Adaptively-Secure Big-Key Identity-Based Encryption

Jeffrey Champion
UT Austin
jchampion@utexas.edu

Brent Waters
UT Austin and NTT Research
bwaters@cs.utexas.edu

David J. Wu
UT Austin
dwu4@cs.utexas.edu

Abstract

Key-exfiltration attacks on cryptographic keys are a significant threat to computer security. One proposed defense against such attacks is big-key cryptography which seeks to make cryptographic secrets so large that it is infeasible for an adversary to exfiltrate the key (without being detected). However, this also introduces an inconvenience to the user who must now store the large key on all of their different devices. The work of Döttling, Garg, Sekar and Wang (TCC 2022) introduces an elegant solution to this problem in the form of big-key *identity-based* encryption (IBE). Here, there is a large master secret key, but very short identity keys. The user can now store the large master secret key as her long-term key, and can provision each of her devices with short ephemeral identity keys (say, corresponding to the current date). In this way, the long-term secret key is protected by conventional big-key cryptography, while the user only needs to distribute short ephemeral keys to their different devices. Döttling et al. introduce and construct big-key IBE from standard pairing-based assumptions. However, their scheme only satisfies *selective* security where the adversary has to declare its challenge set of identities at the beginning of the security game. The more natural notion of security is *adaptive* security where the user can adaptively choose which identities it wants to challenge *after* seeing the public parameters (and part of the master secret key).

In this work, we give the first adaptively-secure construction of big-key IBE from standard cryptographic assumptions. Our first construction relies on indistinguishability obfuscation (and one-way functions), while our second construction relies on witness encryption for NP together with standard pairing-based assumptions (i.e., the SXDH assumption). To prove adaptive security, we show how to implement the classic dual-system methodology with indistinguishability obfuscation as well as witness encryption.

1 Introduction

Security breaches are increasingly common today, and one of the highest-value targets in a security breach are the cryptographic keys residing on a user's system. Once an adversary successfully recovers a user's secret cryptographic key, they gain the ability to decrypt all of the user's potentially sensitive data and can even impersonate the user to other clients. This problem is further aggravated when using more advanced encryption systems such as identity-based encryption (IBE) [BF01, Coc01] where a central authority holds on to a long-term secret key. Such systems introduce a single point of failure and if the central authority's single long-term secret key is compromised, then the adversary breaks security for *all* of the users in the system.

Cryptography in the bounded retrieval model. One proposal to mitigate the threat of a key-exfiltration attack is to make it difficult or infeasible for the adversary to exfiltrate the secret key. This has motivated the "bounded-storage model" and the concept of "big-key" cryptography [Dzi06, CLW06, CDD+07, ADW09, ADN+10, BKR16, MW20, DGSW22]. Here, the idea is to make the cryptographic keys sufficiently large that key exfiltration becomes infeasible to an adversary that only has a bounded amount of storage. In practice, the bounded storage might translate to an adversary being able to retrieve a bounded number of bits from a compromised system before the adversary is detected and its access removed.

A number of works have studied constructions of big-key public-key encryption in the bounded storage model [ADN+10, MW20]. In these settings, the goal is to have a large secret key (which is hard to exfiltrate) and a *short* public key. Moreover, the honest user should not incur the penalty of having to manipulate a large cryptographic key. In particular, encryption and decryption should both be fast; in the case of decryption, the idea is that the decryption algorithm only needs random access to a few *ciphertext-dependent* bits of the secret key to decrypt. The main security requirement is semantic security (for a fresh ciphertext) should hold even if the adversary gets arbitrary *bounded* leakage on the large secret key. As discussed in [DGSW22], a major disadvantage of this model is the fact that the large secret key has to be replicated to each of the user's devices. This can impose significant storage burdens for each device that needs a copy of the secret key.

IBE with incompressible master secret key. Döttling, Garg, Sekar, and Wang [DGSW22] propose an elegant solution to the problem of needing to replicate the large secret key to each device owned by the user. They introduce the notion of a big-key IBE scheme where there is a long incompressible master secret key, but *short* identity keys. Recall first that in an IBE scheme, both secret keys and ciphertexts are associated with an identity id and decryption succeeds (i.e., recovers the plaintext associated with the ciphertext) if the identities associated with the ciphertext matches that of the decryption key. In the setting envisioned by [DGSW22], the long-term key would be the large master secret key for the IBE scheme. Each ciphertext in the system would be encrypted to an identity that identifies a particular time window (e.g., the current date). Users would provision each of their devices with the identity keys for the time intervals of interest. These ephemeral keys are identity keys and thus, are short. Moreover, if an identity key is compromised, it only compromises the security of messages tagged with that particular time window. In a sense, the individual identity keys in the system are viewed as short ephemeral keys while the long-term key is the *large* master secret key for the IBE scheme. Importantly, in this model, the user only needs to store one copy of the long-term master secret key; each of the user's devices would only need to store ephemeral identity keys.

The challenge: adaptive security. In the same work, Döttling et al. [DGSW22] showed how to construct a big-key IBE scheme from standard assumptions on groups with bilinear maps. One limitation of their system is it only provides *selective* security. Namely, the adversary in the IBE security game must pre-declare the set of identities it wants to target at the beginning of the security game (before it sees the public key of the scheme or makes key-generation queries). This is in contrast to the more natural notion of adaptive (or full) security where the adversary can adaptively choose which identities it wants to target *after* it sees the public parameters as well as its choice of leakage on the master secret key. Their work leaves open the question of constructing a big-key IBE scheme with adaptive security.

This work. In this work, we give two constructions of adaptively-secure big-key IBE schemes from standard assumptions. Our first construction relies on indistinguishability obfuscation [BGI+01, GGH+13] (and one-way functions) while our second construction relies on witness encryption [GGSW13] for NP in conjunction with standard pairing-based assumptions. To prove adaptive security of our scheme, we rely on a dual-system proof [Wat09, LW10]. The intricacies of carrying out this dual system proof strategy (see Section 1.1) is a key reason why our approach relies on considerably stronger machinery (either indistinguishability obfuscation or witness encryption) compared to the previous selectively-secure construction. Along the way, we also highlight some issues in the previous definitions and analysis of big-key IBE; we provide a more detailed discussion of these definitional issues in Section 3.

1.1 Technical Overview

In this section, we provide a general overview of our main constructions of adaptively-secure big-key IBE from indistinguishability obfuscation and from witness encryption.

Identity-based encryption. We start by recalling the syntax of a standard identity-based encryption (IBE) scheme [BF01, Coc01]:

- **Setup:** The setup algorithm in an IBE scheme generates the public parameters pp and the master secret key msk for the scheme.
- **Key generation:** The key-generation algorithm takes the master secret key msk and an identity id, and outputs a secret key $\mathsf{sk}_{\mathsf{id}}$ for the particular identity.
- **Encryption:** The encryption algorithm takes the public parameters pp, an identity id, and a message m, and outputs a ciphertext ct.
- **Decryption:** The decryption algorithm takes a ciphertext ct (associated with an identity id and message m) together with a secret key $\mathsf{sk}_{\mathsf{id}'}$ (associated with an identity id') and either outputs the message m if $\mathsf{id} = \mathsf{id}'$ or $\perp$ if $\mathsf{id} \neq \mathsf{id}'$.

The semantic security requirement for an IBE scheme states that the adversary should not be able to distinguish between an encryption of m_0 from an encryption of m_1 for any challenge identity id for which it does not have the corresponding secret key.

Big-key IBE. In a big-key IBE scheme [DGSW22], the correctness requirement is the same as for vanilla IBE. However, the security requirement is modified to give the adversary (bounded) leakage on the master secret key:

- In the big-key security game, the adversary can specify any efficiently-computable leakage function f (with output length at most ℓ) and learn $f(\mathsf{msk})$. The output length $\ell \geq 0$ is the leakage parameter for the scheme.
- Next, instead of a single challenge identity, the adversary specifies a set of k challenge identities $\mathcal{J}$. To win, the adversary must break semantic security for *all* identities within the challenge set $\mathcal{J}$. Here, the parameter k is a function of the security parameter λ and the leakage length ℓ. In the adaptive security game, we allow the adversary to choose the set of identities $\mathcal{J}$ after it receives the public parameters, the leakage on msk, and after it makes key-generation queries on identities of its choosing (with the stipulation that the adversary does not make a key-generation query on any identity in the challenge set $\mathcal{J}$).

The adversary's task is *necessarily* harder in the big-key IBE security game compared to the vanilla IBE security game because the adversary must break semantic security of k identities rather than 1. This is inherent because the leakage function the adversary chooses can allow it to learn the secret keys for a handful of identities. The work of [DGSW22] consider the setting where $k = \ell + 1$; namely, if the adversary gets ℓ bits of leakage about the master secret key, then it wins only if it breaks semantic security on at least $k = \ell + 1$ identities.

In addition, for big-key IBE, we require that the running times of the key-generation, encryption, and decryption algorithms to be efficient and run in time that is $\mathsf{poly}(\lambda, \log \ell)$ on a RAM machine. Notably, while the length of the master secret key msk can (and necessarily) must grow with the leakage parameter ℓ, the key-generation algorithm should only read a few bits of msk to generate an identity key.

In this work, we will focus on the simpler setting where the length of the public parameters can also grow with the leakage size ℓ. However, we maintain the requirement that the encryption and decryption algorithms only need to read $\text{poly}(\lambda, \log \ell)$ bits of the long public key. Döttling et al. [DGSW22] showed how to use a non-interactive secure computation (NISC) scheme to generically transform a big-key IBE scheme with large public parameters (but fast encryption and decryption) into one with short public parameters. As we show in Appendix A, this transformation still preserves adaptive security. Thus, for the remainder of this overview (and throughout this work), we focus on the simpler setting of big-key IBE with long public keys.

The [DGSW22] approach. We begin with a brief description of the approach from [DGSW22]. Their scheme relies on a puncturable pseudo-entropy function (PEF). A PEF [BHK11] is a function whose outputs at certain inputs are statistically unpredictable even given leakage on the key to the PEF. The work of [DGSW22] show how to construct a PEF where the key consists of a large number of blocks $k = (k_1, \ldots, k_N)$ and moreover, the PEF supports *local* evaluation where the value of the PEF at an input x only depends on a small (and random-looking) subset of blocks of the secret key. Their construction then operates as follows:

- The master secret key consists of the PEF key $k = (k_1, \ldots, k_N)$ and the public parameters consist of commitments $\mathsf{pp} = (c_1, \ldots, c_N)$ to the blocks of the secret key.

- The secret key $\mathsf{sk}_{\mathsf{id}}$ for an identity id consists of the evaluation of the $y = \mathsf{PEF}(k, \mathsf{id})$ together with a non-interactive zero-knowledge (NIZK) proof that y was correctly computed with respect to the committed key $\mathsf{pp} = (c_1, \ldots, c_N)$. For this to be succinct, it is critical that the PEF is locally-computable (i.e., the output of $\mathsf{PEF}(k, \mathsf{id})$ only depend on k_i for some $i \in I_{\mathsf{id}} \subset [N]$, where $|I_{\mathsf{id}}| \ll N$).

- An encryption of a message to an identity id is essentially a witness encryption[1] of the message m for the relation that essentially checks that the decrypter possesses a valid NIZK proof that $y = \mathsf{PEF}(k, \mathsf{id})$ with respect to the (subset of) committed keys c_i for $i \in I_{\mathsf{id}}$. Here, the work of [DGSW22] shows that a special witness encryption scheme tailored for NIZK proofs on committed values [BL20] suffices, which can in turn be instantiated by standard pairing-based assumptions.

The proof of selective security then proceeds along the following lines:

- First, the identity keys consist of zero-knowledge proofs of openings to the commitments c_i. Thus, they hide the values of the actual bits k_i in the master secret key. The only leakage on the PEF key k is through the leakage function (applied to the master secret key $\mathsf{msk} = k$).

- Next, [DGSW22] rely on *puncturing*. Namely, they show how to puncture the PEF key at a set of identities $\mathcal{J}$ to obtain a punctured key $k_{\mathcal{J}}$. The property is that the punctured key $k_{\mathcal{J}}$ can be used to evaluate the PEF on all inputs $i \notin \mathcal{J}$ while the values on $\mathcal{J}$ retain high statistical min-entropy. The idea in the *selective* security proof is that the reduction algorithm will first puncture the PEF key on the challenge set $\mathcal{J}$. In this case, they can show that for every challenge set $\mathcal{J}$, there will exist at least one identity $\mathsf{id}^* \in \mathcal{J}$ such that the value of $y_{\mathsf{id}^*} = \mathsf{PEF}(k, \mathsf{id}^*)$ is statistically unpredictable to the adversary (even given the leakage on the PEF key). In combination with the security of the

[1]In a witness encryption scheme [GGSW13], one can encrypt a message m to an arbitrary NP statement x. To decrypt, one provides a witness w for the statement x. If the witness is valid, then decryption recovers the message m. If the statement x is false (i.e., no witness exists), then the ciphertext computationally hides the message m.

witness encryption, they can argue that such an adversary cannot have non-negligible advantage breaking semantic security with respect to id^*.

The use of puncturing means the reduction algorithm needs to know the challenge identities ahead of time in order to program them into the scheme parameters. It is unclear how to extend this approach to the adaptive setting where the reduction algorithm does not know in advance which identities the adversary might query.

While we can envision some type of partitioning strategy [BF01, Wat05] that has been successful for arguing adaptive security in the setting of plain IBE, it is less clear how to execute such a strategy in this setting. In plain IBE, there is just a single challenge identity, so the idea in the partitioning proof is to partition the identity space into two sets S, T, with the property that the reduction algorithm is able to generate secret keys for identities $\mathsf{id} \in S$ but not for identities $\mathsf{id} \in T$. The hope then is that the adversary's key-generation queries fall into set S while its challenge query falls into set T. If the adversary only makes a single challenge query, the reduction can choose S, T such that with inverse polynomial probability, all of the key-generation queries land in S while the single challenge query lands in T. In the big-key setting, the challenge is that the adversary now specifies a *set* $\mathcal{J}$ of challenge identities. For the adversary to be useful, we need to set up the reduction so that an adversary that succeeds in breaking semantic security for any identity $\mathsf{id} \in \mathcal{J}$ in the challenge set can be used to break the computational assumption. In this setting, we do not see a way to partition the identity space into sets S, T such that with good probability, all of the adversary's key-generation queries fall into S while all of the challenge queries fall into T. Thus, proving adaptive security will require a different proof technique.

Our approach. To prove adaptive security, we take a dual-system approach [Wat09, LW10]. Implementing a dual-system proof strategy will require additional machinery and as a result, our constructions either rely on indistinguishability obfuscation or general-purpose witness encryption. We begin by describing our basic template, which is a slimmed-down version of the construction from [DGSW22], where we no longer have a PEF:

- The master secret key is a random bit-string of length $N = \mathsf{poly}(\lambda, \ell)$: $\mathsf{msk} = (r_1, \ldots, r_N)$. The public parameters are commitments to the bits of the master secret key: $\mathsf{pp} = (c_1, \ldots, c_N)$.

- A secret key $\mathsf{sk}_{\mathsf{id}}$ for an identity id is a NIZK proof of the openings to the commitments c_i for $i \in I_{\mathsf{id}}$ where the subset I_{id} is derived from a hash function $I_{\mathsf{id}} = \mathcal{H}(\mathsf{id})$ on the identity. Note that the set of indices I_{id} is substantially smaller than N.

- To encrypt to an identity id, the encrypter prepares a program that takes as input a proof π and checks whether π is a NIZK proof of openings to the commitments c_i for all $i \in I_{\mathsf{id}} = \mathcal{H}(\mathsf{id})$. If so, the program outputs the message, and otherwise, it outputs $\perp$. Decryption just corresponds to evaluating the obfuscated program on the secret key. As we elaborate more below, the obfuscated program that checks the NIZK proof can be implemented using either indistinguishability obfuscation or using witness encryption.

Correctness follows immediately. Moreover, if the size of each set I_{id} is bounded by $\mathsf{poly}(\lambda, \log \ell)$, then the scheme also supports fast key-generation, encryption, and decryption. The high-level idea underlying security is similar to [DGSW22]. First, the identity keys consist of NIZK proofs of openings to the commitments c_i, so they hide the values of the actual bits r_i. Second, the only information the adversary gets on the master secret key then is its ℓ bits of leakage. Next, if the hash function $\mathcal{H}$ that maps identities id to indices I_{id} is "well-spread," then we can hope to argue that there is at least one identity id^* in the challenge set $\mathcal{J}$

for which the adversary does *not* know most of the bits of r_i for $i \in \mathcal{H}(\mathsf{id}^*)$. In this case, the adversary will not be able to construct a NIZK proof that it knows the openings to c_i for $i \in \mathcal{H}(\mathsf{id}^*)$. We can then hope to rely on security of the obfuscation scheme (or witness encryption scheme) to conclude that the message is hidden. We now show how to instantiate this basic template from indistinguishability obfuscation as well as from witness encryption in a way that allows us to prove *adaptive* security.

Big-key IBE from $i\mathcal{O}$. We first describe how to instantiate the above template using indistinguishability obfuscation in conjunction with the following primitives: (1) a plain (adaptively-secure) identity-based encryption scheme; (2) a NIZK proof system for NP; and (3) a (one-time) dual-mode bit commitment scheme [Nao89] (i.e., a commitment scheme where the common reference string can be sampled in one of two computationally indistinguishable modes: one mode is statistically binding while the other is statistically hiding). We then instantiate our template as follows:

- The master secret key $\mathsf{msk} = (r_1, \ldots, r_N)$ is a random bit string: $r_i \xleftarrow{\mathrm{R}} \{0,1\}$. The public parameters $\mathsf{pp} = (\mathsf{crs}_{\mathsf{Com}}, \mathsf{crs}_{\mathsf{NIZK}}, \mathsf{pp}_{\mathsf{IBE}}, c_1, \ldots, c_N)$ for the scheme contains the common reference string $\mathsf{crs}_{\mathsf{Com}}$ for the bit commitment scheme (in binding mode), the common reference string $\mathsf{crs}_{\mathsf{NIZK}}$ for the NIZK, the public parameters $\mathsf{pp}_{\mathsf{IBE}}$ for the plain IBE scheme, and commitments c_i to the bits r_i. The master secret key msk also contains the openings to the commitments $\sigma_1, \ldots, \sigma_N$.

- The secret key for an identity id is $\mathsf{sk}_{\mathsf{id}} = (\mathsf{ct}_{\mathsf{IBE}}, \pi)$. Here,
$$\mathsf{ct}_{\mathsf{IBE}} = \mathsf{IBE.Encrypt}(\mathsf{pp}_{\mathsf{IBE}}, \mathsf{id}, \vec{r}_{\mathsf{id}}; \rho_{\mathsf{enc}}) \tag{1.1}$$
is an IBE ciphertext that encrypts the tuple of bits $\vec{r}_{\mathsf{id}} = (r_i)_{i \in I_{\mathsf{id}}}$ of the secret key indexed by $I_{\mathsf{id}} = \mathcal{H}(\mathsf{id})$, where $\mathcal{H}$ is a (fixed) hash function that maps identities onto a set of indices. We let ρ_{enc} denote the encryption randomness. In addition, the secret key contains a NIZK proof π that the commitments $\vec{c}_{\mathsf{id}} = (c_i)_{i \in I_{\mathsf{id}}}$ in pp is a valid commitment to $\vec{r}_{\mathsf{id}}$ and that ct is an encryption of $\vec{r}_{\mathsf{id}}$ with randomness ρ_{enc}. Here, the statement in the NIZK proof is $(\mathsf{id}, \vec{c}_{\mathsf{id}}, \mathsf{ct}_{\mathsf{IBE}})$ and the witness is $(\vec{r}_{\mathsf{id}}, \rho_{\mathsf{enc}}, \vec{\sigma}_{\mathsf{id}})$, where $\vec{\sigma}_{\mathsf{id}} = (\sigma_i)_{i \in I_{\mathsf{id}}}$.

- To encrypt a message m to an identity id^*, the encrypter computes an obfuscation of a program P that has the identity id^*, the message m, the common reference string $\mathsf{crs}_{\mathsf{NIZK}}$, and the commitments $\vec{c}_{\mathsf{id}^*}$ hard-wired inside. The program takes as input a secret key $\mathsf{sk}_{\mathsf{id}} = (\mathsf{ct}_{\mathsf{IBE}}, \pi)$ and outputs the message if π is a valid proof for the statement $(\mathsf{id}^*, \vec{c}_{\mathsf{id}^*}, \mathsf{ct}_{\mathsf{IBE}})$. Otherwise, the program outputs $\perp$.

Proving adaptive security via a dual-system approach. As mentioned before, we leverage a dual-system strategy [Wat09, LW10] to prove that the above scheme is *adaptively* secure. In a dual-system proof, we define a sequence of hybrid experiments where we gradually replace the normal ciphertexts and secret keys (given out in the security game) with "semi-functional" analogs. The invariant we enforce is that normal keys can decrypt semi-functional ciphertexts and semi-functional keys can decrypt normal ciphertexts. However, semi-functional keys are unable to decrypt semi-functional ciphertexts, and moreover, the adversary is unable to tell whether a key or ciphertext is normal or semi-functional. In particular, this means that it should be hard for an adversary to generate semi-functional ciphertexts on its own (if it could, then it could trivially distinguish semi-functional keys for normal keys). In the final hybrid, all of the keys and ciphertexts the adversary receives from the challenger are semi-functional. At this point, we can rely on a simple statistical argument to argue that the adversary's distinguishing advantage is negligible. We now describe the structure of the semi-functional ciphertexts and keys in the proof:

- **Semi-functional ciphertexts:** The semi-functional ciphertext ct for a message m and identity id^* contains an obfuscation of a modified program P^*. The program P^* additionally contains a secret key $\mathsf{sk}_{\mathsf{id}^*}$ and a bit string $t = \mathsf{PRG}(h(\vec{r}_{\mathsf{id}^*}))$, where PRG is a length-doubling pseudorandom generator (PRG) and h is a universal hash function. The program P^* takes $(\mathsf{ct}_{\mathsf{IBE}}, \pi)$ as input, but outputs m only if π is a valid proof (on the statement $(\mathsf{id}^*, \vec{c}_{\mathsf{id}^*}, \mathsf{ct}_{\mathsf{IBE}})$) *and* $\mathsf{PRG}(h(\mathsf{IBE.Decrypt}(\mathsf{sk}_{\mathsf{id}^*}, \mathsf{id}^*, \mathsf{ct}_{\mathsf{IBE}}))) = t$.

- **Semi-functional keys:** The semi-functional key $\mathsf{sk}_{\mathsf{id}} = (\mathsf{ct}_{\mathsf{IBE}}, \pi)$ has a simulated proof π and moreover, the ciphertext $\mathsf{ct}_{\mathsf{IBE}}$ is an encryption of the all-zeroes string $\mathsf{ct}_{\mathsf{IBE}} = \mathsf{IBE.Encrypt}(\mathsf{pp}_{\mathsf{IBE}}, \mathsf{id}, 0; \rho_{\mathsf{enc}})$.

To show security, we first switch the challenge ciphertexts to be semi-functional. Then we switch each of the keys to be semi-functional. Once all of the challenge ciphertexts and keys are semi-functional, semantic security follows by a simple statistical argument together with security of the obfuscation scheme. We give a sketch below:

- **Switching ciphertexts to be semi-functional:** To switch ciphertexts into semi-functional mode, we appeal to $i\mathcal{O}$ security. Specifically, it suffices to show that the original program P and the modified program P^* are functionally equivalent. This follows immediately by (statistical) soundness of the NIZK and correctness of the IBE scheme. Specifically, statistical soundness of the NIZK means that if the proof π verifies, then
$$\mathsf{ct}_{\mathsf{IBE}} = \mathsf{IBE.Encrypt}(\mathsf{pp}_{\mathsf{IBE}}, \mathsf{id}^*, \vec{r}_{\mathsf{id}^*}).$$
Correctness of IBE now implies that
$$\mathsf{IBE.Decrypt}(\mathsf{sk}_{\mathsf{id}^*}, \mathsf{id}^*, \mathsf{ct}_{\mathsf{IBE}}) = \vec{r}_{\mathsf{id}^*}.$$
In this case, the additional PRG check that P^* performs always succeeds.

- **Switching keys to semi-functional.** We now switch the keys $\mathsf{sk}_{\mathsf{id}} = (\mathsf{ct}_{\mathsf{IBE}}, \pi)$ to semi-functional. To do so, we first appeal to simulation security of the NIZK (to switch from real proofs to simulated proofs). We then leverage semantic security of the IBE scheme to switch $\mathsf{ct}_{\mathsf{IBE}}$ from an encryption of $\vec{r}_{\mathsf{id}}$ to an encryption of the all-zeroes string. Note that at this point in the proof, the challenge ciphertexts are semi-functional, and thus, simulating a challenge ciphertext for an identity id^* requires knowledge of $\mathsf{sk}_{\mathsf{id}^*}$. However, the reduction algorithm can obtain these keys from the IBE challenger. Note that this is admissible because the adversary in the big-key IBE game is not allowed to query for a key for an identity $\mathsf{id}^* \in \mathcal{J}$ that appears in the challenge set $\mathcal{J}$.

- **Completing the proof:** To finish the proof, we switch the commitments to hiding mode. This essentially "erases" the bits $r_1, \ldots, r_N$ from the public parameters. At this point in the proof, the only information on the bits r_i is contained in the leakage function on msk via the ℓ bits of leakage. When the challenge set $\mathcal{J}$ is sufficiently large, there must exist some identity $\mathsf{id}^* \in \mathcal{J}$ such that $\vec{r}_{\mathsf{id}^*}$ has high min-entropy given the leakage. For this to work, we require that the mapping $\mathcal{H}$ from identities to the indices of the master secret key has good "spread." That is, it should be the case that the set $\{\vec{r}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{J}}$ contains many distinct indices of $\vec{r}$. Then, there exists some $\mathsf{id}^* \in \mathcal{J}$ such that $\vec{r}_{\mathsf{id}^*}$ has high min-entropy. At this point, we can appeal to the leftover hash lemma [HILL99] to argue that $t = \mathsf{PRG}(h(\vec{r}_{\mathsf{id}^*}))$ is statistically close to $t = \mathsf{PRG}(u)$, where u is a random seed. By PRG security, the string t is computationally indistinguishable from a uniform string. Since the PRG is length-doubling, with overwhelming probability over the choice of t, it is no longer in the image of the PRG. At this point, the additional check that P^* performs (i.e., that $\mathsf{PRG}(h(\mathsf{IBE.Decrypt}(\mathsf{sk}_{\mathsf{id}^*}, \mathsf{id}^*, \mathsf{ct}_{\mathsf{IBE}}))) = t$)

is unsatisfiable. Correspondingly, the program P^* outputs $\perp$ on all inputs, so by $i\mathcal{O}$ security, it is computationally indistinguishable from the program that always outputs $\perp$. Since this program is independent of the message m, semantic security holds trivially.

We provide the full construction and analysis in Section 4. Thus, we obtain a simple construction of an adaptively-secure big-key IBE scheme from indistinguishability obfuscation and one-way functions; specifically, all of the underlying building blocks can be built from $i\mathcal{O}$ and one-way functions [SW14, ABSV15].

Using witness encryption in place of obfuscation. If we inspect our above template for constructing big-key IBE, we observe that the ciphertext is essentially an obfuscated program that takes as input a proof and checks whether the proof is valid or not. Thus, similar to the approach in [DGSW22], it seems plausible that we could also replace the obfuscated program with a witness encryption scheme [GGSW13]. In this work, we show that this is indeed possible, but will require a more involved construction. Specifically, in witness encryption, a user can encrypt a message m to an NP statement x; the decryption algorithm takes an NP witness w for x and outputs the statement. The security requirement then says that if x is not in the language, then the ciphertext computationally hides the associated message. Witness encryption provides no guarantees if the statement x is in the language, even if the witness is computationally hard to find. In our basic template above, the ciphertext always encodes a *true* instance (since decryption is possible), and we rely on $i\mathcal{O}$ security to (gradually) replace it with an instance that is unsatisfiable (in the final hybrid experiment). Such a proof strategy does not work in the setting of witness encryption since it provides no hiding properties for the underlying NP relation. Thus, substituting witness encryption in place of indistinguishability obfuscation will require some additional tools.

Specifically, in the $i\mathcal{O}$ construction, the semi-functional ciphertexts introduces an *additional* check that the provided secret key $\mathsf{sk}_{\mathsf{id}^*} = (\mathsf{ct}_{\mathsf{IBE}}, \pi)$ satisfies $\mathsf{PRG}(h(\mathsf{IBE.Decrypt}(\mathsf{sk}_{\mathsf{id}^*}, \mathsf{id}^*, \mathsf{ct}_{\mathsf{IBE}}))) = t$. Since this check always passes, security of $i\mathcal{O}$ ensures that the resulting program remains functionally equivalent to a normal program. In the case of witness encryption, we do not have the flexibility to change the NP relation associated with a challenge ciphertext, so we will have to augment the NP relation in the *normal* ciphertexts to also perform this additional check. We now give an outline of our approach, focusing on the places that differ from our $i\mathcal{O}$ construction:

- The master secret key still consists of a random string $r_1, \ldots, r_N \xleftarrow{\mathrm{R}} \{0,1\}$. As before, the public parameters include commitments $c_1, \ldots, c_N$ to $r_1, \ldots, r_N$, and the master secret key contains the corresponding openings.
- The secret key for an identity id will contain a testing key $\mathsf{sk}_{\mathsf{out}}$ for $\vec{r}_{\mathsf{id}}$ (where $\vec{r}_{\mathsf{id}} = (r_i)_{i \in I_{\mathsf{id}}}$ and $I_{\mathsf{id}} = \mathcal{H}(\mathsf{id})$ as before). The testing key $\mathsf{sk}_{\mathsf{out}}$ plays the role of the IBE ciphertext in the $i\mathcal{O}$ construction. In addition, the secret key also contains a NIZK proof (like in the $i\mathcal{O}$ construction) which affirms that $\mathsf{sk}_{\mathsf{out}}$ is an encoding of $\vec{r}_{\mathsf{id}}$ and that $\vec{r}_{\mathsf{id}}$ are the bits associated with the commitment $\vec{c}_{\mathsf{id}}$.
- To encrypt a message m to an identity id^*, the user first samples a random encoding $\mathsf{ct}_{\mathsf{out}}$, and prepares a witness encryption ciphertext for the statement $(\mathsf{id}^*, \vec{c}_{\mathsf{id}^*}, \mathsf{ct}_{\mathsf{out}})$; for simplicity of exposition, we omit the common reference strings for the NIZK and the bit commitments in this sketch. The witness for the witness encryption scheme is a secret key $\mathsf{sk}_{\mathsf{id}} = (\mathsf{sk}_{\mathsf{out}}, \pi)$, and the associated NP relation first checks the proof π is valid, and moreover, that the encoding $\mathsf{ct}_{\mathsf{out}}$ is valid with respect to the testing key $\mathsf{sk}_{\mathsf{out}}$.

The additional validity check between $\mathsf{sk}_{\mathsf{out}}$ and $\mathsf{ct}_{\mathsf{out}}$ is the analog of the additional check that the semi-functional ciphertexts performs in the earlier $i\mathcal{O}$ construction. Specifically, we require the encodings satisfy the following properties:

- There is a public algorithm that allows one to sample a fresh encoding. This is used during encryption to sample $\mathsf{ct}_{\mathsf{out}}$. The first requirement is that the testing key $\mathsf{sk}_{\mathsf{out}}$ always accepts a normal encoding (this ensures correctness).
- Next, we define the notion of a semi-functional encoding. Using a trapdoor, it is possible to sample a semi-functional encoding $\mathsf{ct}_{\mathsf{out}}$ of a vector $\vec{r}_{\mathsf{id}}$. Here, the requirement is that a (normal) testing key $\mathsf{sk}_{\mathsf{out}}$ for $\vec{r}_{\mathsf{id}}$ will always *reject* a semi-functional encoding of $\vec{r}_{\mathsf{id}}$.

We refer to these encodings as a privately-testable encoding since given a trapdoor, it is possible to generate a (semi-functional) key to test whether an encoding is of a particular target value or not. In the security proof, we will switch the encodings in the challenge ciphertexts (for an identity id) from normal encodings (which can be decrypted normally) into semi-functional encodings of $\vec{r}_{\mathsf{id}}$. Consider now a candidate witness $(\mathsf{sk}_{\mathsf{out}}, \pi)$ for a challenge ciphertext:

- If the NIZK proof π verifies, then by statistical binding of the commitment scheme and statistical soundness of the NIZK proof system, it must be the case that $\mathsf{sk}_{\mathsf{out}}$ is an encoding of $\vec{r}_{\mathsf{id}}$.
- However, if the encoding $\mathsf{ct}_{\mathsf{out}}$ in the challenge ciphertext is a semi-functional encoding of $\vec{r}_{\mathsf{id}}$, then $\mathsf{sk}_{\mathsf{out}}$ will always reject $\mathsf{ct}_{\mathsf{out}}$.

Thus, for all candidate witnesses for the statement associated with a challenge ciphertext, either the NIZK proof fails to verify or the encoding check fails. In both cases, the relation is not satisfied, and so the statement is false. We can now appeal to semantic security of the witness encryption scheme.

Simulating NIZK proofs. The above proof strategy critically requires on statistical soundness of the NIZK (to ensure that if the adversary produces a valid proof π, then the associated testing key is bound to the vector $\vec{r}_{\mathsf{id}}$). However, in the reduction, we still require a way to simulate proofs *without* knowledge of $r_1, \ldots, r_N$ (to ensure that the only leakage on the bits $r_1, \ldots, r_N$ is from the leakage function). Essentially, the reduction needs a way to simulate secret keys without knowledge of the randomness $r_1, \ldots, r_N$ and still retain statistical soundness. We achieve this using an or-proof construction. Specifically, we introduce an additional branch into the NIZK proof system so the proof either asserts validity of the testing key $\mathsf{sk}_{\mathsf{out}}$ (with respect to the commitments $\vec{c}_{\mathsf{id}}$) as above, or alternatively, the prover knows a trapdoor embedded within the CRS. In the real scheme, only the first branch will be used while in the security proof, the reduction algorithm will simulate proofs using the simulation trapdoor.

At this point, we still need to ensure that the simulated proofs do not help the adversary break semantic security. In particular, by switching to the or-proof, we can no longer argue that a valid proof π means that the testing key $\vec{r}_{\mathsf{id}}$ is correctly constructed. To get around this problem, we introduce the concept of a split encoding. At a very high level, we include an auxiliary encoding $\mathsf{sk}_{\mathsf{aux}}$ and $\mathsf{ct}_{\mathsf{aux}}$ with each secret key and ciphertext, respectively. The ciphertext component $\mathsf{ct}_{\mathsf{aux}}$ would be embedded as part of the statement in witness encryption while the secret key component $\mathsf{sk}_{\mathsf{aux}}$ would be part of the witness. The NP relation associated with the witness encryption scheme would then check that $\mathsf{ct}_{\mathsf{aux}}$ is valid with respect to $\mathsf{sk}_{\mathsf{aux}}$. To preserve correctness, we require that for normally-generated encodings, the check always passes. However, the check rejects when both $\mathsf{ct}_{\mathsf{aux}}$ and $\mathsf{sk}_{\mathsf{aux}}$ are switched to *semi-functional* encodings. We now modify the trapdoor branch of the or-language in the NIZK proof system to also check that the key encoding $\mathsf{sk}_{\mathsf{aux}}$ is a semi-functional encoding. In the proof, the semi-functional ciphertexts have semi-functional encodings $\mathsf{ct}_{\mathsf{aux}}$. This way, whenever the NIZK proof verifies, one of two properties must hold:

- The provided encoding $\mathsf{sk}_{\mathsf{out}}$ is a testing key for $\vec{r}_{\mathsf{id}}$, which would reject the encoding $\mathsf{ct}_{\mathsf{out}}$ in the challenge ciphertext. Thus, the witness encryption relation is not satisfied.

- The auxiliary encoding $\mathsf{sk}_{\mathsf{aux}}$ is a semi-functional encoding, which would reject the semi-functional $\mathsf{ct}_{\mathsf{aux}}$ in the challenge ciphertext. Once again, the witness encryption relation is not satisfied.

With these two encodings, we now have a way for the reduction algorithm to simulate key-generation queries (without knowledge of $r_1, \ldots, r_N$). Moreover, once all of the secret keys and challenge ciphertexts are semi-functional, the associated relation is false. Semantic security then follows from security of the witness encryption scheme. We provide the formal description of our privately-testable and split encodings as well as our construction of big-key IBE from witness encryption in Section 5.

Constructing privately-testable and split encodings. In Section 6, we show how to construct the encoding schemes we use from standard assumptions over groups. Specifically, privately-testable encodings follow from the standard decisional Diffie-Hellman (DDH) assumption in pairing-free groups while split encodings can be built from the symmetric external Diffie-Hellman (SXDH) assumption in asymmetric pairing groups. Recall that the SXDH assumption essentially amounts to assuming DDH holds in both base groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of an asymmetric pairing group.

Reducing public parameter size. As described so far, our big-key IBE scheme has long public parameters. Critically, the encryption algorithm only requires local access to the long public parameters. Previously, the work of [DGSW22] showed a generic approach based on non-interactive secure computation to compile any big-key IBE scheme with long public parameters (but where the underlying algorithm only require local access to the public parameters) into a scheme with short public parameters. This transformation also applies to our constructions. For completeness, we show that this transformation still preserves adaptive security in Appendix A.

Comparison with [WW24]. The recent work of Waters and Wichs [WW24] shows how to construct adaptively-secure attribute-based encryption from witness encryption. As part of their proof strategy, they introduce the notion of a "functional tag system." A functional tag system consists of function tags and input tags, each of which has a semi-functional mode that is indistinguishable from the normal mode. Our notion of split encoding is conceptually similar to a functional tag system, but specialized to the case of an equality function (since we aim for IBE rather than ABE). However, for our application, we rely on a stronger notion of mode indistinguishability (that asserts computational indistinguishability of normal tags and semi-functional tags). For our applications, we require mode indistinguishability to hold with respect to multiple functions *and* multiple input tags. The Waters-Wichs notion considers many functions, but a single input tag. The need to simulate many input tags comes from the fact that in the big-key IBE security game, the reduction algorithm needs to simulate multiple challenge ciphertexts (specifically, this is needed to estimate the adversary's success probability and determine whether it is successful or not; we refer to Section 3 for further discussion of the advantage checking requirement). We do not see a way to generically amplify a functional tag system where security holds against adversaries that can request a single input tag into one that is secure against adversaries which can request multiple input tags.

