hierarchical IBE*, pages 98–115. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[Ajt96] M. Ajtai. Generating hard instances of lattice problems (extended abstract). In *Proceedings ofn the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pages 99–108, New York, NY, USA, 1996. ACM.

[BB04] Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In *Annual International Cryptology Conference*, pages 443–459. Springer, 2004.

[BST14] Mihir Bellare, Igors Stepanovs, and Stefano Tessaro. *Poly-Many Hardcore Bits for Any One-Way Function and a Framework for Differing-Inputs Obfuscation*, pages 102–121. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[BZ14] Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In *International Cryptology Conference*, pages 480–499. Springer, 2014.

[CHK04] Ran Canetti, Shai Halevi, and Jonathan Katz. *Chosen-Ciphertext Security from Identity-Based Encryption*, pages 207–222. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[CHKP10] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. *Bonsai Trees, or How to Delegate a Lattice Basis*, pages 523–552. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[CLT13] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. *Practical Multilinear Maps over the Integers*, pages 476–493. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. *Candidate Multilinear Maps from Ideal Lattices*, pages 1–17. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[GGH+13b] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *Proceedings of the 2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, FOCS '13, pages 40–49, Washington, DC, USA, 2013. IEEE Computer Society.

[GGH15] Craig Gentry, Sergey Gorbunov, and Shai Halevi. *Graph-Induced Multilinear Maps from Lattices*, pages 498–527. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, August 1986.

[GPSZ16] Sanjam Garg, Omkant Pandey, Akshayaram Srinivasan, and Mark Zhandry. Breaking the sub-exponential barrier in obfustopia. Technical report, Cryptology ePrint Archive, Report 2016/102, 2016. http://eprint. iacr. org/2016/102, 2016.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, pages 197–206, New York, NY, USA, 2008. ACM.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, March 1999.

[HJK+14] Dennis Hofheinz, Tibor Jager, Dakshita Khurana, Amit Sahai, Brent Waters, and Mark Zhandry. How to generate and use universal samplers. Technical report, Cryptology ePrint Archive, Report 2014/507, 2014. http://eprint. iacr. org/2014/507, 2014.

[Lam79] Leslie Lamport. Constructing digital signatures from a one way function. Technical report, October 1979.

[Rom90] J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the Twenty-second Annual ACM Symposium on Theory of Computing*, STOC '90, pages 387–394, New York, NY, USA, 1990. ACM.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: Deniable encryption, and more. In *Proceedings of the Forty-sixth Annual ACM Symposium on Theory of Computing*, STOC '14, pages 475–484, New York, NY, USA, 2014. ACM.

[Zha14] Mark Zhandry. Adaptively secure broadcast encryption with small system parameters, 2014.

[Zha16] Mark Zhandry. *The Magic of ELFs*, pages 479–508. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016.