2 Preliminaries

We write λ to denote the security parameter. For an integer $n \in \mathbb{N}$, we write $[n]$ to denote the set $\{1, 2, \ldots, a\}$. For integers $a < b$, we use $[a, b]$ to denote the set of integers $\{a, a+1, \ldots, b\}$. When $\vec{x} = (x_1, \ldots, x_N)$ is a vector of elements and $S \subseteq [N]$ is a set of indices, we will write $\vec{x}_S$ to denote the ordered sub-vector $(x_i)_{i \in S}$.

For a distribution $\mathcal{D}$ we write $x \leftarrow \mathcal{D}$ to denote that x is a random draw from $\mathcal{D}$. For a finite set S, we write $x \xleftarrow{\text{R}} S$ to denote a uniform random draw from S. When indexing a set S, we write $S[i]$ to denote the i^{th} element of S (in lexicographic order). For distributions $\mathcal{D}_0, \mathcal{D}_1$, we denote the statistical distance between them by $\Delta(\mathcal{D}_0, \mathcal{D}_1)$. We use boldface letters (e.g., $\mathbf{x}$) to denote vectors. We write $\text{poly}(\lambda)$ to denote a fixed function that is $O(\lambda^c)$ for some $c \in \mathbb{N}$ and $\text{negl}(\lambda)$ to denote a function that is $o(\lambda^{-c})$ for all $c \in \mathbb{N}$. We say an event occurs with overwhelming probability if its complement occurs with negligible probability. We say an algorithm is efficient if it runs in probabilistic polynomial time (in the length of its input).

Hoeffding's inequality. We will use Hoeffding's inequality to bound the sum of independent random variables [Hoe63]:

Fact 2.1 (Hoeffding's Inequality [Hoe63])**.** Let $X_1, \ldots, X_T$ be independent random variables where $0 \le X_i \le 1$ for all $i \in [T]$. Let $S = \sum_{i \in [T]} X_i$ and let $\mathbb{E}[S]$ denote the expected value of S. Then, for any $k \ge 0$,

$$\Pr[|S - \mathbb{E}[S]| \ge Tk] \le 2^{-\Omega(Tk^2)}.$$

Min-entropy. We recall some basic definitions on min-entropy. Our definitions are adapted from those in [DORS08]. For a (discrete) random variable X, we write $\mathbf{H}_\infty(X) = -\log(\max_x \Pr[X = x])$ to denote its min-entropy. For two (possibly correlated) discrete random variables X and Y, we define the average min-entropy of X given Y to be $\mathbf{H}_\infty(X \mid Y) = -\log(\mathbb{E}_{y \leftarrow Y} \max_x \Pr[X = x \mid Y = y])$. The optimal probability of an unbounded adversary guessing X given the correlated value Y is $2^{-\mathbf{H}_\infty(X \mid Y)}$. We now state some useful properties on the conditional min-entropy:

Lemma 2.2 (Conditional Min-Entropy [DORS08, Lemma 2.2])**.** *Let A, B, C be random variables and suppose there are at most 2^λ elements in the support of B. Then*

$$\mathbf{H}_\infty(A \mid (B, C)) \ge \mathbf{H}_\infty(A, B \mid C) - \lambda \ge \mathbf{H}_\infty(A \mid C) - \lambda.$$

Additionally, for any $\delta > 0$, with probability at least $1 - \delta$ over the choice of $b \leftarrow B$, we have

$$\mathbf{H}_\infty(A \mid B = b) \ge \mathbf{H}_\infty(A \mid B) - \log(1/\delta).$$

Lemma 2.3 (Min-Entropy Splitting Lemma [DFR+07, DGSW22])**.** *Let $X_1, \ldots, X_\ell$ be a sequence of random variables such that $\mathbf{H}_\infty(X_1, \ldots X_\ell) \ge \alpha$. Then there exists a random variable C over $[\ell]$ such that $\mathbf{H}_\infty(X_C \mid C) \ge \alpha/\ell - \log \ell$.*

Dispersers. Our construction will rely on a disperser (also known as a "one-sided extractor"). At a high level, a disperser can be modeled as a bipartite graph with the property that that every subset of nodes of a certain (minimal) size on the left is guaranteed to have a large number of neighbors on the right. We recall the formal definition from [TUZ07]. First, a bipartite graph $G = (L, R, E)$ consists of two sets of vertices L and R together with a set of edges E, where each edge $e \in E$ is a pair of nodes $(u, v) \in L \times R$. For a set $S \subseteq L$, we write $N(S) \subseteq R$ to denote the neighborhood of S: $N(S) = \{v \in R : \exists (u, v) \in E \wedge u \in S\}$. We say G is D-left-regular if every node $u \in L$ has exactly D neighbors: $|N(\{u\})| = d$ for all $u \in L$.

Definition 2.4 (Disperser [TUZ07, Definition 1.3])**.** Let $G = (L, R, E)$ be a bipartite graph. Then, G is a degree-D (T, ε)-disperser if G is D-left-regular and for all subsets $S \subseteq L$ of size at least T, the neighborhood $N(S)$ has size at least $(1 - \varepsilon) \cdot |R|$. A disperser is explicit if the index of the i^{th} neighbor of a vertex $v \in L$ can be computed in $\text{poly}(\log N, \log D)$ time.

Fact 2.5 (Disperser [Par19, Theorem 3])**.** Let c be a universal constant, $\varepsilon > 0$ be any constant, and n be a set size parameter. Let $k = k(n)$, $D = D(n)$, $k_1 = k_1(n)$ be polynomials such that $c \log(n/\varepsilon) < k < n$ and $k_1 \geq k + O(\log^3(k/\varepsilon))$. Then, there exists an explicit degree-D $(2^{k_1}, \varepsilon)$-disperser $G = (L, R, E)$ where $D = \mathsf{poly}(n/\varepsilon)$, $|L| = 2^n$, and $|R| = 2^{k_1 + \Omega(\log(n/\varepsilon))}$.

Randomness extractors. We now recall the definition of randomness extractors (from the leftover hash lemma).

Definition 2.6 (Strong Randomness Extractor)**.** A function $\mathsf{Ext}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a (k, ε)-strong randomness extractor if for all distributions X over $\{0,1\}^n$ such that $\mathbf{H}_\infty(X) \geq k$, it holds that

$$\Delta\Big((s, \mathsf{Ext}(X, s)), (U_d, U_m)\Big) \leq \varepsilon,$$

where $s \xleftarrow{\text{R}} \{0,1\}^d$, and U_d, U_s are the uniform distributions on $\{0,1\}^d$, and $\{0,1\}^m$, respectively. An extractor is explicit if it is efficiently-computable.

Lemma 2.7 (Leftover Hash Lemma [ILL89, HILL99])**.** *Let X be a random variable with support U and suppose $\mathbf{H}_\infty(X) \geq k$. Take any $\varepsilon > 0$ and let $\mathcal{H}$ be a universal hash family of size 2^d and output length $m = k - 2\log(1/\varepsilon)$. Define $\mathsf{Ext}(x, h) \coloneqq h(x)$. Then Ext is a $(k, \varepsilon/2)$-strong extractor with seed length d and output length m.*

Corollary 2.8 (Explicit Strong Extractor)**.** *Take any $\lambda \in \mathbb{N}$. Then, there exists an explicit $(\lambda + \omega(\log \lambda), \mathsf{negl}(\lambda))$-strong randomness extractor $\mathsf{Ext}\colon \{0,1\}^{\mathsf{poly}(\lambda)} \times \{0,1\}^{\mathsf{poly}(\lambda)} \to \{0,1\}^\lambda$.*

Corollary 2.9 (Inner Product Extractor)**.** *Let $\mathbb{F}$ be a finite field and let X be a random variable with support $U = \mathbb{F}^n$. Take any $\varepsilon > 0$ and suppose $\mathbf{H}_\infty(X) \geq k = 2\log(1/\varepsilon) + \log|\mathbb{F}|$. Let $S = \mathbb{F}^n$ be a seed space and define $\mathsf{Ext}(\mathbf{x}, \mathbf{s}) \coloneqq \mathbf{s}^{\mathsf{T}}\mathbf{x}$. Then Ext is a $(k, \varepsilon/2)$-strong extractor with seed length $n \log|\mathbb{F}|$ and output length $\log|\mathbb{F}|$.*

2.1 Cryptographic Primitives

In this section, we recall the main cryptographic notions we use in this work.

Definition 2.10 (Pseudorandom Generator)**.** Let λ be a security parameter. A pseudorandom generator with output length $m = m(\lambda)$ is an efficiently-computable function family $\mathsf{PRG} = \{\mathsf{PRG}_\lambda\}_{\lambda \in \mathbb{N}}$ where $\mathsf{PRG}_\lambda\colon \{0,1\}^\lambda \to \{0,1\}^{m(\lambda)}$. We say that PRG is secure if for all efficient adversaries $\mathcal{A}$ there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$:

$$\left|\Pr[\mathcal{A}(1^\lambda, \mathsf{PRG}_\lambda(x)) = 1 : x \xleftarrow{\text{R}} \{0,1\}^\lambda] - \Pr[\mathcal{A}(1^\lambda, y) = 1 : y \xleftarrow{\text{R}} \{0,1\}^{m(\lambda)}]\right| = \mathsf{negl}(\lambda).$$

Definition 2.11 (Pseudorandom Function)**.** Let $\mathcal{K} = \{\mathcal{K}_\lambda\}_{\lambda \in \mathbb{N}}$, $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda \in \mathbb{N}}$, and $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda \in \mathbb{N}}$ be ensembles of finite sets indexed by a security parameter λ. Let $\mathsf{PRF} = \{\mathsf{PRF}_\lambda\}_{\lambda \in \mathbb{N}}$ be an efficiently-computable collection of functions $\mathsf{PRF}_\lambda\colon \mathcal{K}_\lambda \times \mathcal{X}_\lambda \to \mathcal{Y}_\lambda$. We say that PRF is secure if for all efficient adversaries $\mathcal{A}$ there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$:

$$\left|\Pr[\mathcal{A}^{\mathsf{PRF}_\lambda(k,\cdot)}(1^\lambda) = 1 : k \xleftarrow{\text{R}} \mathcal{K}_\lambda] - \Pr[\mathcal{A}^{f_\lambda(\cdot)}(1^\lambda) = 1 : f_\lambda \xleftarrow{\text{R}} \mathsf{Funs}[\mathcal{X}_\lambda, \mathcal{Y}_\lambda]]\right| = \mathsf{negl}(\lambda),$$

where $\mathsf{Funs}[\mathcal{X}_\lambda, \mathcal{Y}_\lambda]$ is the set of all functions from $\mathcal{X}_\lambda$ to $\mathcal{Y}_\lambda$.

Definition 2.12 (Identity-Based Encryption [Sha84, BF01, Coc01])**.** An identity-based encryption (IBE) scheme with identity space $\mathcal{ID} = \{\mathcal{ID}_\lambda\}_{\lambda \in \mathbb{N}}$ and message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda \in \mathbb{N}}$ is a tuple of efficient algorithms $\Pi_{\mathsf{IBE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ with the following syntax:

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{pp}, \mathsf{msk})$: On input the security parameter λ, the setup algorithm outputs the set of public parameters pp and a master secret key msk. We assume that pp and msk include the security parameter 1^λ.
- $\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}) \to \mathsf{sk}_{\mathsf{id}}$: On input the master secret key msk and an identity $\mathsf{id} \in \mathcal{ID}_\lambda$, the key-generation algorithm outputs an identity key $\mathsf{sk}_{\mathsf{id}}$. We assume the secret key $\mathsf{sk}_{\mathsf{id}}$ contains the security parameter 1^λ (from msk).
- $\mathsf{Encrypt}(\mathsf{pp}, \mathsf{id}, m) \to \mathsf{ct}$: On input the public parameters pp, an identity $\mathsf{id} \in \mathcal{ID}_\lambda$, and a message $m \in \mathcal{M}_\lambda$, the encryption algorithm outputs a ciphertext ct.
- $\mathsf{Decrypt}(\mathsf{sk}_{\mathsf{id}}, \mathsf{id}, \mathsf{ct}) \to m$: On input an identity secret key $\mathsf{sk}_{\mathsf{id}}$, an identity $\mathsf{id} \in \mathcal{ID}_\lambda$, and a ciphertext ct, the decryption algorithm outputs a message $m \in \mathcal{M}_\lambda$.

Moreover, Π_{IBE} should satisfy the following properties:

- **Correctness:** For all security parameters $\lambda \in \mathbb{N}$, all identities $\mathsf{id} \in \mathcal{ID}_\lambda$, all $(\mathsf{pp}, \mathsf{msk})$ in the support of $\mathsf{Setup}(1^\lambda)$, and all messages $m \in \mathcal{M}_\lambda$,

$$\Pr\left[\mathsf{Decrypt}(\mathsf{sk}_{\mathsf{id}}, \mathsf{ct}) = m : \begin{array}{c} \mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}) \\ \mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{pp}, \mathsf{id}, m) \end{array}\right] = 1.$$

- **Semantic security:** For a security parameter λ and a bit $b \in \{0,1\}$, we define the (adaptive) semantic security game between an adversary $\mathcal{A}$ and a challenger as follows:
  - **Setup:** The challenger starts by sampling $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and gives pp to $\mathcal{A}$.
  - **Pre-challenge queries:** Algorithm $\mathcal{A}$ can now issue key-generation queries to the challenger. On each key-generation query, adversary $\mathcal{A}$ specifies an identity $\mathsf{id} \in \mathcal{ID}_\lambda$, and the challenger replies with $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$.
  - **Challenge:** Algorithm $\mathcal{A}$ outputs a challenge identity id^* and two messages $m_0, m_1 \in \mathcal{M}_\lambda$. The challenger replies with $\mathsf{ct}_b \leftarrow \mathsf{Encrypt}(\mathsf{pp}, \mathsf{id}^*, m_b)$.
  - **Post-challenge queries:** Algorithm $\mathcal{A}$ can continue to make key-generation queries as in the pre-challenge phase.
  - **Output:** At the end of the game, algorithm $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$, which is the output of the experiment.

  An adversary $\mathcal{A}$ is admissible for the semantic security game if it does not issue a key-generation query on the challenge identity id^*. We say Π_{IBE} satisfies adaptive security if for all efficient and admissible adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$|\Pr[b' = 1 \mid b = 1] - \Pr[b' = 1 \mid b = 0]| = \mathsf{negl}(\lambda)$$

  in the semantic security game.

Definition 2.13 (Indistinguishability Obfuscation [BGI+12, GGH+13]). Let $C = \{C_\lambda\}_{\lambda \in \mathbb{N}}$ be a family of polynomial-size circuits and $\ell_C(\lambda)$ be a size parameter, such that every circuit $C \in C_\lambda$ has size exactly $\ell_C(\lambda)$. An indistinguishability obfuscator $i\mathcal{O}$ is an efficient algorithm that takes as input the security parameter 1^λ, a circuit $C \in C_\lambda$, and outputs a circuit C'. An $i\mathcal{O}$ scheme should satisfy the following properties:

- **Functionality-preserving:** For all security parameters $\lambda \in \mathbb{N}$, all $C \in \mathcal{C}_\lambda$, and all inputs x, we have that $C'(x) = C(x)$ where $C' \leftarrow i\mathcal{O}(1^\lambda, C)$.

- **Security:** For all efficient adversaries $\mathcal{A} = (\mathsf{Samp}, \mathcal{A}')$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that the following holds: if for all security parameters $\lambda \in \mathbb{N}$,

  $$\Pr[\forall x, C_0(x) = C_1(x) : (C_0, C_1, \mathsf{st}) \leftarrow \mathsf{Samp}(1^\lambda)] = 1 - \mathsf{negl}(\lambda)$$

  then
  $$\left|\Pr[\mathcal{A}'(\mathsf{st}, i\mathcal{O}(1^\lambda, C_0)) = 1] - \Pr[\mathcal{A}'(\mathsf{st}, i\mathcal{O}(1^\lambda, C_1)) = 1]\right| = \mathsf{negl}(\lambda),$$

  where $(C_0, C_1, \mathsf{st}) \leftarrow \mathsf{Samp}(1^\lambda)$.

Definition 2.14 (Witness Encryption [GGSW13, adapted])**.** Let $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda \in \mathbb{N}}$ be a message space. A witness encryption scheme for an NP language $\mathcal{L}$ with witness relation $\mathcal{R}_\mathcal{L}$ is a tuple of efficient algorithms $\Pi_{\mathsf{WE}} = (\mathsf{Encrypt}, \mathsf{Decrypt})$ with the following syntax:

- $\mathsf{Encrypt}(1^\lambda, m, x) \to \mathsf{ct}$: On input the security parameter λ, a message $m \in \mathcal{M}_\lambda$, and an instance x for the language $\mathcal{L}$, the encryption algorithm outputs a ciphertext ct. We assume ct includes 1^λ and x.

- $\mathsf{Decrypt}(\mathsf{ct}, w) \to m$: On input a ciphertext ct and a witness w, the decryption algorithm outputs a message $m \in \mathcal{M}_\lambda$.

Moreover, Π_{WE} should satisfy the following properties:

- **Correctness:** For all $\lambda \in \mathbb{N}$, messages $m \in \mathcal{M}_\lambda$, and tuples $(x, w) \in \mathcal{R}_\mathcal{L}$, it holds that

  $$\Pr[\mathsf{Decrypt}(\mathsf{ct}, w) = m : \mathsf{ct} \leftarrow \mathsf{Encrypt}(1^\lambda, m, x)] = 1.$$

- **Semantic security:** For a security parameter λ and a bit $b \in \{0, 1\}$, we define the semantic security game between an adversary $\mathcal{A}$ and a challenger as follows:

  - On input the security parameter 1^λ, algorithm $\mathcal{A}$ outputs a statement x and two messages $m_0, m_1 \in \mathcal{M}_\lambda$.
  - The challenger replies with $\mathsf{ct} \leftarrow \mathsf{Encrypt}(1^\lambda, m_b, x)$.
  - Algorithm $\mathcal{A}$ outputs a bit $b' \in \{0, 1\}$, which is the output of the experiment.

  The scheme Π_{WE} satisfies semantic security if for all efficient adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

  $$|\Pr[b' = 1 \wedge x \notin \mathcal{L} \mid b = 1] - \Pr[b' = 1 \wedge x \notin \mathcal{L} \mid b = 0]| = \mathsf{negl}(\lambda)$$

  in the semantic security game.

One-time dual-mode commitment. We recall the notion of a one-time dual-mode commitment, which can be constructed from one-way functions [Nao89].

Definition 2.15 (One-Time Dual-Mode Commitment [DN02])**.** A one-time dual-mode commitment scheme with input space $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda \in \mathbb{N}}$ is a tuple of efficient algorithms $\Pi_{\mathsf{Com}} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Verify})$ with the following syntax:

- $\mathsf{Setup}(1^\lambda, \mathsf{mode}) \to (\mathsf{crs}, \mathsf{td}, c)$: On input the security parameter λ and $\mathsf{mode} \in \{\mathsf{bind}, \mathsf{hide}\}$, the setup algorithm outputs a common reference string crs. When $\mathsf{mode} = \mathsf{hide}$, it also outputs a trapdoor td and commitment c. We assume crs and td (implicitly) contain the security parameter 1^λ.
- $\mathsf{Commit}(\mathsf{crs}, x) \to (c, \sigma)$: On input the common reference string crs and an input $x \in \mathcal{X}_\lambda$, the commit algorithm outputs a commitment c and an opening σ.
- $\mathsf{Verify}(\mathsf{crs}, c, x, \sigma) \to \{0, 1\}$: On input the common reference string crs, a commitment c, a value $x \in \mathcal{X}_\lambda$, and an opening σ, the verification algorithm outputs a bit $b \in \{0, 1\}$.

Moreover, Π_{Com} should satisfy the following properties:

- **Correctness:** For all security parameters $\lambda \in \mathbb{N}$, all inputs $x \in \mathcal{X}_\lambda$, all modes $\mathsf{mode} \in \{\mathsf{bind}, \mathsf{hide}\}$,
$$\Pr\left[\mathsf{Verify}(\mathsf{crs}, c, x, \sigma) = 1 : \begin{array}{c} (\mathsf{crs}, \mathsf{td}, c') \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{mode}); \\ (c, \sigma) \leftarrow \mathsf{Commit}(\mathsf{crs}, x) \end{array}\right] = 1.$$
- **Statistically binding in binding mode:** For all adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,
$$\Pr\left[\mathsf{Verify}(\mathsf{crs}, c, x_0, \sigma_0) = \mathsf{Verify}(\mathsf{crs}, c, x_1, \sigma_1) = 1 \wedge x_0 \neq x_1 : \begin{array}{c} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{bind}); \\ (c, x_0, x_1, \sigma_0, \sigma_1) \leftarrow \mathcal{A}(\mathsf{crs}) \end{array}\right] = \mathsf{negl}(\lambda).$$
- **Mode indistinguishability:** For a security parameter λ, a bit $b \in \{0, 1\}$, and a simulator $\mathcal{S}_{\mathsf{open}}$, we define the mode indistinguishability game between an adversary $\mathcal{A}$ and a challenger as follows:
  1. The challenger samples $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{bind})$ if $b = 0$ or $(\mathsf{crs}, \mathsf{td}, c_1) \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{hide})$ if $b = 1$ and gives crs to $\mathcal{A}$.
  2. Algorithm $\mathcal{A}$ outputs a value $x \in \mathcal{X}_\lambda$.
  3. If $b = 0$ the challenger computes $(c_0, \sigma_0) \leftarrow \mathsf{Commit}(\mathsf{crs}, x)$. If $b = 1$, the challenger computes a simulated opening $\sigma_1 \leftarrow \mathcal{S}_{\mathsf{open}}(\mathsf{td}, x)$. The challenger sends (c_b, σ_b) to $\mathcal{A}$.
  4. Algorithm $\mathcal{A}$ outputs a bit $b' \in \{0, 1\}$, which is the output of the experiment.

  The scheme Π_{Com} satisfies mode indistinguishability if there exists an efficient simulator $\mathcal{S}_{\mathsf{open}}$ such that for all efficient adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,
$$|\Pr[b' = 1 \mid \beta = 0] - \Pr[b' = 1 \mid \beta = 1]| = \mathsf{negl}(\lambda)$$
  in the mode indistinguishability game.

Non-interactive zero-knowledge. Next, we recall the notion of a non-interactive zero-knowledge (NIZK) proof for NP [GMR85, BFM88]. Specifically, we consider NIZKs for the language of Boolean circuit satisfiability which we define below. We also recall the weaker notion of *witness indistinguishability*, which is more convenient to use in some of our proofs. It is easy to see that zero-knowledge implies witness indistinguishability.

Definition 2.16 (Boolean Circuit Satisfiability)**.** The language $\mathcal{L}$ of Boolean circuit satisfiability consists of pairs (C, x) of circuits $C \colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$ and inputs $x \in \{0,1\}^n$ such that there exists $w \in \{0,1\}^h$ where $C(x, w) = 1$:

$$\mathcal{L} = \left\{ (C, x) : \begin{array}{c} C \colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\} \\ x \in \{0,1\}^n \\ \exists w \in \{0,1\}^h : C(x,w) = 1 \end{array} \right\}.$$

Definition 2.17 (NIZK for NP [GMR85, BFM88])**.** A non-interactive zero-knowledge (NIZK) proof for Boolean circuit satisfiability is a tuple of efficient algorithms $\Pi_{\mathsf{NIZK}} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ with the following syntax:

- $\mathsf{Setup}(1^\lambda) \to \mathsf{crs}$: On input the security parameter $\lambda \in \mathbb{N}$, the setup algorithm outputs a common reference string crs. We assume crs implicitly contains a description of the security parameter 1^λ.
- $\mathsf{Prove}(\mathsf{crs}, C, x, w) \to \pi$: On input the common reference string crs, a Boolean circuit $C \colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$, a statement $x \in \{0,1\}^n$, and a witness $w \in \{0,1\}^h$, the prove algorithm outputs a proof π.
- $\mathsf{Verify}(\mathsf{crs}, C, x, \pi) \to b$: On input the common reference string crs, the Boolean circuit $C \colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$, a statement $x \in \{0,1\}^n$, and a proof π, the verification algorithm outputs a bit $b \in \{0,1\}$.

Moreover, Π_{NIZK} should satisfy the following properties:

- **Completeness:** For all $\lambda \in \mathbb{N}$, all Boolean circuits $C \colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$, all statements $x \in \{0,1\}^n$, and all witnesses $w \in \{0,1\}^h$ where $C(x, w) = 1$,

$$\Pr\left[\mathsf{Verify}(\mathsf{crs}, C, x, \pi) = 1 : \begin{array}{c} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda) \\ \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, C, x, w) \end{array}\right] = 1.$$

- **Statistical soundness:** For all adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$\Pr\left[(C, x) \notin \mathcal{L} \wedge \mathsf{Verify}(\mathsf{crs}, C, x, \pi) = 1 : \begin{array}{c} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda) \\ (C, x, \pi) \leftarrow \mathcal{A}(\mathsf{crs}) \end{array}\right] = \mathsf{negl}(\lambda).$$

- **Computational zero-knowledge:** For every efficient adversary $\mathcal{A}$, there exists an efficient simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ and a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$\left| \Pr\left[\mathcal{A}^{O_0(\mathsf{crs}, \cdot, \cdot, \cdot)}(1^\lambda, \mathsf{crs}) = 1\right] - \Pr\left[\mathcal{A}^{O_1(\mathsf{st}_{\mathcal{S}}, \cdot, \cdot, \cdot)}(1^\lambda, \widetilde{\mathsf{crs}}) = 1\right] \right| = \mathsf{negl}(\lambda),$$

where $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$, $(\widetilde{\mathsf{crs}}, \mathsf{st}_{\mathcal{S}}) \leftarrow \mathcal{S}_1(1^\lambda)$, and the oracles O_0 and O_1 are defined as follows:

- $\mathcal{O}_0(\mathsf{crs}, C, x, w)$: On input crs, a circuit $C\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$, a statement $x \in \{0,1\}^n$, and a witness $w \in \{0,1\}^h$, the oracle outputs $\perp$ if $C(x,w) = 0$. If $C(x,w) = 1$, it outputs $\mathsf{Prove}(\mathsf{crs}, C, x, w)$.
- $\mathcal{O}_1(\mathsf{st}_{\mathcal{S}}, C, x, w)$: On input the simulator state $\mathsf{st}_{\mathcal{S}}$, a circuit $C\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$, a statement $x \in \{0,1\}^n$, and a witness $w \in \{0,1\}^h$, the oracle outputs $\perp$ if $C(x,w) = 0$. If $C(x,w) = 1$, it outputs $\mathcal{S}_2(\mathsf{st}_{\mathcal{S}}, C, x)$.

Definition 2.18 (Computational Witness Indistinguishability)**.** Let $\Pi_{\mathsf{NIZK}} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ be a NIZK proof for Boolean circuit satisfiability. We say that Π_{NIZK} satisfies computational witness indistinguishability if for every efficient adversary $\mathcal{A}$, there exists a negligible function negl such that for all $\lambda \in \mathbb{N}$,

$$\left|\Pr\left[\mathcal{A}^{\mathcal{O}_0(\mathsf{crs},\cdot,\cdot,\cdot,\cdot)}(1^\lambda, \mathsf{crs}) = 1\right] - \Pr\left[\mathcal{A}^{\mathcal{O}_1(\mathsf{crs},\cdot,\cdot,\cdot,\cdot)}(1^\lambda, \mathsf{crs}) = 1\right]\right| = \mathsf{negl}(\lambda),$$

where $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$ and for $b \in \{0,1\}$, the oracle $\mathcal{O}_b$ is defined as follows:

- $\mathcal{O}_b(\mathsf{crs}, C, x, w_0, w_1)$: On input crs, a circuit $C\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$, a statement $x \in \{0,1\}^n$, and witnesses $w_0, w_1 \in \{0,1\}^h$, the oracle outputs $\mathsf{Prove}(\mathsf{crs}, C, x, w_b)$ if $C(x, w_0) = 1 = C(x, w_1)$. Otherwise, it outputs $\perp$.

3 Big-Key Identity-Based Encryption

In this section, we give a formal definition of big-key IBE. Our definition is based on the corresponding definition from [DGSW22], but has an important difference where we only consider inverse-polynomial advantage thresholds rather than all non-negligible advantage thresholds. This is an important distinction as the previous notion from [DGSW22] is unsatisfiable (see Remark 3.2). We begin by highlighting the main difference between big-key IBE and vanilla IBE (Definition 2.12):

- In big-key IBE, we allow the adversary to specify any efficiently-computable leakage function f that outputs at most ℓ bits, where ℓ is a leakage parameter. The challenger then replies with $f(\mathsf{msk})$. In the adaptive security experiment, the adversary chooses the challenge identities after it observes the (arbitrary) leakage on the master secret key.
- Since the adversary is given *arbitrary* leakage on the master secret key, its leakage may simply encode a secret key for the challenge identity. Thus, the usual notion of semantic security is not meaningful in this model. Instead, the adversary must declare a set of challenge identities $\mathcal{J}$. To win the game, the adversary must be able to break semantic security for all identities $\mathsf{id} \in \mathcal{J}$ with advantage greater than some threshold ε.

We now provide the formal definition and then discuss how it compares with the previous definition from [DGSW22].

Definition 3.1 (Big-Key Identity-Based Encryption [DGSW22, adapted])**.** A big-key identity-based encryption scheme with identity space $\mathcal{ID} = \{\mathcal{ID}_\lambda\}_{\lambda \in \mathbb{N}}$ and message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda \in \mathbb{N}}$ is a tuple of efficient algorithms $\Pi_{\mathsf{bkIBE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ with the following syntax:

- $\mathsf{Setup}(1^\lambda, 1^\ell) \to (\mathsf{pp}, \mathsf{msk})$: On input the security parameter λ and the leakage parameter ℓ, the setup algorithm outputs public parameters pp and a master secret key msk. We assume that pp and msk include the security parameter 1^λ.

- $\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}) \to \mathsf{sk}_{\mathsf{id}}$: On input the master secret key msk and an identity $\mathsf{id} \in \mathcal{ID}_\lambda$, the key-generation algorithm outputs an identity secret key $\mathsf{sk}_{\mathsf{id}}$. We assume the secret key $\mathsf{sk}_{\mathsf{id}}$ contains the security parameter 1^λ (from msk).
- $\mathsf{Encrypt}(\mathsf{pp}, \mathsf{id}, m) \to \mathsf{ct}$: On input the public parameters pp, an identity $\mathsf{id} \in \mathcal{ID}_\lambda$, and a message $m \in \mathcal{M}_\lambda$, the encryption algorithm outputs a ciphertext ct.
- $\mathsf{Decrypt}(\mathsf{sk}_{\mathsf{id}}, \mathsf{id}, \mathsf{ct}) \to m$: On input an identity secret key $\mathsf{sk}_{\mathsf{id}}$, an identity $\mathsf{id} \in \mathcal{ID}_\lambda$, and a ciphertext ct, the decryption algorithm outputs a message $m \in \mathcal{M}_\lambda$.

Moreover, Π_{bkIBE} should satisfy the following properties:

- **Correctness:** For all security parameters $\lambda \in \mathbb{N}$, all leakage parameters $\ell \in \mathbb{N}$, all identities $\mathsf{id} \in \mathcal{ID}_\lambda$, all $(\mathsf{pp}, \mathsf{msk})$ in the support of $\mathsf{Setup}(1^\lambda, 1^\ell)$, and all messages $m \in \mathcal{M}_\lambda$,
$$\Pr\left[\mathsf{Decrypt}(\mathsf{sk}_{\mathsf{id}}, \mathsf{id}, \mathsf{ct}) = m : \begin{array}{c} \mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}) \\ \mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{pp}, \mathsf{id}, m) \end{array}\right] = 1.$$
- **Efficiency:** We impose the following efficiency requirements on the scheme parameters:
  - **Public key size:** We say that a big-key IBE scheme has short public parameters if the public parameters pp output by $\mathsf{Setup}(1^\lambda, 1^\ell)$ satisfies $|\mathsf{pp}| = \mathsf{poly}(\lambda, \log \ell)$. We say the scheme has long public parameters if $|\mathsf{pp}| = \mathsf{poly}(\lambda, \ell)$.
  - **Secret key size:** We require that the identity secret keys $\mathsf{sk}_{\mathsf{id}}$ output by KeyGen to satisfy $|\mathsf{sk}_{\mathsf{id}}| = \mathsf{poly}(\lambda, \log \ell)$.
  - **Key-generation and encryption time:** We require that KeyGen and Encrypt run in time $\mathsf{poly}(\lambda, \log \ell)$ given *random* access to the master secret key msk and the public parameters pp, respectively. In other words, KeyGen only needs to read $\mathsf{poly}(\lambda, \log \ell)$ bits of msk and Encrypt only needs to read $\mathsf{poly}(\lambda, \log \ell)$ bits of pp. Note that if the scheme has short public parameters (i.e., if $|\mathsf{pp}| = \mathsf{poly}(\lambda, \log \ell)$), then the encryption requirement is trivially satisfied.
- **Adaptive security under bounded leakage:** For a security parameter λ, a challenge parameter $k = k(\lambda, \ell)$, and an advantage function $\varepsilon = \varepsilon(\lambda)$, we define the adaptive security game between an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and a challenger as follows:
  - **Setup:** On input the security parameter, algorithm $\mathcal{A}_1$ starts by outputting a leakage parameter 1^ℓ, which it gives to the challenger. The challenger samples $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$ and gives pp to $\mathcal{A}_1$.
  - **Pre-leakage queries:** Algorithm $\mathcal{A}_1$ can now issue key-generation queries to the challenger. On each key-generation query, algorithm $\mathcal{A}_1$ specifies an identity $\mathsf{id} \in \mathcal{ID}_\lambda$ and the challenger replies with $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$.
  - **Leakage:** Algorithm $\mathcal{A}_1$ outputs the description of an efficiently-computable function f with output length at most ℓ. The challenger replies with $\mathsf{leak} := f(\mathsf{msk})$.
  - **Post-leakage queries:** Algorithm $\mathcal{A}_1$ can continue to make key-generation queries to the challenger.
  - **Challenge:** Algorithm $\mathcal{A}_1$ outputs a set $\mathcal{J} \subseteq \mathcal{ID}_\lambda$ of size at least $k = k(\lambda, \ell)$, two messages $m_0, m_1 \in \mathcal{M}_\lambda$, and a state st.

- **Output:** The output of the adaptive security game is $b' = 1$ if

$$\forall \mathsf{id} \in \mathcal{J} : \mathsf{Adv}^{\mathsf{id}}(\mathsf{msk}, \mathsf{pp}, \mathsf{st}, \mathsf{leak}) \geq \varepsilon(\lambda) \tag{3.1}$$

and $b' = 0$ otherwise. The distinguishing advantage $\mathsf{Adv}^{\mathsf{id}}(\mathsf{msk}, \mathsf{pp}, \mathsf{st}, \mathsf{leak})$ is defined as follows:

For an identity $\mathsf{id} \in \mathcal{ID}_\lambda$, define the experiment $\mathsf{Exp}^{\mathsf{id}}(\mathsf{msk}, \mathsf{pp}, \mathsf{st}, \mathsf{leak})$ as follows:

* The challenger samples $b \xleftarrow{\mathrm{R}} \{0, 1\}$, $\mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{pp}, \mathsf{id}, m_b)$ and gives $(\mathsf{st}, \mathsf{id}, \mathsf{ct})$ to $\mathcal{A}_2$.
* Algorithm $\mathcal{A}_2$ can now issue key-generation queries to the challenger. On each key-generation query, algorithm $\mathcal{A}_2$ specifies an identity $\mathsf{id} \in \mathcal{ID}_\lambda$ and the challenger replies with $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$.
* After $\mathcal{A}_2$ has finished making key-generation queries, it outputs a bit $\beta \in \{0, 1\}$, which is used to compute the output of the experiment as 1 if $b = \beta$ and 0 otherwise.

The advantage $\mathsf{Adv}^{\mathsf{id}}(\mathsf{msk}, \mathsf{pp}, \mathsf{st}, \mathsf{leak})$ is then defined as

$$\mathsf{Adv}^{\mathsf{id}}(\mathsf{msk}, \mathsf{pp}, \mathsf{st}, \mathsf{leak}) = \left|\Pr[\mathsf{Exp}^{\mathsf{id}}(\mathsf{msk}, \mathsf{pp}, \mathsf{st}, \mathsf{leak}) = b] - 1/2\right|.$$

We say that an algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ is admissible for the adaptive security game if neither $\mathcal{A}_1$ nor $\mathcal{A}_2$ makes a key-generation query on any identity $\mathsf{id} \in \mathcal{J}$. We say Π_{bkIBE} satisfies adaptive security under bounded leakage with challenge parameter $k = k(\lambda, \ell)$ if for all efficient adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and every inverse polynomial advantage function $\varepsilon = 1/\mathsf{poly}(\lambda)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $\Pr[b' = 1] = \mathsf{negl}(\lambda)$ in the adaptive security game.

Remark 3.2 (Comparison with [DGSW22])**.** Beyond the extension from selective security to adaptive security, Definition 3.1 differs from the notion in [DGSW22, Definition 3] in an important manner. The definition in [DGSW22] says that a big-key IBE scheme satisfies (selective) security under bounded leakage if for all efficient adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and all *non-negligible* functions ε, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$\Pr[\forall \mathsf{id} \in \mathcal{J} : \mathsf{Adv}^{\mathsf{id}}(\mathsf{msk}, \mathsf{pp}, \mathsf{st}, \mathsf{leak}) \geq \varepsilon(\lambda)] = \mathsf{negl}(\lambda). \tag{3.2}$$

In contrast, our definition (Definition 3.1) requires the advantage threshold ε to be *inverse polynomial*. While "non-negligible" and "inverse-polynomial" may seem like a small distinction, it is an important one. Indeed, we can show that the definition is unsatisfiable if we require Eq. (3.2) to hold for all non-negligible functions ε. To wit, suppose ε is the following piecewise function:

$$\varepsilon(\lambda) = \begin{cases} 1 & \lambda \text{ is odd} \\ 0 & \lambda \text{ is even.} \end{cases}$$

Observe that $\varepsilon(\lambda)$ is non-negligible by construction. However, Eq. (3.2) cannot hold for any scheme with respect to ε. This is because for every adversary $\mathcal{A}$, and every even value of $\lambda \in \mathbb{N}$, it holds that

$$\mathsf{Adv}^{\mathsf{id}}(\mathsf{msk}, \mathsf{pp}, \mathsf{st}, \mathsf{leak}) \geq 0 = \varepsilon(\lambda).$$

This means that whenever λ is even, it follows that

$$\Pr[\forall \mathsf{id} \in \mathcal{J} : \mathsf{Adv}^{\mathsf{id}}(\mathsf{msk}, \mathsf{pp}, \mathsf{st}, \mathsf{leak}) \geq \varepsilon(\lambda)] = 1.$$

As such, we cannot bound the probability in Eq. (3.2) by a negligible function, and Eq. (3.2) does not hold. For this reason, the original definition from [DGSW22] is unsatisfiable. In this work, we require ε to be an inverse polynomial function $1/\mathsf{poly}(\lambda)$, where $\mathsf{poly}(\lambda)$ is a *fixed* polynomial. This rules out such pathological functions.

Advantage checking. In Definition 3.1, the output of the experiment requires checking whether Eq. (3.1) holds or not:

$$\forall \mathsf{id} \in \mathcal{J} : \mathsf{Adv}^{\mathsf{id}}(\mathsf{msk}, \mathsf{pp}, \mathsf{st}, \mathsf{leak}) \geq \varepsilon(\lambda),$$

where $\varepsilon(\lambda)$ is some advantage threshold. We note that in general, the exact advantage of an adversary is *not* efficiently-computable. As such, the challenger in Definition 3.1 cannot necessarily efficiently determine whether the adversary $\mathcal{A}$ is successful or not. While having an inefficient challenger is perfectly acceptable from a definitional standpoint, it introduces new challenges in the security analysis. Namely, given a candidate adversary $\mathcal{A}$, a reduction algorithm that uses $\mathcal{A}$ to solve some computational problem may not be able to determine whether $\mathcal{A}$ was successful or not. To address this problem, we define an alternative version of the adaptive security game where we replace the win condition (Eq. (3.1)) with an efficiently-checkable variant based on *estimating* the success probability of the adversary (Definition 3.3).[2] We then show in Theorem 3.4 that a scheme satisfying our alternative security game implies a scheme that is secure under our main definition (Definition 3.1). Then, in the remainder of this paper, we only consider Definition 3.3 where the output of the game is efficiently-computable.

Definition 3.3 (Adaptive Advantage-Checker Security)**.** Let Π_{bkIBE} be a big-key IBE scheme as in Definition 3.1. We define the following property:

- **Adaptive advantage-checker security under bounded leakage:** This security game is identical to the adaptive security game in Definition 3.1 except the output of the game is $b' = 1$ if

$$\forall \mathsf{id} \in \mathcal{J} : \mathsf{AdvCheck}^{\mathcal{A}_2}\left(1^\lambda, 1^{1/\varepsilon}, \mathsf{id}, \mathsf{msk}, \mathsf{pp}, \mathsf{st}, \mathsf{leak}\right) = 1$$

  and $b' = 0$ otherwise. The algorithm AdvCheck is defined as follows:

[2] We note here that this issue appears to have been glossed over in the previous work of [DGSW22] as their security proofs do not describe how the reduction algorithm uses the adversary's output to solve the underlying computational problem. We believe that their analysis can be repaired by formally defining a similar intermediary game with an efficiently-computable challenger.

Inputs: security parameter λ, advantage threshold $\varepsilon \in (0,1)$, identity $\mathsf{id} \in \mathcal{ID}_\lambda$, master secret key msk, public parameters pp, state st, string leak, and (oracle) access to an algorithm $\mathcal{A}$

- Let $T = \lambda/\varepsilon^2$ and initialize a counter $\mathsf{WINS} \leftarrow 0$.
- The advantage-checker algorithm now simulates T independent executions of experiment $\mathsf{Exp}^{\mathsf{id}}(\mathsf{msk}, \mathsf{pp}, \mathsf{st}, \mathsf{leak})$ for algorithm $\mathcal{A}$.
  1. Sample $\beta \xleftarrow{\text{R}} \{0,1\}$.
  2. Compute $\mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{pp}, \mathsf{id}, m_\beta)$, and start running algorithm $\mathcal{A}$ on input $(\mathsf{st}, \mathsf{id}, \mathsf{ct})$.
  3. Whenever algorithm $\mathcal{A}$ makes a key-generation query on an identity $\mathsf{id} \in \mathcal{ID}_\lambda$, compute $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$ and reply to $\mathcal{A}$ with the identity key $\mathsf{sk}_{\mathsf{id}}$.
  4. After $\mathcal{A}$ has finished making key-generation queries, it outputs a bit $\beta' \in \{0,1\}$.
  5. If $\beta = \beta'$, then increment $\mathsf{WINS} \leftarrow \mathsf{WINS} + 1$.
- Output 1 if $\left|\mathsf{WINS} - \frac{T}{2}\right| \geq \frac{\varepsilon T}{2}$ and 0 otherwise.

Figure 1: Function $\mathsf{AdvCheck}^{\mathcal{A}}(1^\lambda, 1^{1/\varepsilon}, \mathsf{id}, \mathsf{msk}, \mathsf{pp}, \mathsf{st}, \mathsf{leak})$

We say that an algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ is admissible for the k-adaptive advantage-checker security game if neither $\mathcal{A}_1$ nor $\mathcal{A}_2$ makes a key-generation query on any identity $\mathsf{id} \in \mathcal{J}$. The scheme Π_{bkIBE} satisfies adaptive advantage-checker security under bounded leakage with challenge parameter $k = k(\lambda, \ell)$ if for all efficient adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and every inverse polynomial advantage function $\varepsilon = 1/\mathsf{poly}(\lambda)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $\Pr[b' = 1] = \mathsf{negl}(\lambda)$ in the adaptive advantage-checker security game.

Theorem 3.4 (Adaptive Security from Adaptive Advantage-Checker Security). *Suppose* Π_{bkIBE} *is a big-key IBE scheme that satisfies adaptive advantage-checker security under bounded leakage with challenge parameter* $k = k(\lambda, \ell)$*. Then,* Π_{bkIBE} *satisfies adaptive security under bounded leakage with the same challenge parameter* k.

Proof. Let Hyb_0 be the adaptive security experiment from Definition 3.1 and Hyb_1 be the advantage checker security experiment from Definition 3.3. For an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and an advantage function ε, we write $\mathsf{Hyb}_i(\mathcal{A}, \varepsilon)$ to denote the output of Hyb_i with adversary $\mathcal{A}$ and advantage function ε. We now show that for all efficient adversaries $\mathcal{A}$ and all inverse polynomial advantage functions $\varepsilon = 1/\mathsf{poly}(\lambda)$, there exists a negligible function negl such that for all $\lambda \in \mathbb{N}$,

$$\Pr[\mathsf{Hyb}_0(\mathcal{A}, \varepsilon) = 1] \leq \Pr[\mathsf{Hyb}_1(\mathcal{A}, \varepsilon) = 1] + \mathsf{negl}(\lambda),$$

which proves the claim. By construction, the only difference between Hyb_0 and Hyb_1 is how the output bit $b' \in \{0,1\}$ is computed. Suppose in an execution of Hyb_0 that the output bit is 1. This means that for all $\mathsf{id} \in \mathcal{J}$,

$$\mathsf{Adv}^{\mathsf{id}}(\mathsf{msk}, \mathsf{pp}, \mathsf{st}, \mathsf{leak}) \geq \varepsilon. \tag{3.3}$$

Consider the output computed according to the specification of Hyb_1. The $\mathsf{AdvCheck}$ algorithm perfectly simulates T executions of $\mathsf{Exp}^{\mathsf{id}}$. For each $i \in [T]$, let $X_i \in \{0,1\}$ be the random variable for whether algorithm $\mathcal{A}_2$'s output is correct (i.e., if $\beta' = \beta$) on the i^{th} iteration. If Eq. (3.3) holds, then

$$|\mathbb{E}[X_i] - 1/2| = |\Pr[X_i = 1] - 1/2| \geq \varepsilon$$

In Hyb_1, we have $\mathsf{WINS} = \sum_{i \in [T]} X_i$ and since each X_i is identically distributed, it follows that

$$|\mathbb{E}[\mathsf{WINS}] - T/2| \geq \varepsilon T.$$

By Hoeffding's inequality (Fact 2.1),

$$\Pr[|\mathsf{WINS} - T/2| < \varepsilon T/2] \leq \Pr[|\mathsf{WINS} - \mathbb{E}[\mathsf{WINS}]| > \varepsilon T/2] \leq 2^{-\Omega(T\varepsilon^2/4)} = \mathsf{negl}(\lambda),$$

since $T = \lambda/\varepsilon^2$. Thus, if Eq. (3.3) holds, then with probability $1 - \mathsf{negl}(\lambda)$, $|\mathsf{WINS} - T/2| \geq \varepsilon T/2$ in an execution of $\mathsf{AdvCheck}^{\mathcal{A}_2}(1^\lambda, 1^{1/\varepsilon}, \mathsf{id}, \mathsf{msk}, \mathsf{pp}, \mathsf{st}, \mathsf{leak})$. In this case, $\mathsf{AdvCheck}$ outputs 1. By a union bound, if Eq. (3.3) holds for all $\mathsf{id} \in \mathcal{J}$, then $\mathsf{AdvCheck}$ also outputs 1 for all $\mathsf{id} \in \mathcal{J}$ with probability $1 - |\mathcal{J}| \cdot \mathsf{negl}(\lambda)$. If $\mathcal{A}$ is efficient, then the size of the challenge set $\mathcal{J}$ is polynomially-bounded, so we conclude that whenever experiment $\mathsf{Hyb}_0(\mathcal{A}, \varepsilon)$ outputs 1, then with probability $1 - \mathsf{negl}(\lambda)$, experiment $\mathsf{Hyb}_1(\mathcal{A}, \varepsilon)$ also outputs 1, and the claim follows. $\square$

Remark 3.5 (Challenge Parameter k)**.** The challenge parameter k in Definitions 3.1 and 3.3 determines the minimum size of the challenge set $\mathcal{J}$ as a function of the security parameter λ and the leakage parameter ℓ. A larger value of k increases the difficulty for the adversary while a small value of k makes the adversary's job simpler. In [DGSW22], the parameter k was set to be $\ell + 1$; namely, given ℓ bits of leakage, the adversary has to compromise at least $\ell + 1$ identities. In this work, we show multiple bits of leakage are necessary to compromise any single identity key. Namely, we show how to achieve challenge parameter $k = \ell/\mathsf{poly}(\lambda)$.

4 Adaptively-Secure Big-Key IBE from Indistinguishability Obfuscation

In this section, we describe how to construct an adaptively-secure big-key IBE scheme using indistinguishability obfuscation (Definition 2.13), an adaptively-secure IBE scheme (Definition 2.12), a NIZK proof for NP (Definition 2.17), a one-time dual-mode commitment scheme (Definition 2.15), and a pseudorandom generator (Definition 2.10).

Expanding hash function. First, we define the notion of an "expanding" hash function, which will be a useful building block in our constructions. At a high-level, an expanding hashing function $\mathcal{H} \colon \{0,1\}^\lambda \to [N]^d$ maps a string $x \in \{0,1\}^\lambda$ onto a set of elements $S \subseteq [N]$ of size $|S| = d$ with the property that for every collection of inputs $x_1, \ldots, x_k \in \{0,1\}^\lambda$, the set $\bigcup_{i \in [k]} \mathcal{H}(x_i)$ covers almost dk indices of the set $[N]$. In the context of our big-key IBE schemes, we will subdivide the master secret key into N blocks, and the secret key for an identity id will contain the blocks indexed by $\mathcal{H}(\mathsf{id})$. The security analysis will rely on the fact that for any set of k identities that the adversary can possibly corrupt, there will always exist at least one block of the master secret key that the adversary does not know. Namely, the number of blocks of the master secret key covered by every set of k identities is always *greater* than the amount of leakage the adversary is allowed on the master secret key. We now define the property formally, and show that such a hash function can be built from a disperser (Definition 2.4 and Fact 2.5).

Definition 4.1 (Expanding Hash Function)**.** We say a hash function $\mathcal{H} \colon \{0,1\}^\lambda \to [N]^d$ is (k, α)-expanding if there exists an explicit and efficient algorithm for computing $\mathcal{H}(x)$ in $\mathsf{poly}(\lambda, d, \log N)$ time, and moreover, for every collection of exactly k inputs $x_1, \ldots x_k \in \{0,1\}^\lambda$, it holds that $|\bigcup_{i \in [k]} \mathcal{H}(x_i)| \geq \alpha k$.

Lemma 4.2 (Expanding Hash Function). *There exists a constant $c \in \mathbb{N}$ such that for every $\lambda \in \mathbb{N}$, every constant $\delta \in (0, 1)$, and every polynomially-bounded function $t(\lambda) > \lambda^c$ where $t(\lambda)$ is a power of 4, there exists functions $\alpha = \omega(\log \lambda)$, $d = \mathsf{poly}(\lambda)$, and a (t, α)-expanding hash function $\mathcal{H} \colon \{0, 1\}^\lambda \to [N]^d$, where $\alpha t = (1 - \delta)N$.*

Proof. This follows immediately from Fact 2.5. Specifically, let $G = (L, R, E)$ be the construction from Fact 2.5 instantiated with parameters $n = \lambda$, $\varepsilon = \delta$, $k = (\log t)/2$, and $k_1 = 2k = \log t$. Then, G is a degree-D (t, ε)-disperser where $D = \mathsf{poly}(\lambda)$, $|L| = 2^\lambda$ and $|R| = t \cdot 2^{\Omega(\log \lambda)}$. We now construct the expanding hash function $\mathcal{H} \colon \{0, 1\}^\lambda \to [N]^d$ as follows:

- Set $N = |R| = t \cdot 2^{\Omega(\log \lambda)}$, $\alpha = (1 - \delta) \cdot 2^{\Omega(\log \lambda)}$, and $d = D = \mathsf{poly}(\lambda)$.
- For an input $x \in \{0, 1\}^\lambda$, define $\mathcal{H}(x)$ to be the indices of the nodes in the neighborhood of node $x \in G$ (here, we index the 2^λ nodes in L with a bit-string in $\{0, 1\}^\lambda$). Note that computing $\mathcal{H}(x)$ requires time $\mathsf{poly}(\lambda, d, \log N)$ since the disperser is explicit. Thus, $\mathcal{H}$ is efficiently-computable.

We now show the expanding property. This follows immediately from the fact that G is a disperser. Consider any set of t inputs $x_1, \ldots, x_t$. Let $S = \{x_1, \ldots, x_t\}$. Since G is a (t, ε)-disperser, and by construction of $\mathcal{H}$, it follows that

$$\left|\bigcup_{i \in [t]} \mathcal{H}(x_i)\right| = |N(S)| \geq (1 - \varepsilon) \cdot |R| = (1 - \delta)N = \alpha t,$$

where $N(S)$ denotes the neighborhood of S in G. To finish the proof we show the constraint on α and that t is a valid choice in Fact 2.5. Clearly $\alpha = (1 - \delta) \cdot 2^{\Omega(\log \lambda)} = \omega(\log \lambda)$ holds. It is also immediate that choosing $k_1 = 2k \geq k + O(\log^3 k)$ is sufficient. □

Big-key IBE construction. We now give our first construction of an adaptively-secure big-key IBE scheme.

Construction 4.3 (Big-Key IBE from $i\mathcal{O}$). Let $\lambda \in \mathbb{N}$ be a security parameter, $\mathcal{ID} = \{\mathcal{ID}_\lambda\}_{\lambda \in \mathbb{N}}$ be the identity space, $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda \in \mathbb{N}}$ be the message space, ℓ be the leakage parameter, $N = N(\lambda, \ell)$ be a key-size parameter, and $d = d(\lambda)$ be an output-size parameter. Our construction relies on the following primitives:

- Let $i\mathcal{O}$ be an indistinguishability obfuscation scheme. We will assume that all programs described here (and in the proof of Theorem 4.6) are padded to the size $\ell_C(\lambda)$ of the largest program among them.
- Let $\mathsf{PRG} \colon \{0, 1\}^\lambda \to \{0, 1\}^{2\lambda}$ be a pseudorandom generator. Note that the PRG is only used in the security analysis and does not appear in the main construction.
- Let $\mathcal{H} \colon \mathcal{ID}_\lambda \to [N]^d$ be a hash function. We interpret the output elements $[N]^d$ as an ordered list of d indices in $[N]$.
- Let $\Pi_{\mathsf{NIZK}} = (\mathsf{NIZK.Setup}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Verify})$ be a NIZK proof for NP.
- Let $\Pi_{\mathsf{Com}} = (\mathsf{Com.Setup}, \mathsf{Com.Commit}, \mathsf{Com.Verify})$ be a one-time dual-mode commitment scheme (Definition 2.15) with input space $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda \in \mathbb{N}}$, and let $\ell_x = \ell_x(\lambda)$ be the bit-length of an input.
- Let $\Pi_{\mathsf{IBE}} = (\mathsf{IBE.Setup}, \mathsf{IBE.KeyGen}, \mathsf{IBE.Encrypt}, \mathsf{IBE.Decrypt})$ be an IBE scheme with identity space $\mathcal{ID}$ and message space $\mathcal{X}^d$.

- For public parameters $\mathsf{pp}_{\mathsf{IBE}}$, define the NP relation $\mathcal{R}[\mathsf{pp}_{\mathsf{IBE}}]$ as follows:

> **Hard-wired:** public parameters $\mathsf{pp}_{\mathsf{IBE}}$
> **Statement:** a vector of common reference strings $\mathsf{crs}_{\mathsf{Com}}^{(I)}$ indexed by a set $I \subset \mathbb{N}$, a tuple of commitments $\vec{c}_I$, ciphertext ct, identity id
> **Witness:** strings $\vec{r}_I$, randomness ρ_{enc}, openings $\vec{\sigma}_I$
>
> Output 1 if all of the following conditions hold:
>
> – For each $i \in I$, $\mathsf{Com.Verify}(\mathsf{crs}_{\mathsf{Com}}^{(i)}, c_i, r_i, \sigma_i) = 1$;
> – $\mathsf{IBE.Encrypt}(\mathsf{pp}_{\mathsf{IBE}}, \mathsf{id}, \vec{r}_I; \rho_{\mathsf{enc}}) = \mathsf{ct}$
>
> Otherwise, output 0.

Figure 2: Relation $\mathcal{R}[\mathsf{pp}_{\mathsf{IBE}}]$.

Let $C_{\mathcal{R}}[\mathsf{pp}_{\mathsf{IBE}}]$ be the circuit computing the NP relation $\mathcal{R}[\mathsf{pp}_{\mathsf{IBE}}]$.

We now construct our big-key IBE scheme $\Pi_{\mathsf{bkIBE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ as follows:

- $\mathsf{Setup}(1^\lambda, 1^\ell)$: On input the security parameter λ and the leakage parameter ℓ, the setup algorithm proceeds as follows:
  1. Sample $(\mathsf{pp}_{\mathsf{IBE}}, \mathsf{msk}_{\mathsf{IBE}}) \leftarrow \mathsf{IBE.Setup}(1^\lambda)$ and $\mathsf{crs}_{\mathsf{NIZK}} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$.
  2. For each $i \in [N]$, sample a random string $r_i \xleftarrow{\text{R}} \{0,1\}^{\ell_x}$. Then, sample a common reference string $\mathsf{crs}_{\mathsf{Com}}^{(i)} \leftarrow \mathsf{Com.Setup}(1^\lambda, \mathsf{bind})$ and compute $(c_i, \sigma_i) \leftarrow \mathsf{Com.Commit}(\mathsf{crs}_{\mathsf{Com}}^{(i)}, r_i)$.

  Let $\vec{c} = (c_1, \ldots, c_N)$, $\vec{r} = (r_1, \ldots, r_N)$, and $\vec{\sigma} = (\sigma_1, \ldots, \sigma_N)$. For a set $I \subseteq [N]$, we write $\vec{c}_I$, $\vec{r}_I$, and $\vec{\sigma}_I$ to be the respective sub-vector of indices in I. Similarly, we define $\mathsf{crs}_{\mathsf{Com}}^{(I)} := \left(\mathsf{crs}_{\mathsf{Com}}^{(i)}\right)_{i \in I}$. Output

$$\mathsf{pp} = \left(\left\{\mathsf{crs}_{\mathsf{Com}}^{(i)}\right\}_{i \in [N]}, \vec{c}, \mathsf{crs}_{\mathsf{NIZK}}, \mathsf{pp}_{\mathsf{IBE}}\right) \quad \text{and} \quad \mathsf{msk} = (\mathsf{pp}, \vec{r}, \vec{\sigma}). \tag{4.1}$$

- $\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$: On input the master secret key msk (with components as in Eq. (4.1)) and an identity $\mathsf{id} \in \mathcal{ID}_\lambda$, the key generation algorithm proceeds as follows:
  1. Compute $I \leftarrow \mathcal{H}(\mathsf{id})$.
  2. Compute $\mathsf{ct} \leftarrow \mathsf{IBE.Encrypt}(\mathsf{pp}_{\mathsf{IBE}}, \mathsf{id}, \vec{r}_I; \rho_{\mathsf{enc}})$ where ρ_{enc} is the encryption randomness and $\vec{r}_I$ is as defined in Eq. (4.1).
  3. Compute $\pi \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_{\mathsf{NIZK}}, C_{\mathcal{R}}[\mathsf{pp}_{\mathsf{IBE}}], (\mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, \mathsf{ct}, \mathsf{id}), (\vec{r}_I, \rho_{\mathsf{enc}}, \vec{\sigma}_I))$, where $\mathsf{crs}_{\mathsf{Com}}^{(I)}$, $\vec{c}_I$, $\vec{r}_I$, and $\vec{\sigma}_I$ are as defined in Eq. (4.1).

  Output the identity secret key $\mathsf{sk}_{\mathsf{id}} = (\mathsf{ct}, \pi)$.

- $\mathsf{Encrypt}(\mathsf{pp}, \mathsf{id}, m)$: On input the public parameters pp, an identity $\mathsf{id} \in \mathcal{ID}_\lambda$ and a message $m \in \mathcal{M}_\lambda$, the encryption algorithm defines the following program:

Hard-wired: common reference string $\mathsf{crs}_{\mathsf{NIZK}}$, a vector of common reference strings $\mathsf{crs}_{\mathsf{Com}}^{(I)}$ indexed by a set $I \subset \mathbb{N}$, a circuit C, a tuple of commitments $\vec{c}_I$, message m, identity id

Inputs: ciphertext ct, proof π

1. If $\mathsf{NIZK.Verify}\big(\mathsf{crs}_{\mathsf{NIZK}}, C, \big(\mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, \mathsf{ct}, \mathsf{id}\big), \pi\big) = 1$, output m.
2. Otherwise, output $\perp$.

Figure 3: Program $\mathsf{Check\text{-}Bits}\big[\mathsf{crs}_{\mathsf{NIZK}}, C, \mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, m, \mathsf{id}\big]$.

The encryption algorithm then computes $I \leftarrow \mathcal{H}(\mathsf{id})$ and the obfuscated program

$$\widetilde{C} \leftarrow i\mathcal{O}(\mathsf{Check\text{-}Bits}[\mathsf{crs}_{\mathsf{NIZK}}, C_{\mathcal{R}}[\mathsf{pp}_{\mathsf{IBE}}], \mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, m, \mathsf{id}]).$$

It outputs the ciphertext $\mathsf{ct} = \widetilde{C}$.

- $\mathsf{Decrypt}(\mathsf{sk}_{\mathsf{id}}, \mathsf{id}, \mathsf{ct})$: On input an identity secret key $\mathsf{sk}_{\mathsf{id}}$, an identity $\mathsf{id} \in \mathcal{ID}_\lambda$, and a ciphertext $\mathsf{ct} = \widetilde{C}$, the decryption algorithm outputs $\widetilde{C}(\mathsf{sk}_{\mathsf{id}})$.

Theorem 4.4 (Correctness). *Suppose Π_{Com} is correct, $i\mathcal{O}$ is correct, and Π_{NIZK} is complete. Then, Construction 4.3 is correct.*

Proof. Take any security parameter λ, identity $\mathsf{id} \in \mathcal{ID}_\lambda$, and message m. Let $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$, where $\mathsf{pp} = \left(\{\mathsf{crs}_{\mathsf{Com}}^{(i)}\}_{i\in[N]}, \vec{c}, \mathsf{crs}_{\mathsf{NIZK}}, \mathsf{pp}_{\mathsf{IBE}}\right)$, $\mathsf{msk} = (\mathsf{pp}, \vec{r}, \vec{\sigma})$. Let $\mathsf{sk}_{\mathsf{id}} = (\mathsf{ct}, \pi) \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$ and $\widetilde{C} \leftarrow \mathsf{Encrypt}(\mathsf{pp}, \mathsf{id}, m)$. Consider the output of $\mathsf{Decrypt}(\mathsf{sk}_{\mathsf{id}}, \mathsf{id}, \widetilde{C})$:

- By construction of KeyGen and correctness of Π_{Com}, we have $((\mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, \mathsf{ct}, \mathsf{id}), (\vec{r}_I, \rho_{\mathsf{enc}}, \sigma_I)) \in \mathcal{R}[\mathsf{pp}_{\mathsf{IBE}}]$ and $\pi \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_{\mathsf{NIZK}}, C_{\mathcal{R}}[\mathsf{pp}_{\mathsf{IBE}}], (\mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, \mathsf{ct}, \mathsf{id}), (\vec{r}_I, \rho_{\mathsf{enc}}, \vec{\sigma}_I))$.

- By construction of Encrypt and $i\mathcal{O}$ correctness, $\widetilde{C}$ is a program which outputs the message m when the NIZK proof verifies on statement $(\mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, \mathsf{ct}, \mathsf{id})$ where ct is an input.

- By completeness of Π_{NIZK}, the proof π from KeyGen verifies and thus $\widetilde{C}(\mathsf{sk}_{\mathsf{id}}) = m$, as required. □

Theorem 4.5 (Efficiency). *If $\mathcal{H}$ runs in $\mathrm{poly}(\lambda, \log N)$-time, then Construction 4.3 is efficient.*

Proof. This holds by inspection and assumption on $\mathcal{H}$, since our other primitives run in $\mathrm{poly}(\lambda)$-time by definition. Furthermore, the KeyGen and Encrypt algorithms only needs to read $\mathrm{poly}(\lambda) \cdot d(\lambda)$ bits of the master secret key msk and/or the public parameters pp. The size of these quantities are independent of the leakage parameter ℓ. □

Theorem 4.6 (Adaptive Advantage-Checker Security under Bounded Leakage). *Suppose the following conditions hold:*

- *The obfuscator $i\mathcal{O}$ is secure.*

- *The hash function $\mathcal{H}$ is (k, α)-expanding, where $(1 - \eta)\alpha(\lambda)\ell_x(\lambda) \geq \lambda + \omega(\log \lambda)$ for some constant $\eta \in (0, 1)$.*

- *The IBE scheme* Π_{IBE} *satisfies correctness and adaptive semantic security.*
- *The NIZK* Π_{NIZK} *satisfies statistical soundness and computational zero-knowledge.*
- *The one-time dual-mode commitment scheme* Π_{Com} *satisfies mode indistinguishability and statistical binding in binding mode.*
- *The pseudorandom generator* PRG *is secure.*
- *There exists an explicit universal hash family* $\mathcal{H}_{\mathsf{fam}}$ *of size at most* $2^{\mathsf{poly}(\lambda)}$*, where each function* $h\colon \mathcal{X}_\lambda^d \to \{0,1\}^\lambda$ *has domain* $\mathcal{X}_\lambda^d$ *and range* $\{0,1\}^\lambda$*. Moreover, the extractor* $\mathsf{Ext}(x,h) = h(x)$ *is a* $(\lambda + \omega(\log \lambda), \mathsf{negl}(\lambda))$*-strong randomness extractor.*

Then for all polynomially-bounded and sufficiently large $\ell = \ell(\lambda)$*, Construction 4.3 is adaptively advantage-checker secure under bounded leakage with challenge parameter* $k \geq \frac{\ell}{\eta\alpha\ell_x}$*.*

Proof. We define a sequence of hybrid experiments, each parameterized (implicitly) by an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ and an advantage threshold function $\varepsilon = \varepsilon(\lambda)$:

- Hyb_0: This is the adaptive advantage-checker security game from Definition 3.1, which we recall in full below:
  - **Setup:** The challenger starts by sampling $(\mathsf{pp}_{\mathsf{IBE}}, \mathsf{msk}_{\mathsf{IBE}}) \leftarrow \mathsf{IBE.Setup}(1^\lambda)$, and $\mathsf{crs}_{\mathsf{NIZK}} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$. For all $i \in [N]$, it samples a random string $r_i \xleftarrow{\mathrm{R}} \{0,1\}^{\ell_x}$, a CRS $\mathsf{crs}_{\mathsf{Com}}^{(i)} \leftarrow \mathsf{Com.Setup}(1^\lambda, \mathsf{bind})$, and computes $(c_i, \sigma_i) \leftarrow \mathsf{Com.Commit}(\mathsf{crs}_{\mathsf{Com}}^{(i)}, r_i)$. For $\vec{c} = (c_1, \ldots, c_N)$, $\vec{r} = (r_1, \ldots, r_N)$, and $\vec{\sigma} = (\sigma_1, \ldots, \sigma_N)$, the challenger sets
  $$\mathsf{pp} = \left(\left\{\mathsf{crs}_{\mathsf{Com}}^{(i)}\right\}_{i \in [N]}, \vec{c}, \mathsf{crs}_{\mathsf{NIZK}}, \mathsf{pp}_{\mathsf{IBE}}\right) \quad \text{and} \quad \mathsf{msk} = (\mathsf{pp}, \vec{r}, \vec{\sigma})$$
  and gives pp to $\mathcal{A}$.
  - **Pre-leakage queries:** When algorithm $\mathcal{A}_1$ makes a query on $\mathsf{id} \in \mathcal{ID}_\lambda$, the challenger computes $I \leftarrow \mathcal{H}(\mathsf{id})$, $\mathsf{ct} \leftarrow \mathsf{IBE.Encrypt}(\mathsf{pp}_{\mathsf{IBE}}, \mathsf{id}, \vec{r}_I; \rho_{\mathsf{enc}})$, and
  $$\pi \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_{\mathsf{NIZK}}, C_{\mathcal{R}}[\mathsf{pp}_{\mathsf{IBE}}], (\mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, \mathsf{ct}, \mathsf{id}), (\vec{r}_I, \rho_{\mathsf{enc}}, \vec{\sigma}_I)),$$
  where ρ_{enc} is (fresh) encryption randomness. The challenger replies with $\mathsf{sk}_{\mathsf{id}} = (\mathsf{ct}, \pi)$.
  - **Leakage:** After $\mathcal{A}_1$ outputs the description of an efficiently-computable leakage function f, the challenger replies with $\mathsf{leak} \leftarrow f(\mathsf{msk})$.
  - **Post-leakage queries:** The challenger responds to post-leakage key queries exactly as in the pre-leakage phase.
  - **Challenge:** Algorithm $\mathcal{A}_1$ outputs a set $\mathcal{J} \subseteq \mathcal{ID}_\lambda$ of size k, two messages m_0, m_1, and a state st.
  - **Output:** The output of Hyb_0 is $b' = 1$ if $\mathcal{A}$ is admissible and
  $$\forall \mathsf{id} \in \mathcal{J} : \mathsf{AdvCheck}^{\mathcal{A}_2}(1^\lambda, 1^{1/\varepsilon}, \mathsf{id}, \mathsf{msk}, \mathsf{pp}, \mathsf{st}, \mathsf{leak}) = 1, \tag{4.2}$$
  and $b' = 0$ otherwise. The advantage-checker algorithm AdvCheck is defined as follows:

Inputs: security parameter λ, threshold $\varepsilon \in (0,1)$, identity id $\in \mathcal{ID}_\lambda$, master secret key msk $= (\mathsf{pp}, \vec{r}, \vec{\sigma})$, public parameters pp $= \left(\{\mathsf{crs}_{\mathsf{Com}}^{(i)}\}_{i\in[N]}, \vec{c}, \mathsf{crs}_{\mathsf{NIZK}}, \mathsf{pp}_{\mathsf{IBE}}\right)$, state st, string leak, and (oracle) access to an algorithm $\mathcal{A}$

- Let $T = \lambda/\varepsilon^2$ and initialize a counter $K \leftarrow 0$.
- The advantage-checker algorithm now simulates T independent executions of experiment $\mathsf{Exp}^{\mathsf{id}}(\mathsf{msk}, \mathsf{pp}, \mathsf{st}, \mathsf{leak})$ for algorithm $\mathcal{A}$.
  1. Sample $\beta \xleftarrow{R} \{0,1\}$.
  2. Compute $\widetilde{C} \leftarrow i\mathcal{O}(\mathsf{Check\text{-}Bits}[\mathsf{crs}_{\mathsf{NIZK}}, C_{\mathcal{R}}[\mathsf{pp}_{\mathsf{IBE}}], \mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, m_\beta, \mathsf{id}])$, where $I \leftarrow \mathcal{H}(\mathsf{id})$. Set ct $= \widetilde{C}$ and start running algorithm $\mathcal{A}$ on input (st, id, ct).
  3. Whenever algorithm $\mathcal{A}$ makes a key-generation query on an identity id $\in \mathcal{ID}_\lambda$, compute $I \leftarrow \mathcal{H}(\mathsf{id})$, ct $\leftarrow \mathsf{IBE.Encrypt}(\mathsf{pp}_{\mathsf{IBE}}, \mathsf{id}, \vec{r}_I; \rho_{\mathsf{enc}})$, and $\pi \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_{\mathsf{NIZK}}, C_{\mathcal{R}}[\mathsf{pp}_{\mathsf{IBE}}], (\mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, \mathsf{ct}, \mathsf{id}), (\vec{r}_I, \rho_{\mathsf{enc}}, \vec{\sigma}_I))$, where ρ_{enc} is (fresh) encryption randomness. Reply to $\mathcal{A}$ with the identity key $\mathsf{sk}_{\mathsf{id}} = (\mathsf{ct}, \pi)$.
  4. After $\mathcal{A}$ has finished making key-generation queries, it outputs a bit $\beta' \in \{0,1\}$.
  5. If $\beta = \beta'$, then increment $K \leftarrow K + 1$.
- Output 1 if $K \geq \frac{T}{2} + \frac{\varepsilon T}{2}$ and 0 otherwise.

Figure 4: Function $\mathsf{AdvCheck}^{\mathcal{A}}(1^\lambda, 1^{1/\varepsilon}, \mathsf{id}, \mathsf{msk}, \mathsf{pp}, \mathsf{st}, \mathsf{leak})$ in Construction 4.3

- Hyb_1: Same as Hyb_0 except the challenger now samples $h \xleftarrow{R} \mathcal{H}_{\mathsf{fam}}$ at setup time and for each id $\in \mathcal{J}$, the challenger constructs the challenge ciphertext $\widetilde{C}$ in $\mathsf{AdvCheck}^{\mathcal{A}_2}(1^\lambda, 1^{1/\varepsilon}, \mathsf{id}, \mathsf{msk}, \mathsf{pp}, \mathsf{st}, \mathsf{leak})$ using the following modified procedure:
  1. The challenger samples $\beta \xleftarrow{R} \{0,1\}$, computes $I \leftarrow \mathcal{H}(\mathsf{id})$, and computes the components $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{IBE.KeyGen}(\mathsf{msk}, \mathsf{id}), u \leftarrow h(\vec{r}_I), t \leftarrow \mathsf{PRG}(u)$.
  2. The challenger defines the program Check-CT as follows:

Hard-wired: common reference string $\mathsf{crs}_{\mathsf{NIZK}}$, a vector of common reference strings $\mathsf{crs}_{\mathsf{Com}}^{(I)}$ indexed by a set $I \subset \mathbb{N}$, a circuit C, a tuple of commitments $\vec{c}_I$, message m, identity secret key $\mathsf{sk}_{\mathsf{id}}$, identity id, seed h, bit-string t
Inputs: ciphertext ct, proof π
Output m if the following hold and $\perp$ otherwise:

– $\mathsf{NIZK.Verify}(\mathsf{crs}_{\mathsf{NIZK}}, C, (\mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, \mathsf{ct}, \mathsf{id}), \pi) = 1$; and
– $\mathsf{PRG}(h(\mathsf{IBE.Decrypt}(\mathsf{sk}_{\mathsf{id}}, \mathsf{id}, \mathsf{ct}))) = t$.

Figure 5: Program $\mathsf{Check\text{-}CT}[\mathsf{crs}_{\mathsf{NIZK}}, C, \mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, m, \mathsf{sk}_{\mathsf{id}}, \mathsf{id}, h, t]$.

Finally, it sets $\widetilde{C} \leftarrow i\mathcal{O}(\mathsf{Check\text{-}CT}[\mathsf{crs}_{\mathsf{NIZK}}, C_{\mathcal{R}}[\mathsf{pp}_{\mathsf{IBE}}], \mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, m_\beta, \mathsf{sk}_{\mathsf{id}}, \mathsf{id}, h, t])$.

The remainder of AdvCheck proceeds as in Hyb_0. We refer to these ciphertexts as *semi-functional* ciphertexts.

- Hyb_2: Same as Hyb_1 except the challenger simulates the NIZK proofs when answering key-generation queries. Specifically, let $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ be the zero-knowledge simulator associated with Π_{NIZK}. The experiment now proceeds as follows:
  - **Setup:** The challenger now samples $(\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{st}_{\mathcal{S}}) \leftarrow \mathcal{S}_1(1^\lambda)$.
  - **Key-generation queries:** Whenever $\mathcal{A}_1$ makes a key-generation query (in the pre-leakage or the post-leakage phase) or $\mathcal{A}_2$ makes a key-generation query (in $\mathsf{AdvCheck}^{\mathcal{A}_2}$) on $\mathsf{id} \in \mathcal{ID}_\lambda$, the challenger now constructs the NIZK proof π as $\pi \leftarrow \mathcal{S}_2(\mathsf{st}_{\mathcal{S}}, C_{\mathcal{R}}[\mathsf{pp}_{\mathsf{IBE}}], (\mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, \mathsf{ct}, \mathsf{id}))$.
- Hyb_3: Same as Hyb_2 except the challenger samples the commitment CRS in hiding mode and simulates the commitments and openings:
  - **Setup:** For all $i \in [N]$, the challenger now samples the commitment components $(\mathsf{crs}_{\mathsf{Com}}^{(i)}, c_i, \sigma_i)$ as $(\mathsf{crs}_{\mathsf{Com}}^{(i)}, \mathsf{td}_{\mathsf{Com}}^{(i)}, c_i) \leftarrow \mathsf{Com.Setup}(1^\lambda, \mathsf{hide}), \sigma_i \leftarrow \mathcal{S}_{\mathsf{open}}(\mathsf{td}_{\mathsf{Com}}^{(i)}, r_i)$.
- Hyb_4: Same as Hyb_3 except the challenger changes the distribution of secret keys when answering key-generation queries:
  - **Key-generation queries:** Whenever $\mathcal{A}_1$ makes a key-generation query (in the pre-leakage or the post-leakage phase) or $\mathcal{A}_2$ makes a key-generation query (in $\mathsf{AdvCheck}^{\mathcal{A}_2}$) on $\mathsf{id} \in \mathcal{ID}_\lambda$, the challenger now computes ct as $\mathsf{ct} \leftarrow \mathsf{IBE.Encrypt}(\mathsf{pp}_{\mathsf{IBE}}, \mathsf{id}, 0^{d\ell_x}; \rho_{\mathsf{enc}})$.

  We refer to these keys as *semi-functional* keys.
- Hyb_5: Same as Hyb_4 except for all $\mathsf{id} \in \mathcal{J}$, the challenger samples $u \xleftarrow{\mathrm{R}} \{0,1\}^\lambda$ at the counter initialization step in the procedure $\mathsf{AdvCheck}(\mathsf{id})$ and uses it to construct all challenge ciphertexts in $\mathsf{AdvCheck}(\mathsf{id})$. Moreover, the output of this experiment is $b' = 1$ if $\mathcal{A}$ is admissible and

  $$\exists \mathsf{id} \in \mathcal{J} : \mathsf{AdvCheck}^{\mathcal{A}_2}(1^\lambda, 1^{1/\varepsilon}, \mathsf{id}, \mathsf{msk}, \mathsf{pp}, \mathsf{st}, \mathsf{leak}) = 1, \tag{4.3}$$

  and $b' = 0$ otherwise. In other words, Eq. (4.3) replaces Eq. (4.2) as the condition that determines the output of experiment.
- Hyb_6: Same as Hyb_5 except for all $\mathsf{id} \in \mathcal{J}$ the challenger samples $t \xleftarrow{\mathrm{R}} \{0,1\}^{2\lambda}$ at the counter initialization step in the procedure $\mathsf{AdvCheck}(\mathsf{id})$.
- Hyb_7: Same as Hyb_6 except for all $\mathsf{id} \in \mathcal{J}$ the challenger constructs the challenge ciphertext $\widetilde{C}$ as $\widetilde{C} \leftarrow i\mathcal{O}(\mathsf{Bot})$ in the procedure $\mathsf{AdvCheck}(\mathsf{id})$, where Bot is a program that outputs $\perp$ on all inputs.

For convenience, we will refer to $\mathsf{AdvCheck}^{\mathcal{A}}(1^\lambda, 1^{1/\varepsilon}, \mathsf{id}, \mathsf{msk}, \mathsf{pp}, \mathsf{st}, \mathsf{leak})$ as $\mathsf{AdvCheck}(\mathsf{id})$ when the non-id parameters are fixed in a given context. For an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, we write $\mathsf{Hyb}_i(\mathcal{A}, \varepsilon)$ to denote the output of Hyb_i with adversary $\mathcal{A}$ and inner threshold function ε. Our goal is to show that for all efficient adversaries $\mathcal{A}$ and all inverse polynomial functions $\varepsilon = 1/\mathsf{poly}(\lambda)$, $\Pr[\mathsf{Hyb}_0(\mathcal{A}, \varepsilon) = 1] = \mathsf{negl}(\lambda)$. We now analyze each pair of adjacent experiments:

Lemma 4.7. *Suppose $i\mathcal{O}$ satisfies indistinguishability obfuscation security, Π_{NIZK} satisfies statistical soundness, Π_{Com} satisfies statistical binding in binding mode, and Π_{IBE} satisfies correctness. Then, for all efficient and* *admissible adversaries $\mathcal{A}$ and inverse polynomial functions $\varepsilon = 1/\mathsf{poly}(\lambda)$, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all $\lambda \in \mathbb{N}$,*

$$\Pr[\mathsf{Hyb}_1(\mathcal{A}, \varepsilon) = 1] \geq \Pr[\mathsf{Hyb}_0(\mathcal{A}, \varepsilon) = 1] - \mathsf{negl}(\lambda).$$

Proof. We define a sequence of intermediate hybrids:

- $\mathsf{Hyb}_{0,1,0}$: Same as Hyb_0. In particular, the challenge ciphertexts in the procedure $\mathsf{AdvCheck}(\mathsf{id})$ are sampled as $\widetilde{C} \leftarrow i\mathcal{O}(\mathsf{Check\text{-}Bits}[\mathsf{crs}_{\mathsf{NIZK}}, C_{\mathcal{R}}[\mathsf{pp}_{\mathsf{IBE}}], \mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, m_\beta, \mathsf{id}])$, where $I \leftarrow \mathcal{H}(\mathsf{id})$.
- $\mathsf{Hyb}_{0,i,j}$: Same as Hyb_0 except for all (i', j') such that $i' < i$ or $i' = i, j' \leq j$, the challenger sets $\mathsf{id} = \mathcal{J}[i']$, and samples the challenge ciphertext in the j'^{th} execution of $\mathsf{Exp}^{\mathsf{id}}$ in $\mathsf{AdvCheck}$ as

$$\widetilde{C} \leftarrow i\mathcal{O}(\mathsf{Check\text{-}CT}[\mathsf{crs}_{\mathsf{NIZK}}, C_{\mathcal{R}}[\mathsf{pp}_{\mathsf{IBE}}], \mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, m_\beta, \mathsf{sk}_{\mathsf{id}}, \mathsf{id}, h, t]),$$

  where $\beta \xleftarrow{\mathrm{R}} \{0, 1\}, I \leftarrow \mathcal{H}(\mathsf{id}), \mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{IBE.KeyGen}(\mathsf{msk}, \mathsf{id}), u \leftarrow h(\vec{r}_I), t \leftarrow \mathsf{PRG}(u)$ as in Hyb_1. Note that $\mathsf{Hyb}_{0,k,T}$ is the same as Hyb_1 and that $\mathsf{Hyb}_{0,i,T}$ is the same as $\mathsf{Hyb}_{0,i+1,0}$ for $i \in [k-1]$.

We now appeal to the conditions in Lemma 4.7 to show that for all $i \in [k], j \in [T]$ we show that $\mathsf{Hyb}_{0,i,j}$ and $\mathsf{Hyb}_{0,i,j-1}$ are computationally indistinguishable.

Claim 4.8. *Suppose the conditions in Lemma 4.7 hold. Then for all $i \in [k], j \in [T]$, all efficient and admissible adversaries $\mathcal{A}$ and inverse polynomial functions $\varepsilon = 1/\mathsf{poly}(\lambda)$, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all $\lambda \in \mathbb{N}$,*

$$\Pr[\mathsf{Hyb}_{0,i,j}(\mathcal{A}, \varepsilon) = 1] \geq \Pr[\mathsf{Hyb}_{0,i,j-1}(\mathcal{A}, \varepsilon) = 1] - \mathsf{negl}(\lambda).$$

Proof. Suppose there exists an efficient adversary $\mathcal{A}$ that distinguishes $\mathsf{Hyb}_{0,i,j}$ and $\mathsf{Hyb}_{0,i,j-1}$ with non-negligible probability δ. We use $\mathcal{A}$ to construct an algorithm $\mathcal{B}$ that breaks $i\mathcal{O}$ security:

1. Algorithm $\mathcal{B}$ runs the setup, leakage, challenge, key-generation phases as in Hyb_0 with $\mathcal{A}$. In particular:

   (a) Algorithm $\mathcal{B}$ starts by sampling $h \xleftarrow{\mathrm{R}} \mathcal{H}_{\mathsf{fam}}$, $(\mathsf{pp}_{\mathsf{IBE}}, \mathsf{msk}_{\mathsf{IBE}}) \leftarrow \mathsf{IBE.Setup}(1^\lambda)$, and $\mathsf{crs}_{\mathsf{NIZK}} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$. For all $i \in [N]$, algorithm $\mathcal{B}$ samples the components $r_i \xleftarrow{\mathrm{R}} \{0, 1\}^{\ell_x}$, $\mathsf{crs}_{\mathsf{Com}}^{(i)} \leftarrow \mathsf{Com.Setup}(1^\lambda, \mathsf{bind})$, and computes $(c_i, \sigma_i) \leftarrow \mathsf{Com.Commit}(\mathsf{crs}_{\mathsf{Com}}, r_i)$. Algorithm $\mathcal{B}$ sets

   $$\mathsf{pp} = \left(\{\mathsf{crs}_{\mathsf{Com}}^{(i)}\}_{i \in [N]}, \vec{c}, \mathsf{crs}_{\mathsf{NIZK}}, \mathsf{pp}_{\mathsf{IBE}}\right) \quad \text{and} \quad \mathsf{msk} = (\mathsf{pp}, \vec{r}, \vec{\sigma})$$

   and gives pp to $\mathcal{A}$. The components $\vec{c}, \vec{r}$, and $\vec{\sigma}$ are derived as in Eq. (4.1).

   (b) When algorithm $\mathcal{A}$ makes a key-generation query on $\mathsf{id} \in \mathcal{ID}_\lambda$, algorithm $\mathcal{B}$ computes the plain IBE ciphertext $\mathsf{ct} \leftarrow \mathsf{IBE.Encrypt}(\mathsf{pp}_{\mathsf{IBE}}, \mathsf{id}, r_I; \rho_{\mathsf{enc}})$ and the proof

   $$\pi \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_{\mathsf{NIZK}}, C_{\mathcal{R}}[\mathsf{pp}_{\mathsf{IBE}}], (\mathsf{crs}_{\mathsf{Com}}^{(I)}, c_I, \mathsf{ct}, \mathsf{id}), (r_I, \rho_{\mathsf{enc}}, \sigma_I)),$$

   where ρ_{enc} is (fresh) encryption randomness and $I \leftarrow \mathcal{H}(\mathsf{id})$. Algorithm $\mathcal{B}$ replies with $\mathsf{sk}_{\mathsf{id}} = (\mathsf{ct}, \pi)$.

   (c) When algorithm $\mathcal{A}$ outputs an efficiently-computable leakage function f, algorithm $\mathcal{B}$ replies with $\mathsf{leak} \leftarrow f(\mathsf{msk})$. In the challenge phase, algorithm $\mathcal{A}$ outputs $\mathcal{J} \subseteq \mathcal{ID}_\lambda$ of size k, messages m_0, m_1, and a state st.

2. For all (i', j') such that $i' < i$ or $i' = i, j' < j$, algorithm $\mathcal{B}$ computes $I \leftarrow \mathcal{H}(\mathcal{J}[i']), u \leftarrow h(\vec{r}_I), t \leftarrow \mathsf{PRG}(u)$ and samples $\beta \xleftarrow{\mathrm{R}} \{0,1\}, \mathsf{sk}_{\mathcal{J}[i']} \leftarrow \mathsf{IBE.KeyGen}(\mathsf{msk}, \mathcal{J}[i'])$. Algorithm $\mathcal{B}$ computes the challenge ciphertext in the j'^{th} execution of $\mathsf{Exp}^{\mathcal{J}[i']}$ in $\mathsf{AdvCheck}(\mathcal{J}[i'])$ as
$$\widetilde{C} \leftarrow i\mathcal{O}(\mathsf{Check\text{-}CT}[\mathsf{crs}_{\mathsf{NIZK}}, C_{\mathcal{R}}[\mathsf{pp}_{\mathsf{IBE}}], \mathsf{crs}_{\mathsf{Com}}^{(I)}, c_I, m_\beta, \mathsf{sk}_{\mathcal{J}[i']}, \mathcal{J}[i'], s, t]).$$

3. For the j^{th} execution of $\mathsf{Exp}^{\mathcal{J}[i]}$ in $\mathsf{AdvCheck}(\mathcal{J}[i])$, algorithm $\mathcal{B}$ samples the components $\beta \xleftarrow{\mathrm{R}} \{0,1\}, I \leftarrow \mathcal{H}(\mathcal{J}[i]), \mathsf{sk}_{\mathcal{J}[i]} \leftarrow \mathsf{IBE.KeyGen}(\mathsf{msk}, \mathcal{J}[i]), u \leftarrow h(\vec{r}_I), t \leftarrow \mathsf{PRG}(u)$. Algorithm $\mathcal{B}$ sets
$$C_0 = \mathsf{Check\text{-}Bits}[\mathsf{crs}_{\mathsf{NIZK}}, C_{\mathcal{R}}[\mathsf{pp}_{\mathsf{IBE}}], \mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, m_\beta, \mathcal{J}[i]]$$
and
$$C_1 = \mathsf{Check\text{-}CT}[\mathsf{crs}_{\mathsf{NIZK}}, C_{\mathcal{R}}[\mathsf{pp}_{\mathsf{IBE}}], \mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, m_\beta, \mathsf{sk}_{\mathcal{J}[i]}, \mathcal{J}[i], s, t].$$
Algorithm $\mathcal{B}$ submits (C_0, C_1) to the $i\mathcal{O}$ challenger, gets back program P, and uses P as the challenge ciphertext in this execution.

4. In the remaining executions of $\mathsf{Exp}^{\mathsf{id}}$ in procedure $\mathsf{AdvCheck}(\mathsf{id})$, algorithm $\mathcal{B}$ samples challenge ciphertexts as $\widetilde{C} \leftarrow i\mathcal{O}(\mathsf{Check\text{-}Bits}[\mathsf{crs}_{\mathsf{NIZK}}, C_{\mathcal{R}}[\mathsf{pp}_{\mathsf{IBE}}], \mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, m_\beta, \mathsf{id}])$ for $\beta \xleftarrow{\mathrm{R}} \{0,1\}, I \leftarrow \mathcal{H}(\mathsf{id})$. Algorithm $\mathcal{B}$ outputs whatever the experiment outputs.

If $P \leftarrow i\mathcal{O}(C_0)$, algorithm $\mathcal{B}$ simulates $\mathsf{Hyb}_{0,i,j-1}(\mathcal{A}, \varepsilon)$. If $P \leftarrow i\mathcal{O}(C_1)$, algorithm $\mathcal{B}$ simulates $\mathsf{Hyb}_{0,i,j}(\mathcal{A}, \varepsilon)$. All that remains is to show that (C_0, C_1) are functionally equivalent with overwhelming probability. In particular, it suffices to show that
$$\mathsf{NIZK.Verify}(\mathsf{crs}_{\mathsf{NIZK}}, C_{\mathcal{R}}[\mathsf{pp}_{\mathsf{IBE}}], (\mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, \mathsf{ct}, \mathsf{id}), \pi) = 1 \implies \mathsf{IBE.Decrypt}(\mathsf{sk}_{\mathsf{id}}, \mathsf{id}, \mathsf{ct}) = r_I$$
with $1 - \mathsf{negl}(\lambda)$ probability over the choice of pp, where $I \leftarrow \mathcal{H}(\mathsf{id}), \mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{IBE.KeyGen}(\mathsf{msk}, \mathsf{id})$. This is sufficient since PRG and Ext are deterministic. By NIZK verification and statistical soundness, it must be that the statement $(\mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, \mathsf{ct}, \mathsf{id})$ is true with overwhelming probability over the choice of $\mathsf{crs}_{\mathsf{NIZK}}$. By statistical binding, it also must be the case that $\vec{c}_I$ opens to only $\vec{r}_I$ except with negligible probability over the choice of $\mathsf{crs}_{\mathsf{Com}}^{(I)}$ since $\mathcal{R}[\mathsf{pp}_{\mathsf{IBE}}]$ checks that the commitments verify. Since $\vec{r}_I$ must be the corresponding component of the witness for the statement to be true, ct must be an encryption of $\vec{r}_I$. By plain IBE correctness, we have $\mathsf{IBE.Decrypt}(\mathsf{sk}_{\mathsf{id}}, \mathsf{id}, \mathsf{ct}) = \vec{r}_I$, as desired. Since functional equivalence is satisfied, algorithm $\mathcal{B}$ breaks $i\mathcal{O}$ security with advantage δ. □

Since $\mathsf{Hyb}_{0,i,T}$ is identical to $\mathsf{Hyb}_{0,i+1,0}$ for $i \in [k-1]$, the lemma follows from Claim 4.8 and a standard hybrid argument. □

Lemma 4.9. *Suppose Π_{NIZK} satisfies computational zero-knowledge. Then, for all efficient and admissible adversaries $\mathcal{A}$ and inverse polynomial functions $\varepsilon = 1/\mathrm{poly}(\lambda)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,*
$$\Pr[\mathsf{Hyb}_2(\mathcal{A}, \varepsilon) = 1] \geq \Pr[\mathsf{Hyb}_1(\mathcal{A}, \varepsilon) = 1] - \mathsf{negl}(\lambda).$$

Proof. Suppose there exists an efficient adversary $\mathcal{A}$ that distinguishes Hyb_1 and Hyb_2 with non-negligible probability δ. We use $\mathcal{A}$ to construct an algorithm $\mathcal{B}$ that breaks computational zero-knowledge:

1. At the beginning of the game, algorithm $\mathcal{B}$ gets $(1^\lambda, \mathsf{crs}_{\mathsf{NIZK}})$ from the ZK challenger. Algorithm $\mathcal{B}$ samples $h \xleftarrow{\mathrm{R}} \mathcal{H}_{\mathrm{fam}}$ and $(\mathsf{pp}_{\mathsf{IBE}}, \mathsf{msk}_{\mathsf{IBE}}) \leftarrow \mathsf{IBE.Setup}(1^\lambda)$. For all $i \in [N]$, algorithm $\mathcal{B}$ samples $r_i \xleftarrow{\mathrm{R}} \{0,1\}^{\ell_x}$, $\mathsf{crs}_{\mathsf{Com}}^{(i)} \leftarrow \mathsf{Com.Setup}(1^\lambda, \mathsf{bind})$, and computes $(c_i, \sigma_i) \leftarrow \mathsf{Com.Commit}(\mathsf{crs}_{\mathsf{Com}}, r_i)$. Algorithm $\mathcal{B}$ sets
$$\mathsf{pp} = \left(\{\mathsf{crs}_{\mathsf{Com}}^{(i)}\}_{i\in[N]}, \vec{c}, \mathsf{crs}_{\mathsf{NIZK}}, \mathsf{pp}_{\mathsf{IBE}}\right) \quad \text{and} \quad \mathsf{msk} = (\mathsf{pp}, \vec{r}, \vec{\sigma})$$
and gives pp to $\mathcal{A}$. The components $\vec{c}, \vec{r}$, and $\vec{\sigma}$ are derived as in Eq. (4.1).

2. When algorithm $\mathcal{A}$ makes a key-generation query on $\mathsf{id} \in \mathcal{ID}_\lambda$, algorithm $\mathcal{B}$ computes the plain IBE ciphertext $\mathsf{ct} \leftarrow \mathsf{IBE.Encrypt}(\mathsf{pp}_{\mathsf{IBE}}, \mathsf{id}, \vec{r}_I; \rho_{\mathsf{enc}})$ for $I \leftarrow \mathcal{H}(\mathsf{id})$ and randomness ρ_{enc}. Algorithm $\mathcal{B}$ then queries the proof oracle in the ZK game with input $(C_{\mathcal{R}}[\mathsf{pp}_{\mathsf{IBE}}], (\mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, \mathsf{ct}, \mathsf{id}), (\vec{r}_I, \rho_{\mathsf{enc}}, \vec{\sigma}_I))$ and gets back π. Algorithm $\mathcal{B}$ gives (ct, π) to algorithm $\mathcal{A}$.

3. Algorithm $\mathcal{B}$ runs the remainder of the experiment as in Hyb_1 with algorithm $\mathcal{A}$:
   (a) When algorithm $\mathcal{A}$ outputs an efficiently-computable leakage function f, algorithm $\mathcal{B}$ replies with $\mathsf{leak} = f(\mathsf{msk})$. In the challenge phase, algorithm $\mathcal{A}$ outputs $\mathcal{J} \subseteq \mathcal{ID}_\lambda$ of size at least k, messages m_0, m_1, and a state st.
   (b) For all $\mathsf{id} \in \mathcal{J}$, the algorithm $\mathcal{B}$ constructs the challenge ciphertext $\widetilde{C}$ in the procedure $\mathsf{AdvCheck}(\mathsf{id})$ as
$$\widetilde{C} \leftarrow i\mathcal{O}(\mathsf{Check\text{-}CT}[\mathsf{crs}_{\mathsf{NIZK}}, C_{\mathcal{R}}[\mathsf{pp}_{\mathsf{IBE}}], \mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, m_\beta, \mathsf{sk}_{\mathsf{id}}, \mathsf{id}, h, t])$$
   where $\beta \xleftarrow{\mathrm{R}} \{0,1\}, I \leftarrow \mathcal{H}(\mathsf{id}), \mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{IBE.KeyGen}(\mathsf{msk}, \mathsf{id}), u \leftarrow h(\vec{r}_I), t \leftarrow \mathsf{PRG}(u)$. Algorithm $\mathcal{B}$ outputs the output of the experiment.

If the CRS is generated using $\mathsf{NIZK.Setup}$ and the proofs are sampled according to $\mathsf{NIZK.Prove}$, then algorithm $\mathcal{B}$ simulates $\mathsf{Hyb}_1(\mathcal{A}, \varepsilon)$. On the other hand, if the CRS and proofs are generated using the simulator $\mathcal{S}$, then algorithm $\mathcal{B}$ simulates the $\mathsf{Hyb}_2(\mathcal{A}, \varepsilon)$. Thus, algorithm $\mathcal{B}$ breaks computational zero-knowledge with advantage δ. □

Lemma 4.10. *Suppose Π_{Com} satisfies mode indistinguishability. Then, for all admissible adversaries $\mathcal{A}$ and inverse polynomial functions $\varepsilon = 1/\mathsf{poly}(\lambda)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,*
$$\Pr[\mathsf{Hyb}_3(\mathcal{A}, \varepsilon) = 1] \geq \Pr[\mathsf{Hyb}_2(\mathcal{A}, \varepsilon) = 1] - \mathsf{negl}(\lambda).$$

Proof. We start by defining a sequence of intermediate hybrid experiments:

- $\mathsf{Hyb}_{2,0}$: Same as Hyb_2. In particular, the components $(\mathsf{crs}_{\mathsf{Com}}^{(i)}, \mathsf{td}_{\mathsf{Com}}^{(i)}, c_i, \sigma_i)_{i\in[N]}$ in the setup phase are sampled as
$$\mathsf{crs}_{\mathsf{Com}}^{(i)} \leftarrow \mathsf{Com.Setup}(1^\lambda, \mathsf{bind}), (c_i, \sigma_i) \leftarrow \mathsf{Com.Commit}(\mathsf{crs}_{\mathsf{Com}}^{(i)}, r_i).$$

- $\mathsf{Hyb}_{2,i}$: Same as $\mathsf{Hyb}_{2,0}$ except for all $j \leq i$ the challenger samples the components $(\mathsf{crs}_{\mathsf{Com}}^{(j)}, \mathsf{td}_{\mathsf{Com}}^{(j)}, c_j, \sigma_j)$ as $(\mathsf{crs}_{\mathsf{Com}}^{(j)}, \mathsf{td}_{\mathsf{Com}}^{(j)}, c_j) \leftarrow \mathsf{Com.Setup}(1^\lambda, \mathsf{hide})$ and $\sigma_j \leftarrow \mathcal{S}_{\mathsf{open}}(\mathsf{td}_{\mathsf{Com}}^{(j)}, r_j)$. The commitments and openings for $j > i$ are sampled as in $\mathsf{Hyb}_{2,0}$. Note that $\mathsf{Hyb}_{2,N}$ is the same as Hyb_3.

We now appeal to equivocation of Π_{Com} to show that for all $i \in [N]$, the statistical distance between $\mathsf{Hyb}_{2,i-1}$ and $\mathsf{Hyb}_{2,i}$ is negligible.

Claim 4.11. *Suppose Π_{Com} satisfies mode indistinguishability. Then, for all $i \in [N]$, admissible adversaries $\mathcal{A}$, and inverse polynomial functions $\varepsilon = 1/\mathrm{poly}(\lambda)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,*

$$\Pr[\mathsf{Hyb}_{2,i}(\mathcal{A}, \varepsilon) = 1] \geq \Pr[\mathsf{Hyb}_{2,i-1}(\mathcal{A}, \varepsilon) = 1] - \mathsf{negl}(\lambda).$$

Proof. Suppose there exists an adversary $\mathcal{A}$ that distinguishes $\mathsf{Hyb}_{2,i-1}$ and $\mathsf{Hyb}_{2,i}$ with non-negligible probability δ. We use $\mathcal{A}$ to construct an algorithm $\mathcal{B}$ that breaks mode indistinguishability:

1. At the beginning of the game, algorithm $\mathcal{B}$ gets $\mathsf{crs}^{(i)}_{\mathsf{Com}}$ from the mode indistinguishability challenger. Algorithm $\mathcal{B}$ samples $h \xleftarrow{\mathrm{R}} \mathcal{H}_{\mathsf{fam}}$, $(\mathsf{pp}_{\mathsf{IBE}}, \mathsf{msk}_{\mathsf{IBE}}) \leftarrow \mathsf{IBE.Setup}(1^\lambda)$, and $(\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{st}_{\mathcal{S}}) \leftarrow \mathcal{S}_1(1^\lambda)$. For all $j \in [N]$, algorithm $\mathcal{B}$ samples $r_j \xleftarrow{\mathrm{R}} \{0,1\}^{\ell_x}$. For $j < i$, algorithm $\mathcal{B}$ computes

$$\left(\mathsf{crs}^{(j)}_{\mathsf{Com}}, \mathsf{td}^{(j)}_{\mathsf{Com}}, c_j\right) \leftarrow \mathsf{Com.Setup}(1^\lambda, \mathsf{hide}), \sigma_j \leftarrow \mathcal{S}_{\mathsf{open}}(\mathsf{td}^{(j)}_{\mathsf{Com}}, r_j).$$

   Algorithm $\mathcal{B}$ submits r_i to the mode indistinguishability challenger to get (c_i, σ_i). For $j > i$, algorithm $\mathcal{B}$ computes $\mathsf{crs}^{(j)}_{\mathsf{Com}} \leftarrow \mathsf{Com.Setup}(1^\lambda, \mathsf{bind})$, $(c_j, \sigma_j) \leftarrow \mathsf{Com.Commit}(\mathsf{crs}^{(j)}_{\mathsf{Com}}, r_j)$. Algorithm $\mathcal{B}$ sets

$$\mathsf{pp} = \left(\left\{\mathsf{crs}^{(i)}_{\mathsf{Com}}\right\}_{i \in [N]}, \vec{c}, \mathsf{crs}_{\mathsf{NIZK}}, \mathsf{pp}_{\mathsf{IBE}}\right) \quad \text{and} \quad \mathsf{msk} = (\mathsf{pp}, \vec{r}, \vec{\sigma})$$

   and gives pp to $\mathcal{A}$. The components $\vec{c}, \vec{r}$, and $\vec{\sigma}$ are derived as in Eq. (4.1).

2. Algorithm $\mathcal{B}$ runs the remainder of the experiment as in Hyb_2 with algorithm $\mathcal{A}$:

   (a) When algorithm $\mathcal{A}$ makes a key-generation query on $\mathsf{id} \in \mathcal{ID}_\lambda$, algorithm $\mathcal{B}$ computes

$$\mathsf{ct} \leftarrow \mathsf{IBE.Encrypt}(\mathsf{pp}_{\mathsf{IBE}}, \mathsf{id}, \vec{r}_I; \rho_{\mathsf{enc}}) \text{ and } \pi \leftarrow \mathcal{S}_2(\mathsf{st}_{\mathcal{S}}, C_{\mathcal{R}}[\mathsf{pp}_{\mathsf{IBE}}], (\mathsf{crs}^{(I)}_{\mathsf{Com}}, \vec{c}_I, \mathsf{ct}, \mathsf{id})),$$

   where ρ_{enc} is (fresh) encryption randomness and $I \leftarrow \mathcal{H}(\mathsf{id})$. Algorithm $\mathcal{B}$ gives (ct, π) to algorithm $\mathcal{A}$.

   (b) When algorithm $\mathcal{A}$ outputs an efficiently-computable leakage function f, algorithm $\mathcal{B}$ replies with $\mathsf{leak} = f(\mathsf{msk})$. In the challenge phase, algorithm $\mathcal{A}$ outputs $\mathcal{J} \subseteq \mathcal{ID}_\lambda$ of size at least k, messages m_0, m_1, and a state st.

   (c) For all $\mathsf{id} \in \mathcal{J}$, the algorithm $\mathcal{B}$ constructs the challenge ciphertext $\widetilde{C}$ in the procedure $\mathsf{AdvCheck}(\mathsf{id})$ as

$$\widetilde{C} \leftarrow i\mathcal{O}(\mathsf{Check\text{-}CT}[\mathsf{crs}_{\mathsf{NIZK}}, C_{\mathcal{R}}[\mathsf{pp}_{\mathsf{IBE}}], \mathsf{crs}^{(I)}_{\mathsf{Com}}, \vec{c}_I, m_\beta, \mathsf{sk}_{\mathsf{id}}, \mathsf{id}, h, t])$$

   where $\beta \xleftarrow{\mathrm{R}} \{0,1\}, I \leftarrow \mathcal{H}(\mathsf{id}), \mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{IBE.KeyGen}(\mathsf{msk}, \mathsf{id}), u \leftarrow h(\vec{r}_I), t \leftarrow \mathsf{PRG}(u)$. Algorithm $\mathcal{B}$ outputs the output of the experiment.

If components are in hiding mode and simulated, algorithm $\mathcal{B}$ simulates the $\mathsf{Hyb}_{2,i-1}(\mathcal{A}, \varepsilon)$. If components are in binding mode and computed normally, algorithm $\mathcal{B}$ simulates the $\mathsf{Hyb}_{2,i}(\mathcal{A}, \varepsilon)$. Thus, algorithm $\mathcal{B}$ breaks mode indistinguishability with advantage δ. □

The lemma now follows from Claim 4.11 and a standard hybrid argument. □

Lemma 4.12. *Suppose* Π_{IBE} *satisfies adaptive semantic security. Then, for all efficient and admissible adversaries* $\mathcal{A}$ *and inverse polynomial functions* $\varepsilon = 1/\mathsf{poly}(\lambda)$*, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$*,*

$$\Pr[\mathsf{Hyb}_4(\mathcal{A}, \varepsilon) = 1] \geq \Pr[\mathsf{Hyb}_3(\mathcal{A}, \varepsilon) = 1] - \mathsf{negl}(\lambda).$$

Proof. Suppose $\mathcal{A}$ distinguishes the hybrids and makes at most Q total key queries in an experiment. We start by defining a sequence of intermediate hybrid experiments:

- $\mathsf{Hyb}_{3,0}$: Same as Hyb_3. In particular, the ct component for all key-generation queries is computed as $\mathsf{ct} \leftarrow \mathsf{IBE.Encrypt}(\mathsf{pp}_{\mathsf{IBE}}, \mathsf{id}, r_I; \rho_{\mathsf{enc}})$, where $I \leftarrow \mathcal{H}(\mathsf{id})$.

- $\mathsf{Hyb}_{3,i}$: Same as $\mathsf{Hyb}_{3,0}$ except for all $j \leq i$ the ct component for the j^{th} key-generation query is computed as $\mathsf{ct} \leftarrow \mathsf{IBE.Encrypt}(\mathsf{pp}_{\mathsf{IBE}}, \mathsf{id}, 0^{d\ell_x}; \rho_{\mathsf{enc}})$, where $I \leftarrow \mathcal{H}(\mathsf{id})$. For queries $j > i$, the ct component is computed as in $\mathsf{Hyb}_{3,0}$. Note that $\mathsf{Hyb}_{3,Q}$ is the same as Hyb_4.

We now appeal to adaptive semantic security of Π_{IBE} to show that for all $i \in [Q]$, $\mathsf{Hyb}_{3,i-1}$ and $\mathsf{Hyb}_{3,i}$ are computationally indistinguishable.

Claim 4.13. *Suppose* Π_{IBE} *satisfies adaptive semantic security. Then, for all* $i \in [Q]$*, efficient and admissible adversaries* $\mathcal{A}$*, and inverse polynomial functions* $\varepsilon = 1/\mathsf{poly}(\lambda)$*, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$*,*

$$\Pr[\mathsf{Hyb}_{3,i}(\mathcal{A}, \varepsilon) = 1] \geq \Pr[\mathsf{Hyb}_{3,i-1}(\mathcal{A}, \varepsilon) = 1] - \mathsf{negl}(\lambda).$$

Proof. Suppose there exists an efficient adversary $\mathcal{A}$ that distinguishes $\mathsf{Hyb}_{3,i-1}$ and $\mathsf{Hyb}_{3,i}$ with non-negligible advantage δ. We use $\mathcal{A}$ to construct an algorithm $\mathcal{B}$ that breaks adaptive semantic security of Π_{IBE}:

1. At the beginning of the game, $\mathcal{B}$ gets $\mathsf{pp}_{\mathsf{IBE}}$ from the plain IBE challenger. Algorithm $\mathcal{B}$ samples $h \xleftarrow{\text{R}} \mathcal{H}_{\mathsf{fam}}$ and $(\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{st}_{\mathcal{S}}) \leftarrow \mathcal{S}_1(1^\lambda)$. For all $i \in [N]$, algorithm $\mathcal{B}$ samples a random string $r_i \xleftarrow{\text{R}} \{0,1\}^{\ell_x}$, a tuple $\left(\mathsf{crs}_{\mathsf{Com}}^{(i)}, \mathsf{td}_{\mathsf{Com}}^{(i)}, c_i\right) \leftarrow \mathsf{Com.Setup}(1^\lambda, \mathsf{hide})$, and computes $\sigma_i \leftarrow \mathcal{S}_{\mathsf{open}}(\mathsf{td}_{\mathsf{Com}}^{(i)}, r_i)$. Algorithm $\mathcal{B}$ sets
$$\mathsf{pp} = \left(\left\{\mathsf{crs}_{\mathsf{Com}}^{(i)}\right\}_{i \in [N]}, \vec{c}, \mathsf{crs}_{\mathsf{NIZK}}, \mathsf{pp}_{\mathsf{IBE}}\right) \quad \text{and} \quad \mathsf{msk} = (\mathsf{pp}, \vec{r}, \vec{\sigma})$$
and gives pp to $\mathcal{A}$. The components $\vec{c}, \vec{r}$, and $\vec{\sigma}$ are derived as in Eq. (4.1).

2. On the j^{th} key-generation query to id_j when $j \neq i$, algorithm $\mathcal{B}$ computes the ciphertext component as $\mathsf{ct} \leftarrow \mathsf{IBE.Encrypt}(\mathsf{pp}_{\mathsf{IBE}}, \mathsf{id}_j, 0^{d\ell_x}; \rho_{\mathsf{enc}})$ when $j < i$ and $\mathsf{ct} \leftarrow \mathsf{IBE.Encrypt}(\mathsf{pp}_{\mathsf{IBE}}, \mathsf{id}_j, \vec{r}_I; \rho_{\mathsf{enc}})$ when $j > i$. On the i^{th} key-generation query, algorithm $\mathcal{B}$ gives $(\mathsf{id}^* = \mathsf{id}_i, m_0 = \vec{r}_I, m_1 = 0^{d\ell_x})$ to the plain IBE challenger and gets back ct^* which it uses as the ciphertext component of the response to the query. Algorithm $\mathcal{B}$ computes all proof components as $\pi \leftarrow \mathcal{S}_2(\mathsf{st}_{\mathcal{S}}, C_{\mathcal{R}}[\mathsf{pp}_{\mathsf{IBE}}], (\mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, \mathsf{ct}, \mathsf{id}))$, and answers all queries by giving (ct, π) to $\mathcal{A}$.

3. When algorithm $\mathcal{A}$ outputs an efficiently-computable leakage function f, algorithm $\mathcal{B}$ replies with $\mathsf{leak} = f(\mathsf{msk})$. In the challenge phase, algorithm $\mathcal{A}$ outputs $\mathcal{J} \subseteq \mathcal{ID}_\lambda$ of size at least k, messages m_0, m_1, and a state st.

4. For all $\mathsf{id} \in \mathcal{J}$, the algorithm $\mathcal{B}$ constructs the challenge ciphertext $\widetilde{C}$ in the procedure $\mathsf{AdvCheck}(\mathsf{id})$ as
$$\widetilde{C} \leftarrow i\mathcal{O}(\mathsf{Check\text{-}CT}[\mathsf{crs}_{\mathsf{NIZK}}, C_{\mathcal{R}}[\mathsf{pp}_{\mathsf{IBE}}], \mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, m_\beta, \mathsf{sk}_{\mathsf{id}}, \mathsf{id}, h, t])$$
where $\beta \xleftarrow{\text{R}} \{0,1\}, I \leftarrow \mathcal{H}(\mathsf{id}), u \leftarrow h(\vec{r}_I), t \leftarrow \mathsf{PRG}(u)$, and $\mathsf{sk}_{\mathsf{id}}$ is the response to a key-generation query to the plain IBE challenger on id. Algorithm $\mathcal{B}$ outputs the output of the experiment.

Note that algorithm $\mathcal{B}$ is an admissible IBE adversary if algorithm $\mathcal{A}$ is admissible, since the challenge set $\mathcal{J}$ is disjoint from the set of identities queried for key-generation. If $\mathsf{ct}^* \leftarrow \mathsf{IBE.Encrypt}(\mathsf{pp}_{\mathsf{IBE}}, \mathsf{id}_i, \vec{r}_I; \rho_{\mathsf{enc}})$, algorithm $\mathcal{B}$ simulates the challenger for $\mathsf{Hyb}_{3,i-1}$. If $\mathsf{ct}^* \leftarrow \mathsf{IBE.Encrypt}(\mathsf{pp}_{\mathsf{IBE}}, \mathsf{id}_i, 0^{d\ell_x}; \rho_{\mathsf{enc}})$, algorithm $\mathcal{B}$ simulates the challenger for $\mathsf{Hyb}_{3,i}$. Thus, algorithm $\mathcal{B}$ breaks adaptive semantic security with advantage δ. □

The lemma now follows from Claim 4.13 and a standard hybrid argument. □

Lemma 4.14. *Suppose the extractor* Ext *is a* $(\lambda + \omega(\log \lambda), \mathsf{negl}(\lambda))$*-strong extractor and the hash function* $\mathcal{H}$ *is* (k, α)*-expanding, where* $(1 - \eta)\alpha\ell_x \geq \lambda + \omega(\log \lambda)$ *for some constant* $\eta \in (0, 1)$*. Suppose also that the challenge parameter* k *satisfies* $k \geq \frac{\ell}{\eta\alpha\ell_x}$*. Then, for all admissible adversaries* $\mathcal{A}$ *and inverse polynomial functions* $\varepsilon = 1/\mathsf{poly}(\lambda)$*, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$,

$$\Pr[\mathsf{Hyb}_5(\mathcal{A}, \varepsilon) = 1] \geq \Pr[\mathsf{Hyb}_4(\mathcal{A}, \varepsilon) = 1] - \mathsf{negl}(\lambda).$$

Proof. Note that the new condition for outputting 1 can only increase the probability that 1 is output. Other than this condition, the only difference between $\mathsf{Hyb}_4, \mathsf{Hyb}_5$ is the distribution of the challenge ciphertexts in executions of AdvCheck. In particular, in for both hybrids, the challenger proceeds as follows:

- In setup, the challenger samples $h \xleftarrow{R} \mathcal{H}_{\mathsf{fam}}$, $(\mathsf{pp}_{\mathsf{IBE}}, \mathsf{msk}_{\mathsf{IBE}}) \leftarrow \mathsf{IBE.Setup}(1^\lambda)$, and $(\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{st}_{\mathcal{S}}) \leftarrow \mathcal{S}_1(1^\lambda)$. For $i \in [N]$, algorithm $\mathcal{B}$ samples $r_i \xleftarrow{R} \{0,1\}^{\ell_x}$, $\left(\mathsf{crs}_{\mathsf{Com}}^{(i)}, \mathsf{td}_{\mathsf{Com}}^{(i)}, c_i\right) \leftarrow \mathsf{Com.Setup}(1^\lambda, \mathsf{hide})$, and computes $\sigma_i \leftarrow \mathcal{S}_{\mathsf{open}}(\mathsf{td}_{\mathsf{Com}}^{(i)}, r_i)$. Algorithm $\mathcal{B}$ sets

$$\mathsf{pp} = \left(\left\{\mathsf{crs}_{\mathsf{Com}}^{(i)}\right\}_{i \in [N]}, \vec{c}, \mathsf{crs}_{\mathsf{NIZK}}, \mathsf{pp}_{\mathsf{IBE}}\right) \quad \text{and} \quad \mathsf{msk} = (\mathsf{pp}, \vec{r}, \vec{\sigma})$$

  and gives pp to $\mathcal{A}$. The components $\vec{c}, \vec{r}$, and $\vec{\sigma}$ are derived as in Eq. (4.1).

- When algorithm $\mathcal{A}$ makes a key-generation query on $\mathsf{id} \in \mathcal{ID}_\lambda$, the challenger computes

$$\mathsf{ct} \leftarrow \mathsf{IBE.Encrypt}(\mathsf{pp}_{\mathsf{IBE}}, \mathsf{id}, 0^{d\ell_x}; \rho_{\mathsf{enc}}) \text{ and } \pi \leftarrow \mathcal{S}_2(\mathsf{st}_{\mathcal{S}}, C_{\mathcal{R}}[\mathsf{pp}_{\mathsf{IBE}}], (\mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, \mathsf{ct}, \mathsf{id})),$$

  where ρ_{enc} is (fresh) encryption randomness and $I \leftarrow \mathcal{H}(\mathsf{id})$. The challenger gives (ct, π) to algorithm $\mathcal{A}$.

For $\mathsf{id} \in \mathcal{J}$, the challenger constructs the challenge ciphertext in AdvCheck(id) as

$$\mathsf{ct} = \widetilde{C} \leftarrow i\mathcal{O}(\mathsf{Check\text{-}CT}[\mathsf{crs}_{\mathsf{NIZK}}, C_{\mathcal{R}}[\mathsf{pp}_{\mathsf{IBE}}], \mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, m_\beta, \mathsf{sk}_{\mathsf{id}}, \mathsf{id}, h, t]),$$

where $\beta \xleftarrow{R} \{0,1\}$, $I \leftarrow \mathcal{H}(\mathsf{id})$, $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{IBE.KeyGen}(\mathsf{msk}, \mathsf{id})$, and $t \leftarrow \mathsf{PRG}(u)$. The distribution of the u component differs between the two hybrids:

- In Hyb_4, the challenger samples $u \leftarrow h(\vec{r}_I) = \mathsf{Ext}(\vec{r}_I, h)$.
- In Hyb_5, the challenger samples a single value $u \leftarrow \{0,1\}^\lambda$ which is reused across all of the AdvCheck(id) iterations.

We now define the following events

- Let $E_{\mathsf{Ext}}^{\mathsf{id}}$ be the event that $\mathsf{AdvCheck}(\mathsf{id}) = 1$ for $u = h(\vec{r}_I)$.

- Let E^{id} be the event that $\mathsf{AdvCheck}(\mathsf{id}) = 1$ for $u \xleftarrow{\mathrm{R}} \{0,1\}^\lambda$.

We will appeal to security of the extractor to show that there exists $\mathsf{id} \in \mathcal{J}$ such that

$$\Pr[E^{\mathsf{id}}] \geq \Pr[E^{\mathsf{id}}_{\mathsf{Ext}}] - \mathsf{negl}(\lambda).$$

To do this, we lower bound the entropy of some or all of the bits $\{\vec{r}_{\mathcal{H}(\mathsf{id})}\}_{\mathsf{id} \in \mathcal{J}}$ conditioned on the tuple $(\mathsf{pp}, Q, \mathsf{leak}, \mathcal{J})$, where Q denotes the list of outputs of key-generation queries. Fix any k identities $\mathsf{id}_1, \ldots, \mathsf{id}_k$ which belong to $\mathcal{J}$. By assumption, $\mathcal{H}(\mathsf{id}_1), \ldots, \mathcal{H}(\mathsf{id}_k)$ contains at least αk distinct indices of $[N]$, which correspond to $\alpha k \cdot \ell_x$ total bits of msk. Since r is independent of (pp, Q) in Hyb_4 and $\eta < 1$, we can appeal to Lemma 2.2 to get the following:

$$\begin{aligned}
\mathbf{H}_\infty(\{\vec{r}_{\mathcal{H}(\mathsf{id}_i)}\}_{i \in [k]} \mid \mathsf{pp}, Q, \mathsf{leak}) &\geq \mathbf{H}_\infty(\{\vec{r}_{\mathcal{H}(\mathsf{id}_i)}\}_{i \in [k]} \mid \mathsf{pp}, Q) - |\mathsf{leak}| \\
&\geq \mathbf{H}_\infty(\{\vec{r}_{\mathcal{H}(\mathsf{id}_i)}\}_{i \in [k]}) - |\mathsf{leak}| \\
&= \alpha k \ell_x - \eta \alpha k \ell_x \\
&= (1 - \eta)\alpha k \ell_x.
\end{aligned}$$

By Lemma 2.2, with probability $1 - 2^{-\omega(\log \lambda)} = 1 - \mathsf{negl}(\lambda)$ over the fixed choice of $(\mathsf{pp}, Q, \mathsf{leak})$, we have

$$\mathbf{H}_\infty(\{\vec{r}_{\mathcal{H}(\mathsf{id}_i)}\}_{i \in [k]}) \geq (1 - \eta)\alpha k \ell_x - \omega(\log \lambda).$$

Moreover, by Lemma 2.3, there exists a random variable $\mathcal{D}_{[k]}$ over $[k]$ such that

$$\mathbf{H}_\infty(\vec{r}_J \mid \mathcal{D}_{[k]}) \geq \frac{(1 - \eta)\alpha k \ell_x - \omega(\log \lambda)}{k} - \log(k),$$

where $J = \mathcal{H}(\mathsf{id}_{\mathcal{D}_{[k]}})$. By Lemma 2.2, we have with probability $1 - 2^{\omega(\log \lambda)} = 1 - \mathsf{negl}(\lambda)$ over the choice of $i \leftarrow \mathcal{D}_{[k]}$,

$$\mathbf{H}_\infty(\vec{r}_{\mathcal{H}(\mathsf{id}_i)} \mid \mathcal{D}_{[k]} = i) \geq \mathbf{H}_\infty(\vec{r}_J \mid \mathcal{D}_{[k]}) - \omega(\log \lambda)$$

Define $\mathsf{id}^* = \mathsf{id}_i$ where $i \leftarrow \mathcal{D}_{[k]}$. With overwhelming probability over the choice of id^* (alternatively, over the choice of i), we have

$$\mathbf{H}_\infty(\vec{r}_{\mathcal{H}(\mathsf{id}^*)}) = (1 - \eta)\alpha \ell_x - \omega(\log \lambda)$$

for fixed $(\mathsf{pp}, Q, \mathcal{J}, \mathsf{leak})$. Here, we have used the fact that $k = \mathsf{poly}(\lambda)$, so $\log k = O(\log \lambda)$. Since $(1 - \eta)\alpha \ell_x \geq \lambda + \omega(\log \lambda)$ by assumption, we can appeal to extractor security with overwhelming probability in the $(\mathsf{id}^*)^{\text{th}}$ copy of the game. This means $h(\vec{r}_{\mathcal{H}(\mathsf{id}^*)})$ is statistically close to uniform with negligible statistical distance, so we must have

$$\Pr[E^{\mathsf{id}^*}] \geq \Pr[E^{\mathsf{id}^*}_{\mathsf{Ext}}] - \mathsf{negl}(\lambda).$$

By definition, $\Pr[E^{\mathsf{id}^*}_{\mathsf{Ext}}] \geq \Pr[\mathsf{Hyb}_4(\mathcal{A}, \varepsilon) = 1]$ and $\Pr[\mathsf{Hyb}_5(\mathcal{A}, \varepsilon) = 1] \geq \Pr[E^{\mathsf{id}^*}]$, so the lemma follows. $\square$

Lemma 4.15. *Suppose* PRG *satisfies PRG security. Then, for all efficient and admissible adversaries $\mathcal{A}$ and inverse polynomial functions $\varepsilon = 1/\mathsf{poly}(\lambda)$, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all $\lambda \in \mathbb{N}$,*

$$\Pr[\mathsf{Hyb}_6(\mathcal{A}, \varepsilon) = 1] \geq \Pr[\mathsf{Hyb}_5(\mathcal{A}, \varepsilon) = 1] - \mathsf{negl}(\lambda).$$

Proof. We define a sequence of intermediate hybrids:

- $\mathsf{Hyb}_{5,0}$: Same as Hyb_5. Notably, for each $\mathsf{id} \in \mathcal{J}$, the t component of the challenge ciphertext is sampled as $\mathsf{PRG}(u)$ where $u \xleftarrow{\mathrm{R}} \{0,1\}^\lambda$ at the start of $\mathsf{AdvCheck}(\mathsf{id})$ and fixed for all ciphertexts.
- $\mathsf{Hyb}_{5,i}$: Same as Hyb_5 except for $i' \le i$, we sample t as $t \leftarrow \{0,1\}^{2\lambda}$ at the start of $\mathsf{AdvCheck}$ on $\mathsf{id} = \mathcal{J}[i']$. Note that $\mathsf{Hyb}_{5,k}$ is the same as Hyb_6.

We now appeal to PRG security to show that for all $i \in [k]$ we show that $\mathsf{Hyb}_{5,i}$ and $\mathsf{Hyb}_{5,i-1}$ are computationally indistinguishable.

Claim 4.16. *Suppose* PRG *satisfies PRG security. Then for all $i \in [k]$, all efficient and admissible adversaries $\mathcal{A}$ and inverse polynomial functions $\varepsilon = 1/\mathsf{poly}(\lambda)$, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all $\lambda \in \mathbb{N}$,*

$$\Pr[\mathsf{Hyb}_{5,i}(\mathcal{A},\varepsilon) = 1] \ge \Pr[\mathsf{Hyb}_{5,i-1}(\mathcal{A},\varepsilon) = 1] - \mathsf{negl}(\lambda).$$

Proof. Suppose there exists an efficient adversary $\mathcal{A}$ that distinguishes $\mathsf{Hyb}_{5,i}$ and $\mathsf{Hyb}_{5,i-1}$ with non-negligible probability δ. We use $\mathcal{A}$ to construct an algorithm $\mathcal{B}$ that breaks PRG security:

1. Algorithm $\mathcal{B}$ gets a PRG challenge $t^* \in \{0,1\}^{2\lambda}$ and runs the setup through challenge phases as in Hyb_5 with $\mathcal{A}$:

   (a) Algorithm $\mathcal{B}$ samples $h \xleftarrow{\mathrm{R}} \mathcal{H}_{\mathsf{fam}}$, $(\mathsf{pp}_{\mathsf{IBE}}, \mathsf{msk}_{\mathsf{IBE}}) \leftarrow \mathsf{IBE.Setup}(1^\lambda)$, and $(\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{st}_{\mathcal{S}}) \leftarrow \mathcal{S}_1(1^\lambda)$. For all $i \in [N]$, algorithm $\mathcal{B}$ samples a string $r_i \xleftarrow{\mathrm{R}} \{0,1\}^{\ell_x}$, a tuple $\big(\mathsf{crs}_{\mathsf{Com}}^{(i)}, \mathsf{td}_{\mathsf{Com}}^{(i)}, c_i\big) \leftarrow \mathsf{Com.Setup}(1^\lambda, \mathsf{hide})$, and computes $\sigma_i \leftarrow \mathcal{S}_{\mathsf{open}}(\mathsf{td}_{\mathsf{Com}}^{(i)}, r_i)$. Algorithm $\mathcal{B}$ sets

   $$\mathsf{pp} = \left(\left\{\mathsf{crs}_{\mathsf{Com}}^{(i)}\right\}_{i\in[N]}, \vec{c}, \mathsf{crs}_{\mathsf{NIZK}}, \mathsf{pp}_{\mathsf{IBE}}\right) \quad \text{and} \quad \mathsf{msk} = (\mathsf{pp}, \vec{r}, \vec{\sigma})$$

   and gives pp to $\mathcal{A}$. The components $\vec{c}, \vec{r}$, and $\vec{\sigma}$ are derived as in Eq. (4.1).

   (b) When algorithm $\mathcal{A}$ makes a key-generation query on $\mathsf{id} \in \mathcal{ID}_\lambda$, the challenger computes

   $$\mathsf{ct} \leftarrow \mathsf{IBE.Encrypt}(\mathsf{pp}_{\mathsf{IBE}}, \mathsf{id}, 0^{d\ell_x}; \rho_{\mathsf{enc}}) \text{ and } \pi \leftarrow \mathcal{S}_2(\mathsf{st}_{\mathcal{S}}, C_{\mathcal{R}}[\mathsf{pp}_{\mathsf{IBE}}], (\mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, \mathsf{ct}, \mathsf{id})),$$

   where ρ_{enc} is (fresh) encryption randomness and $I \leftarrow \mathcal{H}(\mathsf{id})$. Algorithm $\mathcal{B}$ gives (ct, π) to algorithm $\mathcal{A}$.

   (c) When algorithm $\mathcal{A}$ outputs an efficiently-computable leakage function f, algorithm $\mathcal{B}$ replies with $\mathsf{leak} = f(\mathsf{msk})$. In the challenge phase, algorithm $\mathcal{A}$ outputs $\mathcal{J} \subseteq \mathcal{ID}_\lambda$ of size at least k, messages m_0, m_1, and a state st.

2. For $\mathsf{id} \in \mathcal{J}$, the challenger constructs the challenge ciphertext in $\mathsf{AdvCheck}(\mathsf{id})$ as

   $$\mathsf{ct} = \widetilde{C} \leftarrow i\mathcal{O}(\mathsf{Check\text{-}CT}[\mathsf{crs}_{\mathsf{NIZK}}, C_{\mathcal{R}}[\mathsf{pp}_{\mathsf{IBE}}], \mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, m_\beta, \mathsf{sk}_{\mathsf{id}}, \mathsf{id}, h, t]),$$

   where $\beta \xleftarrow{\mathrm{R}} \{0,1\}, I \leftarrow \mathcal{H}(\mathsf{id}), \mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{IBE.KeyGen}(\mathsf{msk}, \mathsf{id})$. For all i' such that $i' < i$, algorithm $\mathcal{B}$ samples t at the start of $\mathsf{AdvCheck}(\mathcal{J}[i'])$ as $t \xleftarrow{\mathrm{R}} \{0,1\}^{2\lambda}$. For $\mathsf{AdvCheck}(\mathcal{J}[i])$, algorithm $\mathcal{B}$ uses its PRG challenge t^* as the t component. Algorithm $\mathcal{B}$ uses $t \leftarrow \mathsf{PRG}(u)$ where $u \xleftarrow{\mathrm{R}} \{0,1\}^\lambda$ for the t components at the start of $\mathsf{AdvCheck}(\mathsf{id})$ for the remaining $\mathsf{id} \in \mathcal{J}$. Algorithm $\mathcal{B}$ outputs whatever the experiment outputs.

If $t = \mathsf{PRG}(U_\lambda)$, algorithm $\mathcal{B}$ simulates $\mathsf{Hyb}_{5,i-1}(\mathcal{A},\varepsilon)$. If $t \xleftarrow{\mathrm{R}} \{0,1\}^{2\lambda}$, algorithm $\mathcal{B}$ simulates $\mathsf{Hyb}_{5,i}(\mathcal{A},\varepsilon)$. Thus, algorithm $\mathcal{B}$ breaks PRG security with advantage δ. □

The lemma now follows from Claim 4.16 and a standard hybrid argument. □

Lemma 4.17. *Suppose $i\mathcal{O}$ satisfies indistinguishability obfuscation security. Then, for all efficient and admissible adversaries $\mathcal{A}$ and inverse polynomial functions $\varepsilon = 1/\mathsf{poly}(\lambda)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,*

$$\Pr[\mathsf{Hyb}_7(\mathcal{A}, \varepsilon) = 1] \geq \Pr[\mathsf{Hyb}_6(\mathcal{A}, \varepsilon) = 1] - \mathsf{negl}(\lambda).$$

Proof. We define a sequence of intermediate hybrids:

- $\mathsf{Hyb}_{6,1,0}$: Same as Hyb_6. Notably, for all $\mathsf{id} \in \mathcal{J}$ the challenge ciphertext is sampled as

$$\widetilde{C} \leftarrow i\mathcal{O}(\mathsf{Check\text{-}CT}[\mathsf{crs}_{\mathsf{NIZK}}, C_{\mathcal{R}}[\mathsf{pp}_{\mathsf{IBE}}], \mathsf{crs}^{(I)}_{\mathsf{Com}}, \vec{c}_I, m_\beta, \mathsf{sk}_{\mathsf{id}}, \mathsf{id}, h, t]),$$

  where $\beta \xleftarrow{\mathrm{R}} \{0,1\}, I \leftarrow \mathcal{H}(\mathsf{id}), \mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{IBE.KeyGen}(\mathsf{msk}, \mathsf{id})$.

- $\mathsf{Hyb}_{6,i,j}$: Same as Hyb_6 except for all (i', j') such that $i' < i$ or $i' = i, j' \leq j$, the challenge ciphertext in the j'th execution of $\mathsf{Exp}^{\mathcal{J}[i']}$ in $\mathsf{AdvCheck}(\mathcal{J}[i'])$ is sampled as $\widetilde{C} \leftarrow i\mathcal{O}(\mathsf{Bot})$. Note that $\mathsf{Hyb}_{6,k,T}$ is the same as Hyb_7 and that $\mathsf{Hyb}_{6,i,T}$ is the same as $\mathsf{Hyb}_{6,i+1,0}$ for $i \in [k-1]$.

We now appeal to $i\mathcal{O}$ security to show that for all $i \in [k], j \in [T]$ we show that $\mathsf{Hyb}_{6,i,j}$ and $\mathsf{Hyb}_{6,i,j-1}$ are computationally indistinguishable.

Claim 4.18. *Suppose the conditions in Lemma 4.17 hold. Then for all $i \in [k], j \in [T]$, all efficient and admissible adversaries $\mathcal{A}$ and inverse polynomial functions $\varepsilon = 1/\mathsf{poly}(\lambda)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,*

$$\Pr[\mathsf{Hyb}_{6,i,j}(\mathcal{A}, \varepsilon) = 1] \geq \Pr[\mathsf{Hyb}_{6,i,j-1}(\mathcal{A}, \varepsilon) = 1] - \mathsf{negl}(\lambda).$$

Proof. Suppose there exists an efficient adversary $\mathcal{A}$ that distinguishes $\mathsf{Hyb}_{6,i,j}$ and $\mathsf{Hyb}_{6,i,j-1}$ with non-negligible probability δ. We use $\mathcal{A}$ to construct an algorithm $\mathcal{B}$ that breaks $i\mathcal{O}$ security:

1. Algorithm $\mathcal{B}$ runs the setup through challenge phases as in Hyb_5 with $\mathcal{A}$:

   (a) Algorithm $\mathcal{B}$ samples $h \xleftarrow{\mathrm{R}} \mathcal{H}_{\mathsf{fam}}$, $(\mathsf{pp}_{\mathsf{IBE}}, \mathsf{msk}_{\mathsf{IBE}}) \leftarrow \mathsf{IBE.Setup}(1^\lambda)$, and $(\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{st}_{\mathcal{S}}) \leftarrow \mathcal{S}_1(1^\lambda)$. For all $i \in [N]$, algorithm $\mathcal{B}$ samples a string $r_i \xleftarrow{\mathrm{R}} \{0,1\}^{\ell_x}$, a tuple $(\mathsf{crs}^{(i)}_{\mathsf{Com}}, \mathsf{td}^{(i)}_{\mathsf{Com}}, c_i) \leftarrow \mathsf{Com.Setup}(1^\lambda, \mathsf{hide})$, and computes $\sigma_i \leftarrow \mathcal{S}_{\mathsf{open}}(\mathsf{td}^{(i)}_{\mathsf{Com}}, r_i)$. Algorithm $\mathcal{B}$ sets

$$\mathsf{pp} = \left(\{\mathsf{crs}^{(i)}_{\mathsf{Com}}\}_{i \in [N]}, \vec{c}, \mathsf{crs}_{\mathsf{NIZK}}, \mathsf{pp}_{\mathsf{IBE}}\right) \quad \text{and} \quad \mathsf{msk} = (\mathsf{pp}, \vec{r}, \vec{\sigma})$$

   and gives pp to $\mathcal{A}$. The components $\vec{c}, \vec{r}$, and $\vec{\sigma}$ are derived as in Eq. (4.1).

   (b) When algorithm $\mathcal{A}$ makes a key-generation query on $\mathsf{id} \in \mathcal{ID}_\lambda$, the challenger computes

$$\mathsf{ct} \leftarrow \mathsf{IBE.Encrypt}(\mathsf{pp}_{\mathsf{IBE}}, \mathsf{id}, 0^{d\ell_x}; \rho_{\mathsf{enc}}) \text{ and } \pi \leftarrow \mathcal{S}_2(\mathsf{st}_{\mathcal{S}}, C_{\mathcal{R}}[\mathsf{pp}_{\mathsf{IBE}}], (\mathsf{crs}^{(I)}_{\mathsf{Com}}, \vec{c}_I, \mathsf{ct}, \mathsf{id})),$$

   where ρ_{enc} is (fresh) encryption randomness and $I \leftarrow \mathcal{H}(\mathsf{id})$. Algorithm $\mathcal{B}$ gives (ct, π) to algorithm $\mathcal{A}$.

   (c) When algorithm $\mathcal{A}$ outputs an efficiently-computable leakage function f, algorithm $\mathcal{B}$ replies with $\mathsf{leak} = f(\mathsf{msk})$. In the challenge phase, algorithm $\mathcal{A}$ outputs $\mathcal{J} \subseteq \mathcal{ID}_\lambda$ of size at least k, messages m_0, m_1, and a state st.

2. For all (i', j') such that $i' < i$ or $i' = i, j' < j$, algorithm $\mathcal{B}$ samples the challenge ciphertext in the j'^{th} execution of $\mathsf{Exp}^{\mathcal{J}[i']}$ in $\mathsf{AdvCheck}$ as $\widetilde{C} \leftarrow i\mathcal{O}(\mathsf{Bot})$.

3. For the j^{th} execution of $\mathsf{Exp}^{\mathcal{J}[i]}$, algorithm $\mathcal{B}$ samples components $\beta \xleftarrow{\text{R}} \{0,1\}, t \xleftarrow{\text{R}} \{0,1\}^{2\lambda}, I \leftarrow \mathcal{H}(\mathcal{J}[i]), \mathsf{sk}_{\mathcal{J}[i]} \leftarrow \mathsf{IBE.KeyGen}(\mathsf{msk}, \mathcal{J}[i])$ and sets
$$C_0 = \mathsf{Check\text{-}CT}[\mathsf{crs}_{\mathsf{NIZK}}, C_{\mathcal{R}}[\mathsf{pp}_{\mathsf{IBE}}], \mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, m_\beta, \mathsf{sk}_{\mathcal{J}[i]}, \mathcal{J}[i], h, t] \text{ and } C_1 = \mathsf{Bot}.$$
Algorithm $\mathcal{B}$ submits (C_0, C_1) to the $i\mathcal{O}$ challenger and gets back program P, which is used as the challenge ciphertext in this execution.

4. In the remaining executions of $\mathsf{Exp}^{\mathcal{J}[i']}$, algorithm $\mathcal{B}$ samples $\beta \xleftarrow{\text{R}} \{0,1\}, t \xleftarrow{\text{R}} \{0,1\}^{2\lambda}, I \leftarrow \mathcal{H}(\mathcal{J}[i']), \mathsf{sk}_{\mathcal{J}[i']} \leftarrow \mathsf{IBE.KeyGen}(\mathsf{msk}, \mathcal{J}[i'])$ and computes the challenge ciphertext as
$$\widetilde{C} \leftarrow i\mathcal{O}(\mathsf{Check\text{-}CT}[\mathsf{crs}_{\mathsf{NIZK}}, C_{\mathcal{R}}[\mathsf{pp}_{\mathsf{IBE}}], \mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, m_\beta, \mathsf{sk}_{\mathcal{J}[i']}, \mathcal{J}[i'], h, t]).$$
Algorithm $\mathcal{B}$ outputs the output of the experiment.

If $P \leftarrow i\mathcal{O}(C_0)$, algorithm $\mathcal{B}$ simulates $\mathsf{Hyb}_{6,i,j-1}(\mathcal{A}, \varepsilon)$. If $P \leftarrow i\mathcal{O}(C_1)$, algorithm $\mathcal{B}$ simulates $\mathsf{Hyb}_{6,i,j}(\mathcal{A}, \varepsilon)$. Moreover, (C_0, C_1) are functionally-equivalent circuits with overwhelming probability over the choice of t. Namely, the string $t \xleftarrow{\text{R}} \{0,1\}^{2\lambda}$ is contained in the image of PRG with probability at most $2^\lambda/2^{2\lambda} = 2^{-\lambda}$ probability. When t is not in the image of PRG, the program $\mathsf{Check\text{-}CT}$ outputs $\perp$ on all inputs, which coincides with the behavior of Bot. Thus, algorithm $\mathcal{B}$ breaks $i\mathcal{O}$ security with advantage that is negligibly close to δ. □

Since $\mathsf{Hyb}_{6,i,T}$ is identical to $\mathsf{Hyb}_{6,i+1,0}$ for $i \in [k-1]$, the lemma now follows from Claim 4.18 and a standard hybrid argument. □

Lemma 4.19. *For all efficient and admissible adversaries $\mathcal{A}$ and inverse polynomial functions $\varepsilon = 1/\mathsf{poly}(\lambda)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,*
$$\Pr[\mathsf{Hyb}_7(\mathcal{A}, \varepsilon) = 1] = \mathsf{negl}(\lambda).$$

Proof. Since the bit b has been erased from the challenge ciphertexts, for all $\mathsf{id} \in \mathcal{J}$,
$$\mathsf{Adv}^{\mathsf{id}}(\mathsf{msk}, \mathsf{pp}, \mathsf{st}, \mathsf{leak}) = 0. \tag{4.4}$$

For each $i \in [T]$, let $X_i \in \{0,1\}$ be the random variable for whether algorithm $\mathcal{A}_2$'s output is correct (i.e., if $\beta' = \beta$ on the i^{th} iteration). Since Eq. (4.4) holds, $\mathbb{E}[X_i] = \Pr[X_i = 1] = 1/2$. Moreover, $K = \sum_{i \in [T]} X_i$ and $\mathbb{E}[K] = T/2$. By Hoeffding's inequality (Fact 2.1),
$$\Pr[K - T/2 < \varepsilon T/2] \le \Pr[|K - T/2| > \varepsilon T/2] \le 2^{-\Omega(T\varepsilon^2/4)} = \mathsf{negl}(\lambda),$$
since $T = \lambda/\varepsilon^2$. Thus, in an execution of $\mathsf{AdvCheck}^{\mathcal{A}_2}(1^\lambda, 1^{1/\varepsilon}, \mathsf{id}, \mathsf{msk}, \mathsf{pp}, \mathsf{st}, \mathsf{leak})$, $K \ge T/2 + \varepsilon T/2$ with probability $\mathsf{negl}(\lambda)$. Since Eq. (4.4) holds for all $\mathsf{id} \in \mathcal{J}$, $\mathsf{AdvCheck}$ also outputs 1 for any $\mathsf{id} \in \mathcal{J}$ with probability at most $|\mathcal{J}| \cdot \mathsf{negl}(\lambda) = \mathsf{negl}(\lambda)$, as desired. □

Combining Lemmas 4.9, 4.10, 4.12, 4.14, 4.15, 4.17 and 4.19 yields the statement by a hybrid argument. □

Combined with Theorem 3.4, this yields the following corollary:

Corollary 4.20 (Adaptive Security under Bounded Leakage). *Suppose the conditions in Theorem 4.6 hold. Then, Construction 4.3 is adaptively secure under bounded leakage for the same k as in Theorem 4.6.*

Remark 4.21 (Leakage Rate). By the condition on k in Theorem 4.6, we have $\eta \cdot k\alpha\ell_x \geq \ell$ for $\eta \in (0,1)$. By construction, the number of bits in $\vec{r} = (r_1, \ldots, r_N)$ is $\ell_x \cdot N$. By using the hash function from Lemma 4.2, we have that $\alpha k = (1-\delta)N$ for any $\delta \in (0,1)$. Thus, $\eta \cdot (1-\delta)N\ell_x = \eta' \cdot N\ell_x \geq \ell$ for any $\eta' \in (0,1)$. Since the only private components in msk are $\vec{r}$ and $\vec{\sigma}$, the leakage rate is then dependent on the number of bits in $\vec{\sigma}$ compared to $\vec{r}$. With the Naor commitment scheme based on one-way functions [Nao89], we obtain leakage rate $1/O(\lambda)$ since an opening to a single bit is $O(\lambda)$ bits. However, by substituting an algebraic dual-mode commitment where the size of the opening is at most $2\times$ the bit-length of the underlying message (e.g., [GS08, BL20]), we can achieve leakage rate approaching $1/3$, matching [DGSW22].

5 Adaptively-Secure Big-Key IBE from Witness Encryption

In this section, we describe how to construct a big-key IBE scheme from a witness encryption scheme (Definition 2.14), a NIZK (Definition 2.17), a dual-mode commitment scheme (Definition 2.15), and two additional building blocks which we define below.

5.1 Split Encodings and Privately-Testable Encodings

As outlined in Section 1.1, the core building blocks for our second big-key IBE construction are split encodings and privately-testable encodings. The main difference between these primitives is whether the encodings can be tested publicly or privately. We formalize these notions below. Additionally, we show how to construct these primitives from group-based assumptions in Section 6.

Definition 5.1 (Split Encoding). A split encoding scheme with tag space $\mathcal{T} = \{\mathcal{T}_\lambda\}_{\lambda \in \mathbb{N}}$ is a tuple of efficient algorithms $\Pi_{\mathsf{SE}} = (\mathsf{Setup}, \mathsf{SetupSF}, \mathsf{Encode}, \mathsf{EncodeSF}, \mathsf{Test})$ with the following syntax:

- $\mathsf{Setup}(1^\lambda) \to \mathsf{crs}$: On input the security parameter λ, the setup algorithm outputs a common reference string crs.
- $\mathsf{SetupSF}(1^\lambda) \to (\mathsf{crs}, \mathsf{td})$: On input the security parameter λ, the semi-functional setup algorithm outputs a common reference string crs and a trapdoor td.
- $\mathsf{Encode}(\mathsf{crs}, \mathsf{type}) \to \mathsf{enc}$: On input the common reference string crs and type $\in \{0,1\}$, the encode algorithm outputs an encoding enc.
- $\mathsf{EncodeSF}(\mathsf{crs}, \mathsf{td}, \mathsf{tag}, \mathsf{type}) \to \mathsf{enc}$: On input the common reference string crs, a trapdoor td, tag $\in \mathcal{T}_\lambda$ and type $\in \{0,1\}$, the semi-functional encode algorithm outputs an encoding enc.
- $\mathsf{Test}(\mathsf{crs}, \mathsf{enc}_0, \mathsf{enc}_1) \to \{0,1\}$: On input the common reference string crs and a pair of encodings $(\mathsf{enc}_0, \mathsf{enc}_1)$, the testing algorithm outputs a bit $b \in \{0,1\}$.

Moreover, Π_{SE} should satisfy the following properties:

- **Tester correctness:** For all $\lambda \in \mathbb{N}$ and tag $\in \mathcal{T}_\lambda$, all $(\mathsf{crs}, \mathsf{td})$ in the support of $\mathsf{SetupSF}(1^\lambda)$, all enc_0 in the support of $\mathsf{EncodeSF}(\mathsf{crs}, \mathsf{td}, \mathsf{tag}, 0)$ and all enc_1 in the support of $\mathsf{EncodeSF}(\mathsf{crs}, \mathsf{td}, \mathsf{tag}, 1)$,
$$\Pr[\mathsf{Test}(\mathsf{crs}, \mathsf{enc}_0, \mathsf{enc}_1) = 1] = 1.$$

Additionally, there exists a negligible function $\mathsf{negl}(\cdot)$ such that

$$\Pr\left[\mathsf{Test}(\mathsf{crs}, \mathsf{enc}_0, \mathsf{enc}_1) = 1 : \begin{array}{c} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{enc}_0 \leftarrow \mathsf{Encode}(\mathsf{crs}, 0) \\ \mathsf{enc}_1 \leftarrow \mathsf{Encode}(\mathsf{crs}, 1) \end{array}\right] = \mathsf{negl}(\lambda).$$

- **Mode indistinguishability:** For a security parameter λ and a bit $b \in \{0, 1\}$, we define the mode indistinguishability game between an adversary $\mathcal{A}$ and a challenger as follows:

  1. The challenger samples $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$ if $b = 0$ and $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{SetupSF}(1^\lambda)$ if $b = 1$. The challenger gives crs to $\mathcal{A}$.
  2. Algorithm $\mathcal{A}$ can now issue encoding queries to the challenger. On each such query, adversary $\mathcal{A}$ specifies $\mathsf{tag} \in \mathcal{T}_\lambda$ and $\mathsf{type} \in \{0, 1\}$. If $b = 0$, the challenger replies with $\mathsf{Encode}(\mathsf{crs}, \mathsf{type})$. If $b = 1$, the challenger replies with $\mathsf{EncodeSF}(\mathsf{crs}, \mathsf{td}, \mathsf{tag}, \mathsf{type})$.
  3. At the end of the game, algorithm $\mathcal{A}$ outputs a bit $b' \in \{0, 1\}$, which is the output of the experiment.

  An adversary $\mathcal{A}$ is admissible for the mode indistinguishability game if it does not issue two encoding queries on the same tag with different types. We say Π_{SE} satisfies mode indistinguishability if for all efficient and admissible adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$|\Pr[b' = 1 | b = 1] - \Pr[b' = 1 | b = 0]| = \mathsf{negl}(\lambda)$$

  in the mode indistinguishability security game.

Definition 5.2 (Privately-Testable Encoding)**.** A privately testable encoding with input space $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda \in \mathbb{N}}$ is a tuple of efficient algorithms $\Pi_{\mathsf{PTE}} = (\mathsf{Setup}, \mathsf{SetupSF}, \mathsf{SampSF}, \mathsf{Encode}, \mathsf{EncodeSF}, \mathsf{Test})$ with the following syntax:

- $\mathsf{Setup}(1^\lambda) \to \mathsf{crs}$: On input the security parameter λ, the setup algorithm outputs a common reference string crs.

- $\mathsf{SetupSF}(1^\lambda) \to (\mathsf{crs}, \mathsf{td})$: On input the security parameter λ, the semi-functional setup algorithm outputs a common reference string crs and a trapdoor td.

- $\mathsf{Samp}(1^\lambda) \to u$: On input the security parameter λ, the sample algorithm outputs a string u.

- $\mathsf{SampSF}(\mathsf{td}, x) \to \mathsf{td}_x$: On input a trapdoor td and an input $x \in \mathcal{X}_\lambda$, the semi-functional sampling algorithm outputs a trapdoor td_x.

- $\mathsf{Encode}(\mathsf{crs}, x) \to \mathsf{enc}$: On input the common reference string crs and an input$x \in \mathcal{X}_\lambda$, the encode algorithm outputs an encoding enc.

- $\mathsf{EncodeSF}(\mathsf{crs}) \to \mathsf{enc}$: On input the common reference string crs, the semi-functional encode algorithm outputs an encoding enc.

- $\mathsf{Test}(\mathsf{crs}, \mathsf{enc}, s) \to \{0, 1\}$: On input the common reference string crs, an encoding enc, and a string s, the testing algorithm outputs a bit $b \in \{0, 1\}$.

Moreover, Π_{PTE} should satisfy the following properties:

- **Tester correctness:** For all $\lambda \in \mathbb{N}$, all $(\mathsf{crs}, \mathsf{td})$ in the support of $\mathsf{SetupSF}(1^\lambda)$, all inputs $x \in \mathcal{X}_\lambda$, and all encodings enc in the support of $\mathsf{Encode}(\mathsf{crs}, x)$,

$$\Pr\left[\mathsf{Test}(\mathsf{crs}, \mathsf{enc}, \mathsf{td}_x) = 1 : \ \mathsf{td}_x \leftarrow \mathsf{SampSF}(\mathsf{td}, x)\ \right] = 1.$$

In addition, there exists a negligible function $\mathsf{negl}(\cdot)$ such that

$$\Pr\left[\mathsf{Test}(\mathsf{crs}, \mathsf{enc}, u) = 1 : \begin{array}{c} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda); \\ \mathsf{enc} \leftarrow \mathsf{Encode}(\mathsf{crs}, x); \\ u \leftarrow \mathsf{Samp}(1^\lambda) \end{array}\right] = \mathsf{negl}(\lambda).$$

- **Mode indistinguishability:** For a security parameter λ and a bit $b \in \{0, 1\}$, we define the mode indistinguishability game between an adversary $\mathcal{A}$ and a challenger as follows:

  1. The challenger samples $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$ if $b = 0$ and $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{SetupSF}(1^\lambda)$ if $b = 1$. The challenger gives crs to $\mathcal{A}$.
  2. Algorithm $\mathcal{A}$ can now issue encoding queries to the challenger. On each query, the adversary $\mathcal{A}$ specifies an input $x \in \mathcal{X}_\lambda$. If $b = 0$, the challenger replies with $\mathsf{Encode}(\mathsf{crs}, x)$. If $b = 1$, the challenger replies with $\mathsf{EncodeSF}(\mathsf{crs})$.
  3. At the end of the game, algorithm $\mathcal{A}$ outputs a bit $b' \in \{0, 1\}$, which is the output of the experiment.

  We say Π_{PTE} satisfies mode indistinguishability if for all efficient adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$|\Pr[b' = 1 | b = 1] - \Pr[b' = 1 | b = 0]| = \mathsf{negl}(\lambda)$$

  in the mode indistinguishability security game.

- k-**Trapdoor indistinguishability:** Let X be a random variable taking on values in $\mathcal{X}_\lambda$. Suppose $\mathbf{H}_\infty(X) \geq k$. Then, the following distributions are statistically indistinguishable:

$$\left\{(\mathsf{crs}, \mathsf{td}_x) : \begin{array}{c} (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{SetupSF}(1^\lambda) \\ x \leftarrow X, \mathsf{td}_x \leftarrow \mathsf{SampSF}(\mathsf{td}, x) \end{array}\right\} \quad \text{and} \quad \left\{(\mathsf{crs}, u) : \begin{array}{c} (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{SetupSF}(1^\lambda) \\ u \leftarrow \mathsf{Samp}(1^\lambda) \end{array}\right\}$$

5.2 Constructing Big-Key IBE from Witness Encryption

We now describe our big-key IBE scheme based on (plain) witness encryption:

Construction 5.3 (Big-Key IBE from Witness Encryption)**.** Let $\lambda \in \mathbb{N}$ be a security parameter, $\mathcal{ID} = \{\mathcal{ID}_\lambda\}_{\lambda \in \mathbb{N}}$ be the identity space, $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda \in \mathbb{N}}$ be the message space, ℓ be a fixed leakage parameter, $N = N(\lambda, \ell)$ be a key-size parameter, and $d = d(\lambda)$ be an output size parameter. Our construction relies on the following primitives:

- Let $\Pi_{\mathsf{WE}} = (\mathsf{WE.Encrypt}, \mathsf{WE.Decrypt})$ be a witness encryption scheme with message space $\mathcal{M}$ and relation $\mathcal{R}_{\mathsf{WE}}$ (with corresponding language $\mathcal{L}_{\mathsf{WE}}$), which is defined as follows:

Statement: common reference strings $\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{crs}_{\mathsf{SE}}, \mathsf{crs}_{\mathsf{PTE}}$, a vector of common reference strings $\mathsf{crs}^{(I)}_{\mathsf{Com}}$ indexed by a set $I \subset \mathbb{N}$, circuit C, a tuple of commitments $\vec{c}_I$, a trapdoor commitment c_τ, an identity id, a split encoding $\mathsf{ct}_{\mathsf{out}}$, and a privately-testable encoding $\mathsf{ct}_{\mathsf{aux}}$
Witness: a split encoding $\mathsf{sk}_{\mathsf{out}}$, a privately-testable encoding $\mathsf{sk}_{\mathsf{aux}}$, and a proof π

Output 1 if all of the following conditions hold:

- $\mathsf{NIZK.Verify}\big(\mathsf{crs}_{\mathsf{NIZK}}, C, \big(\mathsf{crs}^{(I)}_{\mathsf{Com}}, \vec{c}_I, c_\tau, \mathsf{sk}_{\mathsf{out}}, \mathsf{sk}_{\mathsf{aux}}, \mathsf{id}\big), \pi\big) = 1.$
- $\mathsf{PTE.Test}(\mathsf{crs}_{\mathsf{PTE}}, \mathsf{sk}_{\mathsf{out}}, \mathsf{ct}_{\mathsf{out}}) = 0.$
- $\mathsf{SE.Test}(\mathsf{crs}_{\mathsf{SE}}, \mathsf{sk}_{\mathsf{aux}}, \mathsf{ct}_{\mathsf{aux}}) = 0.$

Otherwise, output 0.

Figure 6: Relation $\mathcal{R}_{\mathsf{WE}}$.

- Let $\mathcal{H} \colon \mathcal{ID}_\lambda \to [N]^d$ be a hash function that can be computed in time $\mathrm{poly}(\lambda, \log N)$ and let the output in $[N]^d$ be interpreted as a set of d indices of $[N]$.
- Let $\Pi_{\mathsf{Com}} = (\mathsf{Com.Setup}, \mathsf{Com.Commit}, \mathsf{Com.Verify})$ be a one-time dual-mode commitment scheme (Definition 2.15) with input space $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda \in \mathbb{N}}$, and let $\ell_x = \ell_x(\lambda)$ be the bit-length of an input. Let $(\mathcal{S}_{\mathsf{com}}, \mathcal{S}_{\mathsf{open}})$ be the simulator associated with the commitment scheme.
- Let $\Pi_{\mathsf{SE}} = (\mathsf{SE.Setup}, \mathsf{SE.SetupSF}, \mathsf{SE.Encode}, \mathsf{SE.EncodeSF}, \mathsf{SE.Test})$ be a split encoding scheme with tag space $\mathcal{ID}$.
- Let $\Pi_{\mathsf{PTE}} = \mathsf{PTE}.(\mathsf{Setup}, \mathsf{SetupSF}, \mathsf{Samp}, \mathsf{SampSF}, \mathsf{Encode}, \mathsf{EncodeSF}, \mathsf{Test})$ be a privately-testable encoding scheme with input space $\mathcal{X}_{\mathsf{PTE}} = \{\mathcal{X}^d_\lambda\}_{\lambda \in \mathbb{N}}$.
- Let $\Pi_{\mathsf{NIZK}} = (\mathsf{NIZK.Setup}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Verify})$ be a NIZK for NP.
- For common reference strings $\mathsf{crs}_{\mathsf{SE}}, \mathsf{crs}_{\mathsf{PTE}}$, define the NP relation $\mathcal{R}[\mathsf{crs}_{\mathsf{SE}}, \mathsf{crs}_{\mathsf{PTE}}]$ as follows:

Hard-wired: common reference strings $\mathsf{crs}_{\mathsf{SE}}, \mathsf{crs}_{\mathsf{PTE}}, \mathsf{crs}^{(\tau)}_{\mathsf{Com}}$
Statement: a vector of common reference string $\mathsf{crs}^{(I)}_{\mathsf{Com}}$ indexed by a set $I \subset \mathbb{N}$, commitments $\vec{c}_I$, c_τ, encoding $\mathsf{sk}_{\mathsf{out}}$, encoding $\mathsf{sk}_{\mathsf{aux}}$, identity id
Witness: $\vec{r}_I$, randomness $s_{\mathsf{out}}, s_{\mathsf{aux}}$, openings $\vec{\sigma}_I$, σ_τ, trapdoor $\mathsf{td}_{\mathsf{SE}}$

Output 1 if either of the following conditions hold:

- $\mathsf{sk}_{\mathsf{out}} = \mathsf{PTE.Encode}(\mathsf{crs}_{\mathsf{PTE}}, \vec{r}_I; s_{\mathsf{out}})$ and for each $i \in I$, $\mathsf{Com.Verify}(\mathsf{crs}^{(i)}_{\mathsf{Com}}, c_i, r_i, \sigma_i) = 1.$
- $\mathsf{sk}_{\mathsf{aux}} = \mathsf{SE.EncodeSF}(\mathsf{crs}_{\mathsf{SE}}, \mathsf{td}_{\mathsf{SE}}, \mathsf{id}, 0; s_{\mathsf{aux}})$ and $\mathsf{Com.Verify}\big(\mathsf{crs}^{(\tau)}_{\mathsf{Com}}, c_\tau, \mathsf{td}_{\mathsf{SE}}, \sigma_\tau\big) = 1.$

Otherwise, output 0.

Figure 7: Relation $\mathcal{R}[\mathsf{crs}_{\mathsf{SE}}, \mathsf{crs}_{\mathsf{PTE}}, \mathsf{crs}^{(\tau)}_{\mathsf{Com}}]$.

We now construct our big-key IBE scheme $\Pi_{\mathsf{bkIBE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ as follows:

- $\mathsf{Setup}(1^\lambda, 1^\ell)$: On input the security parameter λ, the setup algorithm proceeds as follows:
  1. Sample $\mathsf{crs}_{\mathsf{SE}} \leftarrow \mathsf{SE.Setup}(1^\lambda)$, $\mathsf{crs}_{\mathsf{PTE}} \leftarrow \mathsf{PTE.Setup}(1^\lambda)$, $\mathsf{crs}_{\mathsf{NIZK}} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$.
  2. For all $i \in [N]$, sample
  $$r_i \xleftarrow{\mathrm{R}} \mathcal{X}_\lambda$$
  $$\left(\mathsf{crs}_{\mathsf{Com}}^{(i)}, \mathsf{td}_{\mathsf{Com}}^{(i)}, c_i\right) \leftarrow \mathsf{Com.Setup}(1^\lambda, \mathsf{hide})$$
  $$\sigma_i \leftarrow \mathcal{S}_{\mathsf{open}}\left(\mathsf{td}_{\mathsf{Com}}^{(i)}, r_i\right).$$
  3. Finally, sample $\left(\mathsf{crs}_{\mathsf{Com}}^{(\tau)}, \mathsf{td}_{\mathsf{Com}}^{(\tau)}, c_\tau\right) \leftarrow \mathsf{Com.Setup}(1^\lambda, \mathsf{hide})$.

  Let $\vec{c} = (c_1, \ldots, c_N)$, $\vec{r} = (r_1, \ldots, r_N)$ and $\vec{\sigma} = (\sigma_1, \ldots, \sigma_N)$. Output the public parameters
  $$\mathsf{pp} = \left(\left\{\mathsf{crs}_{\mathsf{Com}}^{(i)}\right\}_{i \in [N]}, \vec{c}, \mathsf{crs}_{\mathsf{NIZK}}, \mathsf{crs}_{\mathsf{SE}}, \mathsf{crs}_{\mathsf{PTE}}, \mathsf{crs}_{\mathsf{Com}}^{(\tau)}, c_\tau\right) \tag{5.1}$$
  and the master secret key $\mathsf{msk} = (\mathsf{pp}, \vec{r}, \vec{\sigma})$. For a set $I \subseteq [N]$, we write $\vec{c}_I$, $\vec{r}_I$, and $\vec{\sigma}_I$ to denote the respective sub-vector of indices in I. Similarly, we define $\mathsf{crs}_{\mathsf{Com}}^{(I)} := \left(\mathsf{crs}_{\mathsf{Com}}^{(i)}\right)_{i \in I}$.

- $\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$: On input the master secret key $\mathsf{msk} = (\mathsf{pp}, \vec{r}, \vec{\sigma})$ (with pp parsed according to Eq. (5.1)) and an identity $\mathsf{id} \in \mathcal{ID}_\lambda$, the key generation algorithm proceeds as follows:
  1. Compute $I \leftarrow \mathcal{H}(\mathsf{id})$.
  2. Compute $\mathsf{sk}_{\mathsf{out}} \leftarrow \mathsf{PTE.Encode}(\mathsf{crs}_{\mathsf{PTE}}, \vec{r}_I; s_{\mathsf{out}})$, where s_{out} is the encoding randomness.
  3. Sample $\mathsf{sk}_{\mathsf{aux}} \leftarrow \mathsf{SE.Encode}(\mathsf{crs}_{\mathsf{SE}}, 0)$.
  4. Compute $\pi \leftarrow \mathsf{NIZK.Prove}\left(\mathsf{crs}_{\mathsf{NIZK}}, C, \left(\mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, c_\tau, \mathsf{sk}_{\mathsf{out}}, \mathsf{sk}_{\mathsf{aux}}, \mathsf{id}\right), \left(\vec{r}_I, s_{\mathsf{out}}, \bot, \vec{\sigma}_I, \bot, \bot\right)\right)$, where C is the circuit that computes $\mathcal{R}[\mathsf{crs}_{\mathsf{SE}}, \mathsf{crs}_{\mathsf{PTE}}, \mathsf{crs}_{\mathsf{Com}}^{(\tau)}]$ from Fig. 7.

  Output the identity secret key $\mathsf{sk}_{\mathsf{id}} = (\mathsf{sk}_{\mathsf{out}}, \mathsf{sk}_{\mathsf{aux}}, \pi)$.

- $\mathsf{Encrypt}(\mathsf{pp}, \mathsf{id}, m)$: On input the public parameters pp (parsed as in Eq. (5.1)), an identity $\mathsf{id} \in \mathcal{ID}_\lambda$ and a message m, the encryption algorithm does the following:
  1. Compute $I \leftarrow \mathcal{H}(\mathsf{id})$, sample $\mathsf{ct}_{\mathsf{out}} \leftarrow \mathsf{PTE.Samp}(1^\lambda)$, $\mathsf{ct}_{\mathsf{aux}} \leftarrow \mathsf{SE.Encode}(\mathsf{crs}_{\mathsf{SE}}, 1)$.
  2. Compute $\mathsf{ct} \leftarrow \mathsf{WE.Encrypt}(1^\lambda, m, (\widetilde{\mathsf{crs}}, C, \mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, c_\tau, \mathsf{id}, \mathsf{ct}_{\mathsf{out}}, \mathsf{ct}_{\mathsf{aux}}))$, where C is the circuit that computes $\mathcal{R}[\mathsf{crs}_{\mathsf{SE}}, \mathsf{crs}_{\mathsf{PTE}}, \mathsf{crs}_{\mathsf{Com}}^{(\tau)}]$ from Fig. 7 and $\widetilde{\mathsf{crs}} = (\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{crs}_{\mathsf{SE}}, \mathsf{crs}_{\mathsf{PTE}})$.

  Output the ciphertext ct.

- $\mathsf{Decrypt}(\mathsf{sk}_{\mathsf{id}}, \mathsf{id}, \mathsf{ct})$: On input an identity secret key $\mathsf{sk}_{\mathsf{id}}$, an identity $\mathsf{id} \in \mathcal{ID}_\lambda$, and a ciphertext ct, the decryption algorithm outputs $\mathsf{WE.Decrypt}(\mathsf{ct}, \mathsf{sk}_{\mathsf{id}})$.

Theorem 5.4 (Correctness)**.** *Suppose Π_{SE} satisfies tester correctness, Π_{PTE} satisfies tester correctness, Π_{NIZK} satisfies completeness, Π_{WE} satisfies correctness, and Π_{Com} satisfies correctness. Then, Construction 5.3 is correct.*

Proof. Take any security parameter λ, identity $\mathsf{id} \in \mathcal{ID}_\lambda$, and message m. Let $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$, where
$$\mathsf{pp} = \left(\left\{\mathsf{crs}_{\mathsf{Com}}^{(i)}\right\}_{i \in [N]}, \vec{c}, \mathsf{crs}_{\mathsf{NIZK}}, \mathsf{crs}_{\mathsf{SE}}, \mathsf{crs}_{\mathsf{PTE}}, \mathsf{crs}_{\mathsf{Com}}^{(\tau)}, c_\tau\right)$$
and $\mathsf{msk} = (\mathsf{pp}, \vec{r}, \vec{\sigma})$. Let $\mathsf{sk}_{\mathsf{id}} = (\mathsf{sk}_{\mathsf{out}}, \mathsf{sk}_{\mathsf{aux}}, \pi) \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$ and $\mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{pp}, \mathsf{id}, m)$. Consider the output of the algorithm $\mathsf{Decrypt}(\mathsf{sk}_{\mathsf{id}}, \mathsf{id}, \mathsf{ct})$:

- By construction of KeyGen and correctness of Π_{Com}, we have
$$((\mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, c_\tau, \mathsf{sk}_{\mathsf{out}}, \mathsf{sk}_{\mathsf{aux}}, \mathsf{id}), (\vec{r}_I, s_{\mathsf{out}}, \bot, \vec{\sigma}_I, \bot, \bot)) \in \mathcal{R}[\mathsf{crs}_{\mathsf{SE}}, \mathsf{crs}_{\mathsf{PTE}}, \mathsf{crs}_{\mathsf{Com}}^{(\tau)}],$$
and $\pi \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_{\mathsf{NIZK}}, C, (\mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, c_\tau, \mathsf{sk}_{\mathsf{out}}, \mathsf{sk}_{\mathsf{aux}}, \mathsf{id}), (\vec{r}_I, s_{\mathsf{out}}, \bot, \vec{\sigma}_I, \bot, \bot))$.

- By construction of Encrypt and correctness of Π_{WE}, $\mathsf{WE.Decrypt}(\mathsf{ct}, \mathsf{sk}_{\mathsf{id}}) = m$ if
$$((\vec{\mathsf{crs}}, C, \mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, c_\tau, \mathsf{id}, \mathsf{ct}_{\mathsf{out}}, \mathsf{ct}_{\mathsf{aux}}), (\mathsf{sk}_{\mathsf{id}})) \in \mathcal{R}_{\mathsf{WE}},$$
where $I \leftarrow \mathcal{H}(\mathsf{id}), \mathsf{ct}_{\mathsf{out}} \leftarrow \mathsf{PTE.Samp}(1^\lambda), \mathsf{ct}_{\mathsf{aux}} \leftarrow \mathsf{SE.Encode}(\mathsf{crs}_{\mathsf{SE}}, 1)$, C is the circuit that computes $\mathcal{R}[\mathsf{crs}_{\mathsf{SE}}, \mathsf{crs}_{\mathsf{PTE}}, \mathsf{crs}_{\mathsf{Com}}^{(\tau)}]$ from Fig. 7, and $\vec{\mathsf{crs}} = (\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{crs}_{\mathsf{SE}}, \mathsf{crs}_{\mathsf{PTE}})$.

- By completeness of Π_{NIZK}, the proof π verifies, and by tester correctness for Π_{SE} and Π_{PTE}, with overwhelming probability over the choice of $\mathsf{crs}_{\mathsf{PTE}}$ and $\mathsf{crs}_{\mathsf{SE}}$,
$$\mathsf{PTE.Test}(\mathsf{crs}_{\mathsf{PTE}}, \mathsf{sk}_{\mathsf{out}}, \mathsf{ct}_{\mathsf{out}}) = 0$$
$$\mathsf{SE.Test}(\mathsf{crs}_{\mathsf{SE}}, \mathsf{sk}_{\mathsf{aux}}, \mathsf{ct}_{\mathsf{aux}}) = 0.$$
Thus $\mathsf{WE.Decrypt}(\mathsf{ct}, \mathsf{sk}_{\mathsf{id}}) = m$ with overwhelming probability, as required. □

Theorem 5.5 (Efficiency). *If $\mathcal{H}$ runs in $\mathsf{poly}(\lambda, \log N)$-time, then Construction 5.3 is efficient.*

Proof. This holds by inspection and assumption on $\mathcal{H}$, since the other primitives are all efficient, and thus, run in $\mathsf{poly}(\lambda)$-time by definition. Furthermore, the KeyGen and Encrypt algorithms only require $\mathsf{poly}(\lambda) \cdot d(\lambda)$ bits of msk or pp, which is independent of the leakage parameter ℓ. □

Theorem 5.6 (Adaptive Advantage-Checker Security under Bounded Leakage). *Suppose the following conditions hold:*

- *The witness encryption scheme Π_{WE} satisfies semantic security.*
- *The hash function $\mathcal{H}$ is (k, α)-expanding where $\alpha(\lambda) = \omega(\log \lambda)$.*
- *The split encoding Π_{SE} satisfies mode indistinguishability and tester correctness.*
- *The privately testable encoding Π_{PTE} satisfies tester correctness, $(\omega(\log \lambda) \cdot \ell_x)$-trapdoor indistinguishability, and mode indistinguishability.*
- *The NIZK Π_{NIZK} satisfies statistical soundness and computational witness indistinguishability.*
- *The one-time dual-mode commitment scheme Π_{Com} satisfies mode indistinguishability and statistical binding in binding mode.*

Then for all polynomially-bounded $\ell = \ell(\lambda)$, Construction 5.3 is adaptively advantage-checker secure under bounded leakage for challenge parameter $k \geq \frac{\ell}{\eta \alpha \ell_x}$, where $\eta \in (0, 1)$ is a constant.

Proof. We define a sequence of hybrid experiments, each parameterized (implicitly) by an (admissible) adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and an advantage threshold function $\varepsilon = \varepsilon(\lambda)$:

- Hyb_0: This is the adaptive advantage-checker security game from Definition 3.1, which we recall in full below:

- **Setup:** The challenger runs $\mathsf{crs}_{\mathsf{SE}} \leftarrow \mathsf{SE.Setup}(1^\lambda)$, $\mathsf{crs}_{\mathsf{PTE}} \leftarrow \mathsf{PTE.Setup}(1^\lambda)$, $\mathsf{crs}_{\mathsf{NIZK}} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$. For all $i \in [N]$, it samples $r_i \xleftarrow{\mathrm{R}} \mathcal{X}_\lambda$, $\left(\mathsf{crs}^{(i)}_{\mathsf{Com}}, \mathsf{td}^{(i)}_{\mathsf{Com}}, c_i\right) \leftarrow \mathsf{Com.Setup}(1^\lambda, \mathsf{hide})$ and computes $\sigma_i \leftarrow \mathcal{S}_{\mathsf{open}}\left(\mathsf{td}^{(i)}_{\mathsf{Com}}, r_i\right)$. It also samples $\left(\mathsf{crs}^{(\tau)}_{\mathsf{Com}}, \mathsf{td}^{(\tau)}_{\mathsf{Com}}, c_\tau\right) \leftarrow \mathsf{Com.Setup}(1^\lambda, \mathsf{hide})$, and sets
$$\mathsf{pp} = \left(\left\{\mathsf{crs}^{(i)}_{\mathsf{Com}}\right\}_{i\in[N]}, \vec{c}, \mathsf{crs}_{\mathsf{NIZK}}, \mathsf{crs}_{\mathsf{SE}}, \mathsf{crs}_{\mathsf{PTE}}, \mathsf{crs}^{(\tau)}_{\mathsf{Com}}, c_\tau\right)$$
and $\mathsf{msk} = (\mathsf{pp}, \vec{r}, \vec{\sigma})$. The challenger gives pp to $\mathcal{A}_1$.
- **Pre-leakage queries:** When $\mathcal{A}_1$ makes a query on an identity $\mathsf{id} \in \mathcal{ID}_\lambda$, the challenger proceeds as follows:
  * It computes $I \leftarrow \mathcal{H}(\mathsf{id})$ and encoding keys $\mathsf{sk}_{\mathsf{out}} \leftarrow \mathsf{PTE.Encode}(\mathsf{crs}_{\mathsf{PTE}}, \vec{r}_I; s_{\mathsf{out}})$ and $\mathsf{sk}_{\mathsf{aux}} \leftarrow \mathsf{SE.Encode}(\mathsf{crs}_{\mathsf{SE}}, 0)$.
  * Next, it construct the NIZK proof
$$\pi \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_{\mathsf{NIZK}}, C, (\mathsf{crs}^{(I)}_{\mathsf{Com}}, \vec{c}_I, c_\tau, \mathsf{sk}_{\mathsf{out}}, \mathsf{sk}_{\mathsf{aux}}, \mathsf{id}), (\vec{r}_I, s_{\mathsf{out}}, \bot, \vec{\sigma}_I, \bot, \bot)),$$
where C is the circuit that computes $\mathcal{R}[\mathsf{crs}_{\mathsf{SE}}, \mathsf{crs}_{\mathsf{PTE}}, \mathsf{crs}^{(\tau)}_{\mathsf{Com}}]$ from Fig. 7.

  The challenger replies with $\mathsf{sk}_{\mathsf{id}} = (\mathsf{sk}_{\mathsf{out}}, \mathsf{sk}_{\mathsf{aux}}, \pi)$.
- **Leakage:** After $\mathcal{A}_1$ outputs the description of an efficiently-computable leakage function f, the challenger replies with $\mathsf{leak} \leftarrow f(\mathsf{msk})$.
- **Post-leakage queries:** The challenger responds to post-leakage key queries exactly as in the pre-leakage phase.
- **Challenge:** Algorithm $\mathcal{A}_1$ outputs a set $\mathcal{J} \subseteq \mathcal{ID}_\lambda$ of size $\geq k$, two messages m_0, m_1, and a state st.
- **Output:** The output of Hyb_0 is $b' = 1$ if
$$\forall \mathsf{id} \in \mathcal{J} : \mathsf{AdvCheck}^{\mathcal{A}_2}(1^\lambda, 1^{1/\varepsilon}, \mathsf{id}, \mathsf{msk}, \mathsf{pp}, \mathsf{st}, \mathsf{leak}) = 1, \tag{5.2}$$
and $b' = 0$ otherwise. The advantage-checker algorithm AdvCheck is defined as follows:

Inputs: security parameter λ, threshold $\varepsilon \in (0,1)$, identity $\mathsf{id} \in \mathcal{ID}_\lambda$, master secret key $\mathsf{msk} = (r, \sigma_1, \dots \sigma_N)$, public parameters pp (parsed as in Eq. (5.1)), state st, string leak, and (oracle) access to an algorithm $\mathcal{A}$

- Let $T = \lambda/\varepsilon^2$ and initialize a counter $\mathsf{WINS} \leftarrow 0$.
- The advantage-checker algorithm now simulates T independent executions of experiment $\mathsf{Exp}^{\mathsf{id}}(\mathsf{msk}, \mathsf{pp}, \mathsf{st}, \mathsf{leak})$ for algorithm $\mathcal{A}$.
  1. Sample $\beta \xleftarrow{\mathrm{R}} \{0,1\}$.
  2. Compute $I \leftarrow \mathcal{H}(\mathsf{id})$, $\mathsf{ct}_{\mathsf{out}} \leftarrow \mathsf{PTE.Samp}(1^\lambda)$, $\mathsf{ct}_{\mathsf{aux}} \leftarrow \mathsf{SE.Encode}(\mathsf{crs}_{\mathsf{SE}}, 1)$.
  3. Compute $\mathsf{ct} \leftarrow \mathsf{WE.Encrypt}\big(1^\lambda, m_\beta, (\vec{\mathsf{crs}}, C, \mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, c_\tau, \mathsf{id}, \mathsf{ct}_{\mathsf{out}}, \mathsf{ct}_{\mathsf{aux}})\big)$, where C is the circuit that computes $\mathcal{R}[\mathsf{crs}_{\mathsf{SE}}, \mathsf{crs}_{\mathsf{PTE}}, \mathsf{crs}_{\mathsf{Com}}^{(\tau)}]$ from Fig. 7, and $\vec{\mathsf{crs}} = (\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{crs}_{\mathsf{SE}}, \mathsf{crs}_{\mathsf{PTE}})$.
  4. Whenever algorithm $\mathcal{A}$ makes a key-generation query on an identity $\mathsf{id} \in \mathcal{ID}_\lambda$, compute $I \leftarrow \mathcal{H}(\mathsf{id})$, $\mathsf{sk}_{\mathsf{out}} \leftarrow \mathsf{PTE.Encode}(\mathsf{crs}_{\mathsf{PTE}}, \vec{r}_I; s_{\mathsf{out}})$, $\mathsf{sk}_{\mathsf{aux}} \leftarrow \mathsf{SE.Encode}(\mathsf{crs}_{\mathsf{SE}}, 0)$, and
  $$\pi \leftarrow \mathsf{NIZK.Prove}\big(\mathsf{crs}_{\mathsf{NIZK}}, C, (\mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, c_\tau, \mathsf{sk}_{\mathsf{out}}, \mathsf{sk}_{\mathsf{aux}}, \mathsf{id}), (\vec{r}_I, s_{\mathsf{out}}, \perp, \vec{\sigma}_I, \perp, \perp)\big),$$
  where C is the circuit that computes $\mathcal{R}[\mathsf{crs}_{\mathsf{SE}}, \mathsf{crs}_{\mathsf{PTE}}, \mathsf{crs}_{\mathsf{Com}}^{(\tau)}]$ from Fig. 7. The challenger replies to $\mathcal{A}$ with identity key $\mathsf{sk}_{\mathsf{id}} = (\mathsf{sk}_{\mathsf{out}}, \mathsf{sk}_{\mathsf{aux}}, \pi)$.
  5. After $\mathcal{A}$ has finished making key-generation queries, it outputs a bit $\beta' \in \{0,1\}$.
  6. If $\beta = \beta'$, then increment $\mathsf{WINS} \leftarrow \mathsf{WINS} + 1$.
- Output 1 if $\mathsf{WINS} \geq \frac{T}{2} + \frac{\varepsilon T}{2}$ and 0 otherwise.

Figure 8: Function $\mathsf{AdvCheck}^{\mathcal{A}}(1^\lambda, 1^{1/\varepsilon}, \mathsf{id}, \mathsf{msk}, \mathsf{pp}, \mathsf{st}, \mathsf{leak})$ for Construction 5.3.

- Hyb_1: Same as Hyb_0, except the challenger changes the split encoding to semi-functional mode:
  - **Setup:** The challenger now samples $(\mathsf{crs}_{\mathsf{SE}}, \mathsf{td}_{\mathsf{SE}}) \leftarrow \mathsf{SE.SetupSF}(1^\lambda)$.
  - **Key-generation queries:** Whenever $\mathcal{A}_1$ makes a key-generation query (in the pre-leakage or the post-leakage phase) or $\mathcal{A}_2$ makes a key-generation query (in $\mathsf{AdvCheck}^{\mathcal{A}_2}$) on $\mathsf{id} \in \mathcal{ID}_\lambda$, the challenger computes $\mathsf{sk}_{\mathsf{aux}} \leftarrow \mathsf{SE.EncodeSF}(\mathsf{crs}_{\mathsf{SE}}, \mathsf{td}_{\mathsf{SE}}, \mathsf{id}, 0; s_{\mathsf{aux}})$, where s_{aux} is the encoding randomness.
  - **Output:** In the $\mathsf{AdvCheck}^{\mathcal{A}_2}(1^\lambda, 1^{1/\varepsilon}, \mathsf{id}, \mathsf{msk}, \mathsf{pp}, \mathsf{st}, \mathsf{leak})$ procedure, the challenger now computes $\mathsf{ct}_{\mathsf{aux}} \leftarrow \mathsf{SE.EncodeSF}(\mathsf{crs}_{\mathsf{SE}}, \mathsf{td}_{\mathsf{SE}}, \mathsf{id}, 1)$ when constructing the challenge ciphertext ct.
- Hyb_2: Same as Hyb_1 except the challenger answers key-generation queries with the witness $(s_{\mathsf{aux}}, \sigma_\tau, \mathsf{td})$:
  - **Setup:** The challenger now computes $\sigma_\tau \leftarrow \mathcal{S}_{\mathsf{open}}(\mathsf{td}_{\mathsf{Com}}^{(\tau)}, \mathsf{td}_{\mathsf{SE}})$.[3]

[3]Here, we assume that we can interpret $\mathsf{td}_{\mathsf{SE}}$ as elements of the input space $\mathcal{X}_\lambda$. Note that this is without loss of generality since we can always take the binary representation of $\mathsf{td}_{\mathsf{SE}}$ and commit bit-by-bit.

  - **Key-generation queries:** Whenever $\mathcal{A}_1$ makes a key-generation query (in the pre-leakage or the post-leakage phase) or $\mathcal{A}_2$ makes a key-generation query (in $\mathsf{AdvCheck}^{\mathcal{A}_2}$) on $\mathsf{id} \in \mathcal{ID}_\lambda$, the challenger now constructs the NIZK proof π as

$$\pi \leftarrow \mathsf{NIZK.Prove}\big(\mathsf{crs}_{\mathsf{NIZK}}, C, \big(\mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, c_\tau, \mathsf{sk}_{\mathsf{out}}, \mathsf{sk}_{\mathsf{aux}}, \mathsf{id}\big), (\bot, \bot, s_{\mathsf{aux}}, \bot, \sigma_\tau, \mathsf{td}_{\mathsf{SE}})\big).$$

- Hyb_3: Same as Hyb_2 except the challenger changes the privately-testable encoding to semi-functional mode:
  - **Setup:** The challenger now samples $(\mathsf{crs}_{\mathsf{PTE}}, \mathsf{td}_{\mathsf{PTE}}) \leftarrow \mathsf{PTE.SetupSF}(1^\lambda)$.
  - **Key-generation queries:** Whenever $\mathcal{A}_1$ makes a key-generation query (in the pre-leakage or the post-leakage phase) or $\mathcal{A}_2$ makes a key-generation query (in $\mathsf{AdvCheck}^{\mathcal{A}_2}$) on $\mathsf{id} \in \mathcal{ID}_\lambda$, the challenger samples $\mathsf{sk}_{\mathsf{out}} \leftarrow \mathsf{PTE.EncodeSF}(\mathsf{crs}_{\mathsf{PTE}})$.
- Hyb_4: Same as Hyb_3 except during the output phase, for each challenge identity $\mathsf{id} \in \mathcal{J}$, the challenger computes

$$\mathsf{ct}_{\mathsf{out}}^{(\mathsf{id})} \leftarrow \mathsf{PTE.SampSF}\big(\mathsf{td}_{\mathsf{PTE}}, \vec{r}_{\mathcal{H}(\mathsf{id})}\big).$$

  When computing the advantage-checker algorithm $\mathsf{AdvCheck}$ for the identity $\mathsf{id} \in \mathcal{J}$, the challenger uses $\mathsf{ct}_{\mathsf{out}}^{(\mathsf{id})}$ in place of $\mathsf{ct}_{\mathsf{out}}$ in all T iterations. In other words, for each identity $\mathsf{id} \in \mathcal{J}$, the challenger now uses the *same* $\mathsf{ct}_{\mathsf{out}} := \mathsf{ct}_{\mathsf{out}}^{(\mathsf{id})}$ in the T executions of $\mathsf{AdvCheck}$ for id. In addition, in this experiment, $\mathsf{AdvCheck}^{\mathcal{A}_2}(1^\lambda, 1^{1/\varepsilon}, \mathsf{id}, \mathsf{msk}, \mathsf{pp}, \mathsf{st}, \mathsf{leak})$ outputs 1 if

$$\mathsf{WINS} \ge \frac{T}{2} + \frac{\varepsilon T}{16}$$

  and 0 otherwise. Finally, the output of this experiment is $b' = 1$ only if

$$\exists \mathsf{id} \in \mathcal{J} : \mathsf{AdvCheck}^{\mathcal{A}_2}\big(1^\lambda, 1^{1/\varepsilon}, \mathsf{id}, \mathsf{msk}, \mathsf{pp}, \mathsf{st}, \mathsf{leak}\big) = 1. \tag{5.3}$$

- Hyb_5: Same as Hyb_4 except the challenger samples the commitment CRS in binding mode and constructs the commitments and openings without the simulation algorithms:
  - **Setup:** The challenge now computes the quantities $\big(\mathsf{crs}_{\mathsf{Com}}^{(i)}, c_i, \sigma_i\big)$ as

$$\mathsf{crs}_{\mathsf{Com}}^{(i)} \leftarrow \mathsf{Com.Setup}(1^\lambda, \mathsf{bind}) \quad \text{and} \quad (c_i, \sigma_i) \leftarrow \mathsf{Com.Commit}(\mathsf{crs}_{\mathsf{Com}}, r_i)$$

    for all $i \in [N]$ and $\big(\mathsf{crs}_{\mathsf{Com}}^{(\tau)}, c_\tau, \sigma_\tau\big)$ as

$$\mathsf{crs}_{\mathsf{Com}}^{(\tau)} \leftarrow \mathsf{Com.Setup}(1^\lambda, \mathsf{bind}) \quad \text{and} \quad (c_\tau, \sigma_\tau) \leftarrow \mathsf{Com.Commit}(\mathsf{crs}_{\mathsf{Com}}, \mathsf{td}_{\mathsf{SE}}).$$

- Hyb_6: Same as Hyb_5 except the challenge constructs the challenge ciphertext ct in the procedure $\mathsf{AdvCheck}^{\mathcal{A}_2}$ as $\mathsf{ct} \leftarrow \mathsf{WE.Encrypt}\big(1^\lambda, 0, \big(\vec{\mathsf{crs}}, C, \mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, c_\tau, \mathsf{id}, \mathsf{ct}_{\mathsf{out}}, \mathsf{ct}_{\mathsf{aux}}\big)\big)$.

For convenience, we will refer to $\mathsf{AdvCheck}^{\mathcal{A}}(1^\lambda, 1^{1/\varepsilon}, \mathsf{id}, \mathsf{msk}, \mathsf{pp}, \mathsf{st}, \mathsf{leak})$ as $\mathsf{AdvCheck}(\mathsf{id})$ when the non-id parameters are fixed in a given context. For an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, we write $\mathsf{Hyb}_i(\mathcal{A}, \varepsilon)$ to denote the output of Hyb_i with adversary $\mathcal{A}$ and inner threshold function ε. Our goal is to show that for all efficient adversaries $\mathcal{A}$ and all inverse polynomial functions $\varepsilon = 1/\mathrm{poly}(\lambda)$, $\Pr[\mathsf{Hyb}_0(\mathcal{A}, \varepsilon) = 1] = \mathsf{negl}(\lambda)$. We now analyze each pair of adjacent experiments:

Lemma 5.7. *Suppose* Π_{SE} *satisfies mode indistinguishability. Then, for all efficient and admissible adversaries* $\mathcal{A}$ *and inverse polynomial functions* $\varepsilon = 1/\mathsf{poly}(\lambda)$*, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$,

$$\Pr[\mathsf{Hyb}_1(\mathcal{A}, \varepsilon) = 1] \geq \Pr[\mathsf{Hyb}_0(\mathcal{A}, \varepsilon) = 1] - \mathsf{negl}(\lambda).$$

Proof. Suppose there exists an efficient adversary $\mathcal{A}$ that distinguishes Hyb_0 and Hyb_1 with non-negligible advantage δ. We use $\mathcal{A}$ to construct an algorithm $\mathcal{B}$ that breaks mode indistinguishability:

1. At the beginning of the game, algorithm $\mathcal{B}$ gets $\mathsf{crs}_{\mathsf{SE}}$ from the mode indistinguishability challenger. Algorithm $\mathcal{B}$ samples $\mathsf{crs}_{\mathsf{PTE}} \leftarrow \mathsf{PTE.Setup}(1^\lambda)$ and $\mathsf{crs}_{\mathsf{NIZK}} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$. For all $i \in [N]$, algorithm $\mathcal{B}$ samples $r_i \xleftarrow{\mathrm{R}} \mathcal{X}_\lambda$, $\left(\mathsf{crs}_{\mathsf{Com}}^{(i)}, \mathsf{td}_{\mathsf{Com}}^{(i)}, c_i\right) \leftarrow \mathsf{Com.Setup}(1^\lambda, \mathsf{hide})$ and computes $\sigma_i \leftarrow \mathcal{S}_{\mathsf{open}}(\mathsf{td}_{\mathsf{Com}}^{(i)}, r_i)$. Algorithm $\mathcal{B}$ also samples $\left(\mathsf{crs}_{\mathsf{Com}}^{(\tau)}, \mathsf{td}_{\mathsf{Com}}^{(\tau)}, c_\tau\right) \leftarrow \mathsf{Com.Setup}(1^\lambda, \mathsf{hide})$ and sets
$$\mathsf{pp} = \left(\left\{\mathsf{crs}_{\mathsf{Com}}^{(i)}\right\}_{i \in [N]}, \ \vec{c}, \ \mathsf{crs}_{\mathsf{NIZK}}, \ \mathsf{crs}_{\mathsf{SE}}, \ \mathsf{crs}_{\mathsf{PTE}}, \ \mathsf{crs}_{\mathsf{Com}}^{(\tau)}, \ c_\tau\right) \quad \text{and} \quad \mathsf{msk} = (\mathsf{pp}, \vec{r}, \vec{\sigma})$$
and gives pp to $\mathcal{A}$.

2. When algorithm $\mathcal{A}$ makes a key-generation query on $\mathsf{id} \in \mathcal{ID}_\lambda$, algorithm $\mathcal{B}$ computes $\mathsf{sk}_{\mathsf{aux}}$ by sending an encoding query $(\mathsf{id}, 0)$ to the mode indistinguishability challenger, and setting $\mathsf{sk}_{\mathsf{aux}}$ as the output of the query. Algorithm $\mathcal{B}$ computes $I \leftarrow \mathcal{H}(\mathsf{id})$, $\mathsf{sk}_{\mathsf{out}} \leftarrow \mathsf{PTE.Encode}(\mathsf{crs}_{\mathsf{PTE}}, \vec{r}_I; s_{\mathsf{out}})$, and
$$\pi \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_{\mathsf{NIZK}}, C, (\mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, c_\tau, \mathsf{sk}_{\mathsf{out}}, \mathsf{sk}_{\mathsf{aux}}, \mathsf{id}), (\vec{r}_I, s_{\mathsf{out}}, \perp, \vec{\sigma}_I, \perp, \perp)),$$
where C is the circuit that computes $\mathcal{R}[\mathsf{crs}_{\mathsf{SE}}, \mathsf{crs}_{\mathsf{PTE}}, \mathsf{crs}_{\mathsf{Com}}^{(\tau)}]$ (Fig. 7). Algorithm $\mathcal{B}$ gives the key $\mathsf{sk}_{\mathsf{id}} = (\mathsf{sk}_{\mathsf{out}}, \mathsf{sk}_{\mathsf{aux}}, \pi)$ to algorithm $\mathcal{A}$.

3. When algorithm $\mathcal{A}$ outputs an efficiently-computable leakage function f, algorithm $\mathcal{B}$ replies with $\mathsf{leak} = f(\mathsf{msk})$. In the challenge phase, algorithm $\mathcal{A}$ outputs $\mathcal{J} \subseteq \mathcal{ID}_\lambda$ of size at least k, messages m_0, m_1, and a state st.

4. For all $\mathsf{id} \in \mathcal{J}$ and $i \in [T]$, algorithm $\mathcal{B}$ computes $\mathsf{ct}_{\mathsf{aux}}$ by sending an encoding query $(\mathsf{id}, 1)$ to the mode indistinguishability challenger, and setting $\mathsf{ct}_{\mathsf{aux}}$ as the output of the query. Algorithm $\mathcal{B}$ computes $I \leftarrow \mathcal{H}(\mathsf{id})$, samples $\beta \xleftarrow{\mathrm{R}} \{0,1\}$, $\mathsf{ct}_{\mathsf{out}} \leftarrow \mathsf{PTE.Samp}(1^\lambda)$, and constructs the challenge ciphertext ct in iteration i of the procedure $\mathsf{AdvCheck}(\mathsf{id})$ as
$$\mathsf{ct} \leftarrow \mathsf{WE.Encrypt}\left(1^\lambda, m_\beta, \left(\vec{\mathsf{crs}}, C, \mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, c_\tau, \mathsf{id}, \mathsf{ct}_{\mathsf{out}}, \mathsf{ct}_{\mathsf{aux}}\right)\right),$$
where C is the circuit that computes $\mathcal{R}[\mathsf{crs}_{\mathsf{SE}}, \mathsf{crs}_{\mathsf{PTE}}, \mathsf{crs}_{\mathsf{Com}}^{(\tau)}]$ (Fig. 7), and $\vec{\mathsf{crs}} = (\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{crs}_{\mathsf{SE}}, \mathsf{crs}_{\mathsf{PTE}})$. Algorithm $\mathcal{B}$ outputs the output of the experiment.

If algorithm $\mathcal{A}$ is admissible, then so is algorithm $\mathcal{B}$ since the identities correspond to the tags in Π_{SE} (namely, the identities in the challenge set $\mathcal{J}$ are disjoint from the ones that algorithm $\mathcal{A}$ queried to the key-generation oracle). If the setup and encodings are in normal mode, algorithm $\mathcal{B}$ simulates $\mathsf{Hyb}_0(\mathcal{A}, \varepsilon)$. If the setup and encodings are in semi-functional mode, algorithm $\mathcal{B}$ simulates $\mathsf{Hyb}_1(\mathcal{A}, \varepsilon)$. Thus, algorithm $\mathcal{B}$ has advantage δ in the mode indistinguishability game. □

Lemma 5.8. *Suppose* Π_{NIZK} *satisfies computational witness indistinguishability. Then, for all efficient and admissible adversaries* $\mathcal{A}$ *and inverse polynomial functions* $\varepsilon = 1/\mathsf{poly}(\lambda)$*, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$,

$$\Pr[\mathsf{Hyb}_2(\mathcal{A}, \varepsilon) = 1] \geq \Pr[\mathsf{Hyb}_1(\mathcal{A}, \varepsilon) = 1] - \mathsf{negl}(\lambda).$$

Proof. Suppose there exists an efficient adversary $\mathcal{A}$ that distinguishes Hyb_1 and Hyb_2 with non-negligible advantage δ. We use $\mathcal{A}$ to construct an algorithm $\mathcal{B}$ that breaks computational witness-indistinguishability:

1. At the beginning of the game, algorithm $\mathcal{B}$ gets $(1^\lambda, \mathsf{crs}_{\mathsf{NIZK}})$ from the witness indistinguishability challenger. Algorithm $\mathcal{B}$ samples $\mathsf{crs}_{\mathsf{PTE}} \leftarrow \mathsf{PTE.Setup}(1^\lambda)$ and $(\mathsf{crs}_{\mathsf{SE}}, \mathsf{td}_{\mathsf{SE}}) \leftarrow \mathsf{SE.SetupSF}(1^\lambda)$. For all $i \in [N]$, algorithm $\mathcal{B}$ samples $r_i \xleftarrow{\text{R}} \mathcal{X}_\lambda$, $(\mathsf{crs}^{(i)}_{\mathsf{Com}}, \mathsf{td}^{(i)}_{\mathsf{Com}}, c_i) \leftarrow \mathsf{Com.Setup}(1^\lambda, \mathsf{hide})$ and computes $\sigma_i \leftarrow \mathcal{S}_{\mathsf{open}}(\mathsf{td}^{(i)}_{\mathsf{Com}}, r_i)$. Algorithm $\mathcal{B}$ samples $\left(\mathsf{crs}^{(\tau)}_{\mathsf{Com}}, \mathsf{td}^{(\tau)}_{\mathsf{Com}}, c_\tau\right) \leftarrow \mathsf{Com.Setup}(1^\lambda, \mathsf{hide})$ and computes $\sigma_\tau \leftarrow \mathcal{S}_{\mathsf{open}}(\mathsf{td}^{(\tau)}_{\mathsf{Com}}, \mathsf{td}_{\mathsf{SE}})$, which is possible by assumption. Algorithm $\mathcal{B}$ sets
$$\mathsf{pp} = \left(\left\{\mathsf{crs}^{(i)}_{\mathsf{Com}}\right\}_{i\in[N]}, \vec{c}, \mathsf{crs}_{\mathsf{NIZK}}, \mathsf{crs}_{\mathsf{SE}}, \mathsf{crs}_{\mathsf{PTE}}, \mathsf{crs}^{(\tau)}_{\mathsf{Com}}, c_\tau\right) \quad \text{and} \quad \mathsf{msk} = (\mathsf{pp}, \vec{r}, \vec{\sigma})$$
and gives pp to $\mathcal{A}$.

2. When algorithm $\mathcal{A}$ makes a key-generation query on $\mathsf{id} \in \mathcal{ID}_\lambda$, algorithm $\mathcal{B}$ computes $I \leftarrow \mathcal{H}(\mathsf{id})$, $\mathsf{sk}_{\mathsf{out}} \leftarrow \mathsf{PTE.Encode}(\mathsf{crs}_{\mathsf{PTE}}, \vec{r}_I; s_{\mathsf{out}})$ and $\mathsf{sk}_{\mathsf{aux}} \leftarrow \mathsf{SE.EncodeSF}(\mathsf{crs}_{\mathsf{SE}}, \mathsf{td}_{\mathsf{SE}}, \mathsf{id}, 0; s_{\mathsf{aux}})$, where s_{aux} is the encoding randomness. Algorithm $\mathcal{B}$ computes π by querying the proof oracle in the witness indistinguishability game with input (C, x, w_0, w_1) where
$$\begin{aligned} x &= (\mathsf{crs}^{(I)}_{\mathsf{Com}}, \vec{c}_I, c_\tau, \mathsf{sk}_{\mathsf{out}}, \mathsf{sk}_{\mathsf{aux}}, \mathsf{id}) \\ w_0 &= (\vec{r}_I, s_{\mathsf{out}}, \perp, \vec{\sigma}_I, \perp, \perp) \\ w_1 &= (\perp, \perp, s_{\mathsf{aux}}, \perp, \sigma_\tau, \mathsf{td}_{\mathsf{SE}})), \end{aligned}$$
where C is the circuit that computes $\mathcal{R}[\mathsf{crs}_{\mathsf{SE}}, \mathsf{crs}_{\mathsf{PTE}}, \mathsf{crs}^{(\tau)}_{\mathsf{Com}}]$ (Fig. 7). Algorithm $\mathcal{B}$ gives the identity secret key $\mathsf{sk}_{\mathsf{id}} = (\mathsf{sk}_{\mathsf{out}}, \mathsf{sk}_{\mathsf{aux}}, \pi)$ to algorithm $\mathcal{A}$.

3. When algorithm $\mathcal{A}$ outputs an efficiently-computable leakage function f, algorithm $\mathcal{B}$ replies with $\mathsf{leak} = f(\mathsf{msk})$. In the challenge phase, algorithm $\mathcal{A}$ outputs $\mathcal{J} \subseteq \mathcal{ID}_\lambda$ of size at least k, messages m_0, m_1, and a state st.

4. For all $\mathsf{id} \in \mathcal{J}$ and $i \in [T]$, algorithm $\mathcal{B}$ computes $I \leftarrow \mathcal{H}(\mathsf{id})$, samples $\beta \xleftarrow{\text{R}} \{0,1\}$, $\mathsf{ct}_{\mathsf{out}} \leftarrow \mathsf{PTE.Samp}(1^\lambda)$, $\mathsf{ct}_{\mathsf{aux}} \leftarrow \mathsf{SE.EncodeSF}(\mathsf{crs}_{\mathsf{SE}}, \mathsf{td}_{\mathsf{SE}}, \mathsf{id}, 1)$, and constructs the challenge ciphertext ct in iteration i of the procedure $\mathsf{AdvCheck}(\mathsf{id})$ as
$$\mathsf{ct} \leftarrow \mathsf{WE.Encrypt}\left(1^\lambda, m_\beta, \left(\vec{\mathsf{crs}}, C, \mathsf{crs}^{(I)}_{\mathsf{Com}}, \vec{c}_I, c_\tau, \mathsf{id}, \mathsf{ct}_{\mathsf{out}}, \mathsf{ct}_{\mathsf{aux}}\right)\right),$$
where C is the circuit that computes $\mathcal{R}[\mathsf{crs}_{\mathsf{SE}}, \mathsf{crs}_{\mathsf{PTE}}, \mathsf{crs}^{(\tau)}_{\mathsf{Com}}]$ (Fig. 7), and $\vec{\mathsf{crs}} = (\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{crs}_{\mathsf{SE}}, \mathsf{crs}_{\mathsf{PTE}})$. Algorithm $\mathcal{B}$ outputs the output of the experiment.

If the challenger constructs the proofs π using witness w_0, then algorithm $\mathcal{B}$ perfectly simulates Hyb_1. If the challenger constructs π using witness w_1, then algorithm $\mathcal{B}$ perfectly simulates Hyb_2. Thus, algorithm $\mathcal{B}$ breaks computational witness-indistinguishability with advantage δ. □

Lemma 5.9. *Suppose Π_{PTE} satisfies mode indistinguishability. Then, for all efficient and admissible adversaries $\mathcal{A}$ and inverse polynomial functions $\varepsilon = 1/\mathsf{poly}(\lambda)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,*
$$\Pr[\mathsf{Hyb}_3(\mathcal{A}, \varepsilon) = 1] \geq \Pr[\mathsf{Hyb}_2(\mathcal{A}, \varepsilon) = 1] - \mathsf{negl}(\lambda).$$

Proof. Suppose there exists an efficient adversary $\mathcal{A}$ that distinguishes Hyb_2 and Hyb_3 with non-negligible probability δ. We use $\mathcal{A}$ to construct an algorithm $\mathcal{B}$ that breaks mode indistinguishability:

1. At the beginning of the game, algorithm $\mathcal{B}$ gets $\mathsf{crs}_{\mathsf{PTE}}$ from the mode indistinguishability challenger. Algorithm $\mathcal{B}$ samples $(\mathsf{crs}_{\mathsf{SE}}, \mathsf{td}_{\mathsf{SE}}) \leftarrow \mathsf{SE.SetupSF}(1^\lambda)$ and $\mathsf{crs}_{\mathsf{NIZK}} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$. For all $i \in [N]$, algorithm $\mathcal{B}$ samples $r_i \xleftarrow{\mathrm{R}} \mathcal{X}_\lambda$, $(\mathsf{crs}_{\mathsf{Com}}^{(i)}, \mathsf{td}_{\mathsf{Com}}^{(i)}, c_i) \leftarrow \mathsf{Com.Setup}(1^\lambda, \mathsf{hide})$ and computes $\sigma_i \leftarrow \mathcal{S}_{\mathsf{open}}(\mathsf{td}_{\mathsf{Com}}^{(i)}, r_i)$. Algorithm $\mathcal{B}$ also samples $(\mathsf{crs}_{\mathsf{Com}}^{(\tau)}, \mathsf{td}_{\mathsf{Com}}^{(\tau)}, c_\tau) \leftarrow \mathsf{Com.Setup}(1^\lambda, \mathsf{hide})$ and computes $\sigma_\tau \leftarrow \mathcal{S}_{\mathsf{open}}(\mathsf{td}_{\mathsf{Com}}^{(\tau)}, \mathsf{td}_{\mathsf{SE}})$. Algorithm $\mathcal{B}$ sets

$$\mathsf{pp} = \left(\left\{\mathsf{crs}_{\mathsf{Com}}^{(i)}\right\}_{i \in [N]}, \vec{c}, \mathsf{crs}_{\mathsf{NIZK}}, \mathsf{crs}_{\mathsf{SE}}, \mathsf{crs}_{\mathsf{PTE}}, \mathsf{crs}_{\mathsf{Com}}^{(\tau)}, c_\tau\right) \quad \text{and} \quad \mathsf{msk} = (\mathsf{pp}, \vec{r}, \vec{\sigma})$$

   and gives pp to $\mathcal{A}$.

2. When algorithm $\mathcal{A}$ makes a key-generation query on $\mathsf{id} \in \mathcal{ID}_\lambda$, algorithm $\mathcal{B}$ computes $I \leftarrow \mathcal{H}(\mathsf{id})$ and makes an encoding query on $\vec{r}_I$ to the mode indistinguishability challenger to get $\mathsf{sk}_{\mathsf{out}}$. Algorithm $\mathcal{B}$ computes $\mathsf{sk}_{\mathsf{aux}} \leftarrow \mathsf{SE.EncodeSF}(\mathsf{crs}_{\mathsf{SE}}, \mathsf{td}_{\mathsf{SE}}, \mathsf{id}, 0; s_{\mathsf{aux}})$, where s_{aux} is the encoding randomness, and

$$\pi \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_{\mathsf{NIZK}}, C, (\mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, c_\tau, \mathsf{sk}_{\mathsf{out}}, \mathsf{sk}_{\mathsf{aux}}, \mathsf{id}), (\bot, \bot, s_{\mathsf{aux}}, \bot, \sigma_\tau, \mathsf{td})),$$

   where C is the circuit that computes $\mathcal{R}[\mathsf{crs}_{\mathsf{SE}}, \mathsf{crs}_{\mathsf{PTE}}, \mathsf{crs}_{\mathsf{Com}}^{(\tau)}]$ (Fig. 7). Algorithm $\mathcal{B}$ gives the identity secret key $\mathsf{sk}_{\mathsf{id}} = (\mathsf{sk}_{\mathsf{out}}, \mathsf{sk}_{\mathsf{aux}}, \pi)$ to algorithm $\mathcal{A}$.

3. When algorithm $\mathcal{A}$ outputs an efficiently-computable leakage function f, algorithm $\mathcal{B}$ replies with $\mathsf{leak} = f(\mathsf{msk})$. In the challenge phase, algorithm $\mathcal{A}$ outputs $\mathcal{J} \subseteq \mathcal{ID}_\lambda$ of size at least k, messages m_0, m_1, and a state st.

4. For all $\mathsf{id} \in \mathcal{J}$ and $i \in [T]$, algorithm $\mathcal{B}$ computes $I \leftarrow \mathcal{H}(\mathsf{id})$, samples $\beta \xleftarrow{\mathrm{R}} \{0,1\}$, $\mathsf{ct}_{\mathsf{out}} \leftarrow \mathsf{PTE.Samp}(1^\lambda)$, $\mathsf{ct}_{\mathsf{aux}} \leftarrow \mathsf{SE.EncodeSF}(\mathsf{crs}_{\mathsf{SE}}, \mathsf{td}_{\mathsf{SE}}, \mathsf{id}, 1)$, and constructs the challenge ciphertext ct in iteration i of the procedure $\mathsf{AdvCheck}(\mathsf{id})$ as

$$\mathsf{ct} \leftarrow \mathsf{WE.Encrypt}\left(1^\lambda, m_\beta, (\vec{\mathsf{crs}}, C, \mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, c_\tau, \mathsf{id}, \mathsf{ct}_{\mathsf{out}}, \mathsf{ct}_{\mathsf{aux}})\right),$$

   where C is the circuit that computes $\mathcal{R}[\mathsf{crs}_{\mathsf{SE}}, \mathsf{crs}_{\mathsf{PTE}}, \mathsf{crs}_{\mathsf{Com}}^{(\tau)}]$ (Fig. 7), and $\vec{\mathsf{crs}} = (\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{crs}_{\mathsf{SE}}, \mathsf{crs}_{\mathsf{PTE}})$. Algorithm $\mathcal{B}$ outputs the output of the experiment.

If the setup and encodings are in normal mode, algorithm $\mathcal{B}$ simulates $\mathsf{Hyb}_2(\mathcal{A}, \varepsilon)$. If the setup and encodings are in semi-functional mode, algorithm $\mathcal{B}$ simulates $\mathsf{Hyb}_3(\mathcal{A}, \varepsilon)$. Thus, algorithm $\mathcal{B}$ has advantage δ in the mode indistinguishability game. □

Lemma 5.10. *Suppose $\mathcal{H}$ is (k, α)-expanding, where $\alpha(\lambda) = \omega(\log \lambda)$ and $k \geq \frac{\ell}{\eta \alpha \ell_x}$ Suppose also that Π_{PTE} satisfies $(\omega(\log \lambda) \cdot \ell_x)$-trapdoor indistinguishability. Then, for all admissible adversaries $\mathcal{A}$ and inverse polynomial functions $\varepsilon = 1/\mathrm{poly}(\lambda)$, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all $\lambda \in \mathbb{N}$,*

$$\Pr[\mathsf{Hyb}_4(\mathcal{A}, \varepsilon) = 1] \geq \frac{\varepsilon}{4} \cdot \Pr[\mathsf{Hyb}_3(\mathcal{A}, \varepsilon) = 1] - \mathsf{negl}(\lambda).$$

Proof. Let $\delta = \Pr[\mathsf{Hyb}_3(\mathcal{A}, \varepsilon) = 1]$. First, note that lowering the threshold for WINS to $\frac{T}{2} + \frac{\varepsilon T}{16}$ and changing the condition for the experiment to output 1 from Eq. (5.2) to Eq. (5.3) only increases the probability that the experiment outputs 1. Other than these changes, the only difference between Hyb_3 and Hyb_4 is the distribution of $\mathsf{ct}_{\mathsf{out}}$:

- In Hyb_3, the challenger samples a fresh $\mathsf{ct}_{\mathsf{out}} \leftarrow \mathsf{PTE.Samp}(1^\lambda)$ on each of the T iterations of the advantage checker.
- In Hyb_4, the challenger samples $\mathsf{ct}_{\mathsf{out}} \leftarrow \mathsf{PTE.SampSF}(\mathsf{td}_{\mathsf{PTE}}, \vec{r}_{\mathcal{H}(\mathsf{id})})$ and reuses it across the T iterations of the advantage checker.

We now proceed via an averaging argument. Specifically, for each $i \in [T]$, let $X_i^{\mathsf{id}} \in \{0,1\}$ be the indicator random variable for whether algorithm $\mathcal{A}_2$'s output is correct on the i^{th} iteration of the advantage checker (i.e., $X_i^{\mathsf{id}} = 1$ if $\beta' = \beta$ in the i^{th} iteration of $\mathsf{AdvCheck}^{\mathcal{A}_2}(1^\lambda, 1^{1/\varepsilon}, \mathsf{id}, \mathsf{msk}, \mathsf{pp}, \mathsf{st}, \mathsf{leak})$) in Hyb_3 (when $\mathsf{ct}_{\mathsf{out}} \leftarrow \mathsf{PTE.Samp}(1^\lambda)$). Since the T iterations of the advantage-checker algorithm are *independent*, the random variables $X_1^{\mathsf{id}}, \ldots, X_T^{\mathsf{id}}$ are *identically* distributed. Let X^{id} be the distribution of each X_i^{id}. We now show the following claims:

Claim 5.11. *Let $\mathcal{A}$ be any admissible adversary and $(\mathsf{msk}, \mathsf{pp}, \mathsf{st}, \mathsf{leak}, \mathcal{J})$ be sampled as in $\mathsf{Hyb}_3(\mathcal{A}, \varepsilon)$. We have:*

$$\Pr_{(\mathsf{msk},\mathsf{pp},\mathsf{st},\mathsf{leak},\mathcal{J})}\left[\forall \mathsf{id} \in \mathcal{J} : \Pr[X^{\mathsf{id}} = 1] \geq 1/2 + \varepsilon/4\right] \geq \delta - \mathsf{negl}(\lambda),$$

where the inner probability is over the randomness of an iteration of $\mathsf{AdvCheck}^{\mathcal{A}_2}(1^\lambda, 1^{1/\varepsilon}, \mathsf{id}, \mathsf{msk}, \mathsf{pp}, \mathsf{st}, \mathsf{leak})$.

Proof. Suppose $\Pr[X^{\mathsf{id}} = 1] < 1/2 + \varepsilon/4$ for some $\mathsf{id} \in \mathcal{J}$. Since X^{id} is an indicator random variable, this means $\mathbb{E}[X^{\mathsf{id}}] < 1/2 + \varepsilon/4$. Let $\mathsf{WINS} = \sum_{i \in [T]} X_i^{\mathsf{id}}$. By linearity of expectation, this means

$$\mathbb{E}[\mathsf{WINS}] = \sum_{i \in [T]} \mathbb{E}[X^{\mathsf{id}}] \leq \frac{T}{2} + \frac{\varepsilon T}{4}.$$

By Hoeffding's inequality (Fact 2.1),

$$\Pr[\mathsf{WINS} - T/2 - \varepsilon T/2 > 0] \leq \Pr[|\mathsf{WINS} - T/2 - \varepsilon T/4| > \varepsilon T/4] \leq 2^{-\Omega(T\varepsilon^2/16)} = \nu(\lambda),$$

where ν is some negligible function. This means that when $\mathbb{E}[X^{\mathsf{id}}] < 1/2 + \varepsilon/4$ for some $\mathsf{id} \in \mathcal{J}$, the probability that $\mathsf{Hyb}_3(\mathcal{A}, \varepsilon) = 1$ is at most $\nu(\lambda)$. Thus, if $\Pr[\mathsf{Hyb}_3(\mathcal{A}, \varepsilon) = 1] = \delta$, then it must be the case that

$$\Pr[\forall \mathsf{id} \in \mathcal{J} : \mathbb{E}[X^{\mathsf{id}}] \geq 1/2 + \varepsilon/4] \geq \delta - \nu,$$

as required. ☐

Claim 5.12. *Let $E_{\mathsf{avg}}^{\mathsf{id}}$ be the event that fixing $\mathsf{ct}_{\mathsf{out}} = \mathsf{ct}_{\mathsf{out}}^{(\mathsf{id})}$ where $\mathsf{ct}_{\mathsf{out}}^{(\mathsf{id})} \leftarrow \mathsf{PTE.Samp}(1^\lambda)$ across all T iterations of $\mathsf{AdvCheck}(\mathsf{id})$ results in an output of 1 for $\mathsf{AdvCheck}(\mathsf{id})$ when the WINS threshold is $\frac{T}{2} + \frac{\varepsilon T}{16}$. Then, for all $\mathsf{id} \in \mathcal{J}$,*

$$\Pr\left[E_{\mathsf{avg}}^{\mathsf{id}}\right] \geq \frac{\varepsilon\delta}{4} - \mathsf{negl}(\lambda),$$

where the probability is taken over the randomness of $(\mathsf{msk}, \mathsf{pp}, \mathsf{st}, \mathsf{leak}, \mathcal{J})$ in Hyb_4.

Proof. Take any $\mathsf{id} \in \mathcal{J}$. First, we say that a ciphertext $\mathsf{ct}_{\mathsf{out}}$ in the support of $\mathsf{PTE.Samp}(1^\lambda)$ is "good" if $\Pr[X^{\mathsf{id}} = 1] \geq 1/2 + \varepsilon/8$, where the probability is conditioned on the fixed choice of $\mathsf{ct}_{\mathsf{out}}$. Suppose $\Pr[X^{\mathsf{id}} = 1] \geq 1/2 + \varepsilon/4$. Then, by an averaging argument,

$$\Pr[\mathsf{ct}_{\mathsf{out}} \text{ is good} : \mathsf{ct}_{\mathsf{out}} \leftarrow \mathsf{PTE.Samp}(1^\lambda)] > \frac{\varepsilon}{4}.$$

We now show that $\mathsf{AdvCheck}(\mathsf{id})$ outputs 1 with probability $1 - \mathsf{negl}(\lambda)$ whenever $\mathsf{ct}_{\mathsf{out}}$ is good and the WINS threshold is $T/2 + \varepsilon T/16$. When $\mathsf{ct}_{\mathsf{out}}$ is good,

$$\mathbb{E}[\mathsf{WINS}] = \sum_{i \in [T]} \mathbb{E}[X^{\mathsf{id}}] = \frac{T}{2} + \frac{\varepsilon T}{8}.$$

By Hoeffding's inequality (Fact 2.1),

$$\Pr[\mathsf{WINS} - T/2 < \varepsilon T/16] \leq \Pr[|\mathsf{WINS} - T/2 - \varepsilon T/8| > \varepsilon T/16] \leq 2^{-\Omega(T\varepsilon^2/256)} = \mathsf{negl}(\lambda). \tag{5.4}$$

Thus, whenever $\mathsf{ct}_{\mathsf{out}}$ is good and the WINS threshold is $T/2 + \varepsilon T/16$, we conclude that $\mathsf{AdvCheck}(\mathsf{id})$ outputs 1 with overwhelming probability. Next, by Claim 5.11, with probability at least $\delta - \mathsf{negl}(\lambda)$, we have that for all $\mathsf{id} \in \mathcal{J}$, $\Pr[X^{\mathsf{id}} = 1] \geq 1/2 + \varepsilon/4$. Thus, for every $\mathsf{id} \in \mathcal{J}$, with probability $\varepsilon\delta/4 - \mathsf{negl}(\lambda)$, it will be the case that $\Pr[X^{\mathsf{id}} = 1] \geq 1/2 + \varepsilon/4$ and $\mathsf{ct}_{\mathsf{out}} \leftarrow \mathsf{PTE.Samp}(1^\lambda)$ is good. When this is the case, Eq. (5.4) says that the probability $\mathsf{AdvCheck}(\mathsf{id})$ outputting 0 is negligible. Thus, we conclude that

$$\begin{aligned}\Pr\left[E^{\mathsf{id}}_{\mathsf{avg}}\right] &\geq \Pr\left[\mathsf{WINS} \geq \frac{T}{2} + \frac{\varepsilon T}{16} : \begin{array}{c}\Pr[X^{\mathsf{id}} = 1] \geq 1/2 + \varepsilon/4 \\ \mathsf{ct}_{\mathsf{out}} \leftarrow \mathsf{PTE.Samp}(1^\lambda) \text{ is good}\end{array}\right] - \mathsf{negl}(\lambda) \\ &\geq \frac{\varepsilon\delta}{4} - \mathsf{negl}(\lambda).\end{aligned}$$

The claim follows. □

Proof of Lemma 5.10. We now return to the proof of Lemma 5.10. Let $E^{\mathsf{id}}_{\mathsf{td}}$ be the event that $E^{\mathsf{id}}_{\mathsf{avg}}$ occurs when $\mathsf{ct}_{\mathsf{out}} \leftarrow \mathsf{PTE.SampSF}(\mathsf{td}_{\mathsf{PTE}}, \vec{r}_{\mathcal{H}(\mathsf{id})})$; namely, when $\mathsf{ct}_{\mathsf{out}}$ is sampled according to the specification of Hyb_4. Note that when $E^{\mathsf{id}}_{\mathsf{td}}$ occurs for *any* $\mathsf{id} \in \mathcal{J}$, then $\mathsf{Hyb}_4(\mathcal{A}, \varepsilon)$ outputs 1. To argue this, we first appeal to trapdoor indistinguishability to show that there exists an identity $\mathsf{id} \in \mathcal{J}$ such that

$$\Pr[E^{\mathsf{id}}_{\mathsf{td}}] \geq \Pr[E^{\mathsf{id}}_{\mathsf{avg}}] - \mathsf{negl}(\lambda).$$

To do so, we lower bound the entropy of the bits $\{\vec{r}_{\mathcal{H}(\mathsf{id})}\}_{\mathsf{id} \in \mathcal{J}}$ conditioned on the tuple $(\mathsf{pp}, \mathcal{Q}, \mathsf{leak}, \mathcal{J})$, where $\mathcal{Q}$ is the adversary's view from the key-generation queries. Fix any k identities $\mathsf{id}_1, \ldots, \mathsf{id}_k \in \mathcal{J}$. Since $\mathcal{H}$ is expanding, $\mathcal{H}(\mathsf{id}_1), \ldots, \mathcal{H}(\mathsf{id}_k)$ contains at least αk distinct indices of $[N]$, which correspond to $\alpha k \cdot \ell_x$ total bits of msk. By construction, the bits $\vec{r}$ are independent of $(\mathsf{pp}, \mathcal{Q})$ in Hyb_4 and $\eta < 1$. Thus, we can appeal to Lemma 2.2 to get the following:

$$\begin{aligned}\mathbf{H}_\infty(\{\vec{r}_{\mathcal{H}(\mathsf{id}_i)}\}_{i \in [k]} \mid \mathsf{pp}, \mathcal{Q}, \mathsf{leak}) &\geq \mathbf{H}_\infty(\{\vec{r}_{\mathcal{H}(\mathsf{id}_i)}\}_{i \in [k]} \mid \mathsf{pp}, \mathcal{Q}) - |\mathsf{leak}| \\ &\geq \mathbf{H}_\infty(\{\vec{r}_{\mathcal{H}(\mathsf{id}_i)}\}_{i \in [k]}) - |\mathsf{leak}| \\ &= \alpha k \ell_x - \eta \alpha k \ell_x \\ &= (1 - \eta)\alpha k \ell_x.\end{aligned}$$

By Lemma 2.2, with probability $1 - 2^{-\omega(\log \lambda)} = 1 - \mathsf{negl}(\lambda)$ over the fixed choice of $(\mathsf{pp}, \mathcal{Q}, \mathsf{leak})$, we have

$$\mathbf{H}_\infty(\{\vec{r}_{\mathcal{H}(\mathsf{id}_i)}\}_{i \in [k]}) \geq (1 - \eta)\alpha k \ell_x - \omega(\log \lambda).$$

Moreover, by Lemma 2.3, there exists random variable $\mathcal{D}_{[k]}$ over $[k]$ such that

$$\mathbf{H}_\infty(\vec{r}_J \mid \mathcal{D}_{[k]}) \geq \frac{(1 - \eta)\alpha k \ell_x - \omega(\log \lambda)}{k} - \log(k),$$

where $J = \mathcal{H}(\mathsf{id}_{\mathcal{D}_{[k]}})$. By Lemma 2.2, we have with probability $1 - 2^{-\omega(\log\lambda)} = 1 - \mathsf{negl}(\lambda)$ over the choice of $i \leftarrow \mathcal{D}_{[k]}$,

$$\mathbf{H}_\infty(\vec{r}_{\mathcal{H}(\mathsf{id}_i)} \mid \mathcal{D}_{[k]} = i) \geq \mathbf{H}_\infty(r_J \mid \mathcal{D}_{[k]}) - \omega(\log\lambda).$$

Thus, with overwhelming probability over $\mathsf{id}^* \leftarrow \mathsf{id}_{\mathcal{D}_{[k]}}$, we have $\mathbf{H}_\infty(\vec{r}_{\mathcal{H}(\mathsf{id}^*)}) = (1-\eta)\alpha\ell_x - \omega(\log\lambda)$ for fixed $(\mathsf{pp}, \mathcal{Q}, \mathcal{J}, \mathsf{leak})$. Since $\alpha\ell_x = \omega(\log\lambda)\cdot\ell_x$ by assumption, we can appeal to $(\omega(\log\lambda)\cdot\ell_x)$-trapdoor indistinguishability with overwhelming probability in the id^* copy of the game. This means the distribution of $\mathsf{ct}_{\mathsf{out}} \leftarrow \mathsf{PTE.SampSF}(\mathsf{td}_{\mathsf{PTE}}, \vec{r}_{\mathcal{H}(\mathsf{id}^*)})$ is statistically close to the distribution of $\mathsf{ct}_{\mathsf{out}} \leftarrow \mathsf{PTE.Samp}(1^\lambda)$. Correspondingly, this means

$$\Pr[E_{\mathsf{td}}^{\mathsf{id}^*}] \geq \Pr[E_{\mathsf{avg}}^{\mathsf{id}^*}] - \mathsf{negl}(\lambda).$$

By Claim 5.12, $\Pr[E_{\mathsf{avg}}^{\mathsf{id}^*}] \geq \varepsilon\delta/4 - \mathsf{negl}(\lambda)$. Finally, by definition of δ,

$$\Pr[\mathsf{Hyb}_4(\mathcal{A},\varepsilon) = 1] \geq \Pr[E_{\mathsf{td}}^{\mathsf{id}^*}] \geq \frac{\varepsilon\delta}{4} - \mathsf{negl}(\lambda) = \frac{\varepsilon}{4}\cdot\Pr[\mathsf{Hyb}_3(\mathcal{A},\varepsilon) = 1] - \mathsf{negl}(\lambda). \qquad \square$$

Lemma 5.13. *Suppose Π_{Com} satisfies mode indistinguishability. Then, for all efficient and admissible adversaries $\mathcal{A}$ and inverse polynomial functions $\varepsilon = 1/\mathsf{poly}(\lambda)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,*

$$\Pr[\mathsf{Hyb}_5(\mathcal{A},\varepsilon) = 1] \geq \Pr[\mathsf{Hyb}_4(\mathcal{A},\varepsilon) = 1] - \mathsf{negl}(\lambda).$$

Proof. We start by defining a sequence of intermediate hybrid experiments:

- $\mathsf{Hyb}_{4,0}$: Same as Hyb_4. In particular, the components $(\mathsf{crs}_{\mathsf{Com}}^{(i)}, c_i, \sigma_i)_{i\in[N]}$ in the setup phase are sampled as
$$\left(\mathsf{crs}_{\mathsf{Com}}^{(i)}, \mathsf{td}_{\mathsf{Com}}^{(i)}, c_i\right) \leftarrow \mathsf{Com.Setup}(1^\lambda, \mathsf{hide}), \sigma_i \leftarrow \mathcal{S}_{\mathsf{open}}(\mathsf{td}_{\mathsf{Com}}^{(i)}, r_i).$$
Additionally, $\left(\mathsf{crs}_{\mathsf{Com}}^{(\tau)}, \mathsf{td}_{\mathsf{Com}}^{(\tau)}, c_\tau\right) \leftarrow \mathsf{Com.Setup}(1^\lambda, \mathsf{hide}), \sigma_\tau \leftarrow \mathcal{S}_{\mathsf{open}}(\mathsf{td}_{\mathsf{Com}}^{(\tau)}, \mathsf{td}_{\mathsf{SE}})$.
- $\mathsf{Hyb}_{4,i}$: Same as $\mathsf{Hyb}_{4,0}$, except for all $j \leq i$ the challenger samples the components $(\mathsf{crs}_{\mathsf{Com}}^{(j)}, c_j, \sigma_j)$ as
$$\mathsf{crs}_{\mathsf{Com}}^{(j)} \leftarrow \mathsf{Com.Setup}(1^\lambda, \mathsf{bind}), (c_j, \sigma_j) \leftarrow \mathsf{Com.Commit}(\mathsf{crs}_{\mathsf{Com}}^{(j)}, r_j)$$
- $\mathsf{Hyb}_{4,N+1}$: Same as $\mathsf{Hyb}_{4,N}$ except the challenger samples $(\mathsf{crs}_{\mathsf{Com}}^{(\tau)}, c_\tau, \sigma_\tau)$ as
$$\mathsf{crs}_{\mathsf{Com}}^{(\tau)} \leftarrow \mathsf{Com.Setup}(1^\lambda, \mathsf{bind}), (c_\tau, \sigma_\tau) \leftarrow \mathsf{Com.Commit}(\mathsf{crs}_{\mathsf{Com}}, \mathsf{td}_{\mathsf{SE}}).$$
This is the same as Hyb_5.

We now appeal to mode indistinguishability of Π_{Com} to show that for all $i \in [N+1]$, $\mathsf{Hyb}_{4,i-1}$ and $\mathsf{Hyb}_{4,i}$ are statistically indistinguishable. We omit the proof of indistinguishability for $\mathsf{Hyb}_{4,N}$ and $\mathsf{Hyb}_{4,N+1}$ since it is analogous.

Claim 5.14. *Suppose the conditions in Lemma 5.13 hold. Then, for all $i \in [N+1]$, admissible adversaries $\mathcal{A}$, and inverse polynomial functions $\varepsilon = 1/\mathsf{poly}(\lambda)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,*

$$\Pr[\mathsf{Hyb}_{4,i}(\mathcal{A},\varepsilon) = 1] \geq \Pr[\mathsf{Hyb}_{4,i-1}(\mathcal{A},\varepsilon) = 1] - \mathsf{negl}(\lambda).$$

Proof. Suppose there exists an adversary $\mathcal{A}$ that distinguishes $\mathsf{Hyb}_{4,i-1}$ and $\mathsf{Hyb}_{4,i}$ for some $i \in [N]$ with non-negligible probability δ. We use $\mathcal{A}$ to construct an algorithm $\mathcal{B}$ that breaks mode indistinguishability:

1. At the beginning of the game, algorithm $\mathcal{B}$ gets $\mathsf{crs}_{\mathsf{Com}}^{(i)}$ from the mode indistinguishability challenger. Algorithm $\mathcal{B}$ samples $(\mathsf{crs}_{\mathsf{SE}}, \mathsf{td}_{\mathsf{SE}}) \leftarrow \mathsf{SE.SetupSF}(1^\lambda)$, $(\mathsf{crs}_{\mathsf{PTE}}, \mathsf{td}_{\mathsf{PTE}}) \leftarrow \mathsf{PTE.SetupSF}(1^\lambda)$, and $\mathsf{crs}_{\mathsf{NIZK}} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$. For all $j \in [N]$, algorithm $\mathcal{B}$ samples $r_j \xleftarrow{\mathrm{R}} \mathcal{X}_\lambda$. For $j < i$, algorithm $\mathcal{B}$ computes regular commitments and openings
$$\mathsf{crs}_{\mathsf{Com}}^{(j)} \leftarrow \mathsf{Com.Setup}(1^\lambda, \mathsf{bind}), (c_j, \sigma_j) \leftarrow \mathsf{Com.Commit}(\mathsf{crs}_{\mathsf{Com}}^{(j)}, r_j).$$
Algorithm $\mathcal{B}$ submits r_i to the mode indistinguishability challenger to get (c_i, σ_i). For $j > i$, algorithm $\mathcal{B}$ computes simulated commitments and openings
$$\left(\mathsf{crs}_{\mathsf{Com}}^{(j)}, \mathsf{td}_{\mathsf{Com}}^{(j)}, c_j\right) \leftarrow \mathsf{Com.Setup}(1^\lambda, \mathsf{hide}), \sigma_j \leftarrow \mathcal{S}_{\mathsf{open}}(\mathsf{td}_{\mathsf{Com}}^{(j)}, r_j).$$
Algorithm $\mathcal{B}$ also samples $\left(\mathsf{crs}_{\mathsf{Com}}^{(\tau)}, \mathsf{td}_{\mathsf{Com}}^{(\tau)}, c_\tau\right) \leftarrow \mathsf{Com.Setup}(1^\lambda, \mathsf{hide})$ and computes the simulated opening $\sigma_\tau \leftarrow \mathcal{S}_{\mathsf{open}}(\mathsf{td}_{\mathsf{Com}}^{(\tau)}, \mathsf{td}_{\mathsf{SE}})$. Algorithm $\mathcal{B}$ sets
$$\mathsf{pp} = \left(\left\{\mathsf{crs}_{\mathsf{Com}}^{(i)}\right\}_{i \in [N]}, \vec{c}, \mathsf{crs}_{\mathsf{NIZK}}, \mathsf{crs}_{\mathsf{SE}}, \mathsf{crs}_{\mathsf{PTE}}, \mathsf{crs}_{\mathsf{Com}}^{(\tau)}, c_\tau\right) \quad \text{and} \quad \mathsf{msk} = (\mathsf{pp}, \vec{r}, \vec{\sigma})$$
and gives pp to $\mathcal{A}$.

2. When algorithm $\mathcal{A}$ makes a key-generation query on $\mathsf{id} \in \mathcal{ID}_\lambda$, algorithm $\mathcal{B}$ computes $I \leftarrow \mathcal{H}(\mathsf{id})$, $\mathsf{sk}_{\mathsf{out}} \leftarrow \mathsf{PTE.EncodeSF}(\mathsf{crs}_{\mathsf{PTE}})$, $\mathsf{sk}_{\mathsf{aux}} \leftarrow \mathsf{SE.EncodeSF}(\mathsf{crs}_{\mathsf{SE}}, \mathsf{td}_{\mathsf{SE}}, \mathsf{id}, 0; s_{\mathsf{aux}})$, where s_{aux} is the encoding randomness, and
$$\pi \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_{\mathsf{NIZK}}, C, (\mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, c_\tau, \mathsf{sk}_{\mathsf{out}}, \mathsf{sk}_{\mathsf{aux}}, \mathsf{id}), (\bot, \bot, s_{\mathsf{aux}}, \bot, \sigma_\tau, \mathsf{td})),$$
where C is the circuit that computes $\mathcal{R}[\mathsf{crs}_{\mathsf{SE}}, \mathsf{crs}_{\mathsf{PTE}}, \mathsf{crs}_{\mathsf{Com}}^{(\tau)}]$ (Fig. 7). Algorithm $\mathcal{B}$ gives the identity secret key $\mathsf{sk}_{\mathsf{id}} = (\mathsf{sk}_{\mathsf{out}}, \mathsf{sk}_{\mathsf{aux}}, \pi)$ to algorithm $\mathcal{A}$.

3. When algorithm $\mathcal{A}$ outputs an efficiently-computable leakage function f, algorithm $\mathcal{B}$ replies with $\mathsf{leak} = f(\mathsf{msk})$. In the challenge phase, algorithm $\mathcal{A}$ outputs $\mathcal{J} \subseteq \mathcal{ID}_\lambda$ of size at least k, messages m_0, m_1, and a state st.

4. For all $\mathsf{id} \in \mathcal{J}$, algorithm $\mathcal{B}$ computes $I \leftarrow \mathcal{H}(\mathsf{id})$ and $\mathsf{ct}_{\mathsf{out}} \leftarrow \mathsf{PTE.SampSF}(\mathsf{td}_{\mathsf{PTE}}, \vec{r}_I)$. Then, for each $i \in [T]$, algorithm $\mathcal{B}$ samples $\beta \xleftarrow{\mathrm{R}} \{0,1\}$, $\mathsf{ct}_{\mathsf{aux}} \leftarrow \mathsf{SE.EncodeSF}(\mathsf{crs}_{\mathsf{SE}}, \mathsf{td}_{\mathsf{SE}}, \mathsf{id}, 1)$, and constructs the challenge ciphertext ct according to the specification of iteration i of the procedure $\mathsf{AdvCheck}(\mathsf{id})$ in Hyb_4 and Hyb_5:
$$\mathsf{ct} \leftarrow \mathsf{WE.Encrypt}\left(1^\lambda, m_\beta, \left(\vec{\mathsf{crs}}, C, \mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, c_\tau, \mathsf{id}, \mathsf{ct}_{\mathsf{out}}, \mathsf{ct}_{\mathsf{aux}}\right)\right),$$
where C is the circuit that computes $\mathcal{R}[\mathsf{crs}_{\mathsf{SE}}, \mathsf{crs}_{\mathsf{PTE}}, \mathsf{crs}_{\mathsf{Com}}^{(\tau)}]$ (Fig. 7), and $\vec{\mathsf{crs}} = (\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{crs}_{\mathsf{SE}}, \mathsf{crs}_{\mathsf{PTE}})$. Algorithm $\mathcal{B}$ outputs the output of the experiment.

If the challenger samples the CRS and the commitments in hiding mode, then algorithm $\mathcal{B}$ simulates an execution of $\mathsf{Hyb}_{4,i-1}(\mathcal{A}, \varepsilon)$. Conversely, if the challenger samples the CRS and the commitments in binding mode, algorithm $\mathcal{B}$ simulates an execution of $\mathsf{Hyb}_{4,i}(\mathcal{A}, \varepsilon)$. Thus, algorithm $\mathcal{B}$ breaks mode indistinguishability with advantage δ. □

The lemma now follows from Claim 5.14 and a standard hybrid argument. □

Lemma 5.15. *Suppose Π_{WE} satisfies semantic security, Π_{SE} and Π_{PTE} satisfy tester correctness, Π_{NIZK} satisfies statistical soundness, and Π_{Com} satisfies statistical binding in binding mode. Then, for all efficient and admissible adversaries $\mathcal{A}$ and inverse polynomial functions $\varepsilon = 1/\mathsf{poly}(\lambda)$, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all $\lambda \in \mathbb{N}$,*

$$\Pr[\mathsf{Hyb}_6(\mathcal{A}, \varepsilon) = 1] \geq \Pr[\mathsf{Hyb}_5(\mathcal{A}, \varepsilon) = 1] - \mathsf{negl}(\lambda).$$

Proof. We define a sequence of intermediate hybrids:

- $\mathsf{Hyb}_{5,1,0}$: Same as Hyb_5. Notably, for $\mathsf{id} \in \mathcal{J}$ and $i \in [T]$, the challenger samples $\beta \xleftarrow{\mathrm{R}} \{0,1\}, I \leftarrow \mathcal{H}(\mathsf{id}), \mathsf{ct}_{\mathsf{out}} \leftarrow \mathsf{PTE.SampSF}(\mathsf{td}_{\mathsf{PTE}}, \vec{r}_I), \mathsf{ct}_{\mathsf{aux}} \leftarrow \mathsf{SE.EncodeSF}(\mathsf{crs}_{\mathsf{SE}}, \mathsf{td}_{\mathsf{SE}}, \mathsf{id}, 1)$, and computes the challenge ciphertext in iteration i of $\mathsf{AdvCheck}(\mathsf{id})$ as

$$\mathsf{ct} \leftarrow \mathsf{WE.Encrypt}\big(1^\lambda, m_\beta, \big(\vec{\mathsf{crs}}, C, \mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, c_\tau, \mathsf{id}, \mathsf{ct}_{\mathsf{out}}, \mathsf{ct}_{\mathsf{aux}}\big)\big),$$

  where C is the circuit that computes $\mathcal{R}[\mathsf{crs}_{\mathsf{SE}}, \mathsf{crs}_{\mathsf{PTE}}, \mathsf{crs}_{\mathsf{Com}}^{(\tau)}]$ (Fig. 7), and $\vec{\mathsf{crs}} = (\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{crs}_{\mathsf{SE}}, \mathsf{crs}_{\mathsf{PTE}})$.

- $\mathsf{Hyb}_{5,i,j}$: Same as Hyb_5 except for all (i', j') such that $i' < i$ or $i' = i, j' \leq j$, the challenger samples $I \leftarrow \mathcal{H}(\mathcal{J}[i']), \mathsf{ct}_{\mathsf{out}} \leftarrow \mathsf{PTE.SampSF}(\mathsf{td}_{\mathsf{PTE}}, \vec{r}_I), \mathsf{ct}_{\mathsf{aux}} \leftarrow \mathsf{SE.EncodeSF}(\mathsf{crs}_{\mathsf{SE}}, \mathsf{td}_{\mathsf{SE}}, \mathcal{J}[i'], 1)$, and computes the challenge ciphertext in the $(j')^{\text{th}}$ execution of $\mathsf{Exp}^{\mathcal{J}[i']}$ in $\mathsf{AdvCheck}$ as

$$\mathsf{ct} \leftarrow \mathsf{WE.Encrypt}(1^\lambda, 0, (\vec{\mathsf{crs}}, C, \mathsf{crs}_{\mathsf{Com}}^{(I)}, c_I, c_\tau, \mathcal{J}[i'], \mathsf{ct}_{\mathsf{out}}, \mathsf{ct}_{\mathsf{aux}})),$$

  where C is the circuit that computes $\mathcal{R}[\mathsf{crs}_{\mathsf{SE}}, \mathsf{crs}_{\mathsf{PTE}}, \mathsf{crs}_{\mathsf{Com}}^{(\tau)}]$ (Fig. 7), and $\vec{\mathsf{crs}} = (\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{crs}_{\mathsf{SE}}, \mathsf{crs}_{\mathsf{PTE}})$. Note that $\mathsf{Hyb}_{5,k,T}$ is the same as Hyb_6 and that $\mathsf{Hyb}_{5,i,T}$ is the same as $\mathsf{Hyb}_{5,i+1,0}$ for $i \in [k-1]$.

We now appeal to semantic security of Π_{WE} to show that $\mathsf{Hyb}_{5,i,j}$ and $\mathsf{Hyb}_{5,i,j-1}$ are computationally indistinguishable for all $i \in [k], j \in [T]$.

Claim 5.16. *Suppose the conditions in Lemma 5.15 hold. Then for all $i \in [k], j \in [T]$, efficient and admissible adversaries $\mathcal{A}$, and inverse polynomial functions $\varepsilon = 1/\mathsf{poly}(\lambda)$, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all $\lambda \in \mathbb{N}$,*

$$\Pr[\mathsf{Hyb}_{5,i,j}(\mathcal{A}, \varepsilon) = 1] \geq \Pr[\mathsf{Hyb}_{5,i,j-1}(\mathcal{A}, \varepsilon) = 1] - \mathsf{negl}(\lambda).$$

Proof. Suppose there exists an efficient adversary $\mathcal{A}$ that distinguishes $\mathsf{Hyb}_{5,i,j}$ and $\mathsf{Hyb}_{5,i,j-1}$ with non-negligible advantage δ. We use $\mathcal{A}$ to construct an algorithm $\mathcal{B}$ that breaks semantic security of Π_{WE}:

1. Algorithm $\mathcal{B}$ runs the setup through challenge phases as in Hyb_5 with $\mathcal{A}$:

   (a) Algorithm $\mathcal{B}$ samples $(\mathsf{crs}_{\mathsf{SE}}, \mathsf{td}_{\mathsf{SE}}) \leftarrow \mathsf{SE.SetupSF}(1^\lambda), (\mathsf{crs}_{\mathsf{PTE}}, \mathsf{td}_{\mathsf{PTE}}) \leftarrow \mathsf{PTE.SetupSF}(1^\lambda)$, and $\mathsf{crs}_{\mathsf{NIZK}} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$. For all $i \in [N]$, algorithm $\mathcal{B}$ samples components $r_i \xleftarrow{\mathrm{R}} \mathcal{X}_\lambda$, $\mathsf{crs}_{\mathsf{Com}}^{(i)} \leftarrow \mathsf{Com.Setup}(1^\lambda, \mathsf{bind})$ and $(c_i, \sigma_i) \leftarrow \mathsf{Com.Commit}(\mathsf{crs}_{\mathsf{Com}}^{(i)}, r_i)$. Algorithm $\mathcal{B}$ also samples $\mathsf{crs}_{\mathsf{Com}}^{(\tau)} \leftarrow \mathsf{Com.Setup}(1^\lambda, \mathsf{bind})$ and $(c_\tau, \sigma_\tau) \leftarrow \mathsf{Com.Commit}(\mathsf{crs}_{\mathsf{Com}}, \mathsf{td}_{\mathsf{SE}})$. Algorithm $\mathcal{B}$ sets

$$\mathsf{pp} = \left(\{\mathsf{crs}_{\mathsf{Com}}^{(i)}\}_{i \in [N]}\,,\ \vec{c}\,,\ \mathsf{crs}_{\mathsf{NIZK}}\,,\ \mathsf{crs}_{\mathsf{SE}}\,,\ \mathsf{crs}_{\mathsf{PTE}}\,,\ \mathsf{crs}_{\mathsf{Com}}^{(\tau)}\,,\ c_\tau\right) \quad \text{and} \quad \mathsf{msk} = (\mathsf{pp}, \vec{r}, \vec{\sigma})$$

   and gives pp to $\mathcal{A}$.

(b) When algorithm $\mathcal{A}$ makes a key-generation query on id $\in \mathcal{ID}_\lambda$, algorithm $\mathcal{B}$ computes $I \leftarrow \mathcal{H}(\mathsf{id})$, $\mathsf{sk}_{\mathsf{out}} \leftarrow \mathsf{PTE.EncodeSF}(\mathsf{crs}_{\mathsf{PTE}})$, $\mathsf{sk}_{\mathsf{aux}} \leftarrow \mathsf{SE.EncodeSF}(\mathsf{crs}_{\mathsf{SE}}, \mathsf{td}_{\mathsf{SE}}, \mathsf{id}, 0; s_{\mathsf{aux}})$, where s_{aux} is the encoding randomness, and

$$\pi \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_{\mathsf{NIZK}}, C, (\mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, c_\tau, \mathsf{sk}_{\mathsf{out}}, \mathsf{sk}_{\mathsf{aux}}, \mathsf{id}), (\perp, \perp, s_{\mathsf{aux}}, \perp, \sigma_\tau, \mathsf{td})),$$

where C is the circuit that computes $\mathcal{R}[\mathsf{crs}_{\mathsf{SE}}, \mathsf{crs}_{\mathsf{PTE}}, \mathsf{crs}_{\mathsf{Com}}^{(\tau)}]$ (Fig. 7). Algorithm $\mathcal{B}$ gives the identity secret key $\mathsf{sk}_{\mathsf{id}} = (\mathsf{sk}_{\mathsf{out}}, \mathsf{sk}_{\mathsf{aux}}, \pi)$ to algorithm $\mathcal{A}$.

(c) When algorithm $\mathcal{A}$ outputs an efficiently-computable leakage function f, algorithm $\mathcal{B}$ replies with leak $= f(\mathsf{msk})$. In the challenge phase, algorithm $\mathcal{A}$ outputs $\mathcal{J} \subseteq \mathcal{ID}_\lambda$ of size at least k, messages m_0, m_1, and a state st.

2. For all (i', j') such that $i' < i$ or $i' = i, j' < j$, algorithm $\mathcal{B}$ computes $I \leftarrow \mathcal{H}(\mathcal{J}[i']), \mathsf{ct}_{\mathsf{out}} \leftarrow \mathsf{PTE.SampSF}(\mathsf{td}_{\mathsf{PTE}}, \vec{r}_I), \mathsf{ct}_{\mathsf{aux}} \leftarrow \mathsf{SE.EncodeSF}(\mathsf{crs}_{\mathsf{SE}}, \mathsf{td}_{\mathsf{SE}}, \mathcal{J}[i'], 1)$, and constructs the challenge ciphertext in the j'^{th} execution of $\mathsf{Exp}^{\mathcal{J}[i']}$ in AdvCheck as

$$\mathsf{ct} \leftarrow \mathsf{WE.Encrypt}(1^\lambda, 0, (\vec{\mathsf{crs}}, C, \mathsf{crs}_{\mathsf{Com}}^{(I)}, c_I, c_\tau, \mathcal{J}[i'], \mathsf{ct}_{\mathsf{out}}, \mathsf{ct}_{\mathsf{aux}})),$$

where C is the circuit that computes $\mathcal{R}[\mathsf{crs}_{\mathsf{SE}}, \mathsf{crs}_{\mathsf{PTE}}, \mathsf{crs}_{\mathsf{Com}}^{(\tau)}]$ (Fig. 7), and $\vec{\mathsf{crs}} = (\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{crs}_{\mathsf{SE}}, \mathsf{crs}_{\mathsf{PTE}})$.

3. For the j^{th} execution of $\mathsf{Exp}^{\mathcal{J}[i]}$, algorithm $\mathcal{B}$ computes $I \leftarrow \mathcal{H}(\mathcal{J}[i]), \mathsf{ct}_{\mathsf{out}} \leftarrow \mathsf{PTE.SampSF}(\mathsf{td}_{\mathsf{PTE}}, \vec{r}_I)$, $\mathsf{ct}_{\mathsf{aux}} \leftarrow \mathsf{SE.EncodeSF}(\mathsf{crs}_{\mathsf{SE}}, \mathsf{td}_{\mathsf{SE}}, \mathcal{J}[i], 1)$, and sets

$$x = (\vec{\mathsf{crs}}, C, \mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, c_\tau, \mathcal{J}[i], \mathsf{ct}_{\mathsf{out}}, \mathsf{ct}_{\mathsf{aux}}),$$

where C is the circuit that computes $\mathcal{R}[\mathsf{crs}_{\mathsf{SE}}, \mathsf{crs}_{\mathsf{PTE}}, \mathsf{crs}_{\mathsf{Com}}^{(\tau)}]$ (Fig. 7), and $\vec{\mathsf{crs}} = (\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{crs}_{\mathsf{SE}}, \mathsf{crs}_{\mathsf{PTE}})$. Algorithm $\mathcal{B}$ samples $\beta \xleftarrow{\text{R}} \{0,1\}$, sends $(x, m_\beta, 0)$ to the WE challenger, and uses the response ct from the WE challenger as the challenge ciphertext in this execution.

4. In the remaining executions of $\mathsf{Exp}^{\mathcal{J}[i']}$ for $i' \geq i$, algorithm $\mathcal{B}$ samples components $\beta \xleftarrow{\text{R}} \{0,1\}, I \leftarrow \mathcal{H}(\mathcal{J}[i']), \mathsf{ct}_{\mathsf{out}} \leftarrow \mathsf{PTE.SampSF}(\mathsf{td}_{\mathsf{PTE}}, \vec{r}_I), \mathsf{ct}_{\mathsf{aux}} \leftarrow \mathsf{SE.EncodeSF}(\mathsf{crs}_{\mathsf{SE}}, \mathsf{td}_{\mathsf{SE}}, \mathcal{J}[i'], 1)$, and constructs the challenge ciphertext as

$$\mathsf{ct} \leftarrow \mathsf{WE.Encrypt}(1^\lambda, m_\beta, (\vec{\mathsf{crs}}, C, \mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, c_\tau, \mathcal{J}[i'], \mathsf{ct}_{\mathsf{out}}, \mathsf{ct}_{\mathsf{aux}})),$$

where C is the circuit that computes $\mathcal{R}[\mathsf{crs}_{\mathsf{SE}}, \mathsf{crs}_{\mathsf{PTE}}, \mathsf{crs}_{\mathsf{Com}}^{(\tau)}]$ (Fig. 7), and $\vec{\mathsf{crs}} = (\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{crs}_{\mathsf{SE}}, \mathsf{crs}_{\mathsf{PTE}})$. Algorithm $\mathcal{B}$ outputs the output of the experiment.

If ct from the WE challenger is constructed using m_β, algorithm $\mathcal{B}$ simulates $\mathsf{Hyb}_{5,i,j-1}(\mathcal{A}, \varepsilon)$. If ct from the WE challenger is constructed using 0, algorithm $\mathcal{B}$ simulates $\mathsf{Hyb}_{5,i,j}(\mathcal{A}, \varepsilon)$. We now show that for $\mathsf{id} = \mathcal{J}[i]$ the statement

$$x = (\vec{\mathsf{crs}}, C, \mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, c_\tau, \mathsf{id}, \mathsf{ct}_{\mathsf{out}}, \mathsf{ct}_{\mathsf{aux}})$$

sampled in Step 3 of the above reduction is a false statement with overwhelming probability. In particular, we show that for candidate witness $w = (\mathsf{sk}_{\mathsf{out}}, \mathsf{sk}_{\mathsf{aux}}, \pi)$

$$\mathsf{NIZK.Verify}(\mathsf{crs}_{\mathsf{NIZK}}, C, (\mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, c_\tau, \mathsf{sk}_{\mathsf{out}}, \mathsf{sk}_{\mathsf{aux}}, \mathsf{id}), \pi) = 1 \implies \begin{array}{c} \mathsf{PTE.Test}(\mathsf{crs}_{\mathsf{PTE}}, \mathsf{sk}_{\mathsf{out}}, \mathsf{ct}_{\mathsf{out}}) = 1 \\ \text{or} \\ \mathsf{SE.Test}(\mathsf{crs}_{\mathsf{SE}}, \mathsf{sk}_{\mathsf{aux}}, \mathsf{ct}_{\mathsf{aux}}) = 1. \end{array}$$

with $1 - \mathsf{negl}(\lambda)$ probability over the choice of pp, where $I = \mathcal{H}(\mathsf{id})$:

- First, $\vec{c}$ is an honestly-generated commitment to $\vec{r}$ and c_τ is an honestly-generated commitment to $\mathsf{td}_{\mathsf{SE}}$. Since Π_{Com} is statistically binding, with overwhelming probability over the choice of $\mathsf{crs}_{\mathsf{Com}}^{(I)}$ and $\mathsf{crs}_{\mathsf{Com}}^{(\tau)}$, the only valid openings for $\vec{c}_I$ is to $\vec{r}_I = \vec{r}_{\mathcal{H}(\mathsf{id})}$ and c_τ opens only to $\mathsf{td}_{\mathsf{SE}}$ except with negligible probability over the choice of $\mathsf{crs}_{\mathsf{Com}}^{(I)}$ and $\mathsf{crs}_{\mathsf{Com}}^{(\tau)}$.

- Since $\mathsf{NIZK.Verify}(\mathsf{crs}_{\mathsf{NIZK}}, C, (\mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, c_\tau, \mathsf{sk}_{\mathsf{out}}, \mathsf{sk}_{\mathsf{aux}}, \mathsf{id}), \pi)$ and Π_{NIZK} is statistically sound, it must be the case that $(\mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, c_\tau, \mathsf{sk}_{\mathsf{out}}, \mathsf{sk}_{\mathsf{aux}}, \mathsf{id})$ is a true statement with overwhelming probability over the choice of $\mathsf{crs}_{\mathsf{NIZK}}$. This means that either $\mathsf{sk}_{\mathsf{out}} = \mathsf{PTE.Encode}(\mathsf{crs}_{\mathsf{PTE}}, \vec{r}_{\mathcal{H}(\mathsf{id})}; s_{\mathsf{out}})$ or $\mathsf{sk}_{\mathsf{aux}} = \mathsf{SE.EncodeSF}(\mathsf{crs}_{\mathsf{SE}}, \mathsf{td}_{\mathsf{SE}}, \mathsf{id}, 0; s_{\mathsf{aux}})$ for some randomness $s_{\mathsf{out}}, s_{\mathsf{aux}}$. We consider the two possibilities:
  - Suppose $\mathsf{sk}_{\mathsf{out}} = \mathsf{PTE.Encode}(\mathsf{crs}_{\mathsf{PTE}}, \vec{r}_{\mathcal{H}(\mathsf{id})}; s_{\mathsf{out}})$ for some s_{out}. Now, the reduction algorithm samples $\mathsf{ct}_{\mathsf{out}} \leftarrow \mathsf{SampSF}(\mathsf{td}_{\mathsf{PTE}}, \vec{r}_{\mathcal{H}(\mathsf{id})})$. By tester correctness of Π_{PTE}, this means
  $$\mathsf{PTE.Test}(\mathsf{crs}_{\mathsf{PTE}}, \mathsf{sk}_{\mathsf{out}}, \mathsf{ct}_{\mathsf{out}}) = 1.$$
  - Suppose $\mathsf{sk}_{\mathsf{aux}} = \mathsf{SE.EncodeSF}(\mathsf{crs}_{\mathsf{SE}}, \mathsf{td}_{\mathsf{SE}}, \mathsf{id}, 0; s_{\mathsf{aux}})$ for some s_{aux}. Since $\mathsf{ct}_{\mathsf{aux}}$ is a Type-1 semi-functional encoding with identity id, if $\mathsf{sk}_{\mathsf{aux}}$ is a Type-0 semi-functional encoding of the same id, we have $\mathsf{SE.Test}(\mathsf{crs}_{\mathsf{SE}}, \mathsf{sk}_{\mathsf{aux}}, \mathsf{ct}_{\mathsf{aux}}) = 1$ by tester correctness of Π_{SE}.

We conclude that with overwhelming probability over the choice of pp, for every candidate witness $w = (\mathsf{sk}_{\mathsf{out}}, \mathsf{sk}_{\mathsf{aux}}, \pi)$ at least one of the following conditions hold:

- $\mathsf{NIZK.Verify}(\mathsf{crs}_{\mathsf{NIZK}}, C, (\mathsf{crs}_{\mathsf{Com}}^{(I)}, \vec{c}_I, c_\tau, \mathsf{sk}_{\mathsf{out}}, \mathsf{sk}_{\mathsf{aux}}, \mathsf{id}), \pi) = 0$;
- $\mathsf{PTE.Test}(\mathsf{crs}_{\mathsf{PTE}}, \mathsf{sk}_{\mathsf{out}}, \mathsf{ct}_{\mathsf{out}}) = 1$; or
- $\mathsf{SE.Test}(\mathsf{crs}_{\mathsf{SE}}, \mathsf{sk}_{\mathsf{aux}}, \mathsf{ct}_{\mathsf{aux}}) = 1$.

In particular, this means that $\mathcal{R}_{\mathsf{WE}}(x, w) = 0$. We conclude that algorithm $\mathcal{B}$ breaks semantic security of Π_{WE} with advantage at least $\delta - \mathsf{negl}(\lambda)$, which is still non-negligible. □

Since $\mathsf{Hyb}_{5,i,T}$ is identical to $\mathsf{Hyb}_{5,i+1,0}$ for $i \in [k-1]$, the lemma follows from Claim 5.16 and a standard hybrid argument. □

Lemma 5.17. *For all efficient and admissible adversaries $\mathcal{A}$ and inverse polynomial functions $\varepsilon = 1/\mathsf{poly}(\lambda)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,*

$$\Pr[\mathsf{Hyb}_6(\mathcal{A}, \varepsilon) = 1] = \mathsf{negl}(\lambda).$$

Proof. Take any $\mathsf{id} \in \mathcal{J}$. For each $i \in [T]$, let $X_i^{\mathsf{id}} \in \{0,1\}$ be the random variable for whether algorithm $\mathcal{A}_2$'s output is correct (i.e., if $\beta' = \beta$ on the i^{th} iteration of AdvCheck with identity id). By construction, in Hyb_6, the adversary's view is *independent* of the bit $\beta \in \{0,1\}$. Since the challenger samples $\beta \xleftarrow{\mathrm{R}} \{0,1\}$ on each iteration, the probability that the adversary's guess $\beta' = \beta$ is exactly $1/2$. This means $\mathbb{E}[X_i^{\mathsf{id}}] = 1/2$ for all $i \in [T]$ and all $\mathsf{id} \in \mathcal{J}$. Moreover, $\mathsf{WINS} = \sum_{i \in [T]} X_i^{\mathsf{id}}$ and $\mathbb{E}[\mathsf{WINS}] = T/2$. By Hoeffding's inequality (Fact 2.1),

$$\Pr[\mathsf{WINS} - T/2 > \varepsilon T/16] \le \Pr[|\mathsf{WINS} - T/2| > \varepsilon T/16] \le 2^{-\Omega(T\varepsilon^2/256)} = \mathsf{negl}(\lambda),$$

since $T = \lambda/\varepsilon^2$. Thus, in an execution of $\mathsf{AdvCheck}^{\mathcal{A}_2}(1^\lambda, 1^{1/\varepsilon}, \mathsf{id}, \mathsf{msk}, \mathsf{pp}, \mathsf{st}, \mathsf{leak})$, $\mathsf{WINS} \ge T/2 + \varepsilon T/16$ with negligible probability. By a union bound, AdvCheck outputs 1 for any $\mathsf{id} \in \mathcal{J}$ with probability at most $|\mathcal{J}| \cdot \mathsf{negl}(\lambda) = \mathsf{negl}(\lambda)$ since $|\mathcal{J}| = \mathsf{poly}(\lambda)$. □

Security now follows by combining Lemmas 5.7 to 5.10, 5.13, 5.15 and 5.17. □

Combined with Theorem 3.4, this yields the following corollary:

Corollary 5.18 (Adaptive Security under Bounded Leakage). *Suppose the conditions in Theorem 5.6 hold. Then, Construction 5.3 is adaptively secure under bounded leakage for the same k as in Theorem 5.6.*

6 Constructing Split Encodings and Privately-Testable Encodings

In this section, we construct split encodings and privately-testable encodings from SXDH in pairing groups and DDH in pairing-free groups, respectively. We start by recalling the necessary notions for these constructions.

Definition 6.1 (Prime-Order Group). A prime-order group generator GroupGen is an efficient algorithm that takes as input the security parameter 1^λ and outputs a description $\mathcal{G} = (\mathbb{G}, p, g)$ of a group $\mathbb{G}$ with prime order $p = 2^{\Theta(\lambda)}$ and generator g. We require the group operation in $\mathbb{G}$ to be efficiently computable. We assume that the order of the group output by GroupGen is a fixed function of the security parameter λ.

Definition 6.2 (Prime-Order Bilinear Group). A prime-order (asymmetric) bilinear group generator BilinearGroupGen is an efficient algorithm that takes as input the security parameter 1^λ and outputs a description $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e)$ of two base groups $\mathbb{G}_1$ and $\mathbb{G}_2$ with generators g_1 and g_2, respectively, a target group $\mathbb{G}_T$, all of prime order $p = 2^{\Theta(\lambda)}$, and a non-degenerate bilinear map $e \colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. We require that the group operation in $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ and the pairing operations to be efficiently computable. We assume that the order of the group output by BilinearGroupGen is a fixed function of the security parameter λ.

Notation. Throughout this section, we will use the implicit representation of group elements [EHK+13]. Specifically, if $\mathcal{G} = (\mathbb{G}, p, g)$ is a prime-order group, and $\mathbf{M}$ is a matrix over $\mathbb{Z}_p$, we write $[\mathbf{M}]$ to denote $g^{\mathbf{M}}$, where exponentiation is defined component-wise. For a scalar $s \in \mathbb{Z}_p$, we write $s[\mathbf{M}] := [s \cdot \mathbf{M}]$. When working with an asymmetric prime-order pairing group $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e)$, we write $[\mathbf{M}]_1 := g_1^{\mathbf{M}}$, $[\mathbf{M}]_2 := g_2^{\mathbf{M}}$, and $[\mathbf{M}]_T := g_T^{\mathbf{M}}$, where $\mathbf{M}$ is a matrix over $\mathbb{Z}_p$ and $g_T = e(g_1, g_2)$.

Definition 6.3 (Decisional Diffie-Hellman). Let GroupGen be a prime-order group generator. The decisional Diffie-Hellman assumption DDH holds with respect to GroupGen if for all efficient adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$|\Pr[\mathcal{A}(\mathcal{G}, [x], [y], [xy]) = 1] - \Pr[\mathcal{A}(\mathcal{G}, [x], [y], [r]) = 1]| = \mathsf{negl}(\lambda),$$

where $\mathcal{G} \leftarrow \mathsf{GroupGen}(1^\lambda)$, and $x, y, r \xleftarrow{\mathrm{R}} \mathbb{Z}_p$.

Definition 6.4 (Symmetric External Diffie-Hellman). Let BilinearGroupGen be a prime-order asymmetric bilinear group generator. The symmetric external Diffie-Hellman assumption SXDH holds with respect to BilinearGroupGen if for all efficient adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$ and all $i \in \{1, 2\}$,

$$|\Pr[\mathcal{A}(\mathcal{G}, [x]_i, [y]_i, [xy]_i) = 1] - \Pr[\mathcal{A}(\mathcal{G}, [x]_i, [y]_i, [r]_i) = 1]| = \mathsf{negl}(\lambda),$$

where $\mathcal{G} \leftarrow \mathsf{BilinearGroupGen}(1^\lambda)$, and $x, y, r \xleftarrow{\mathrm{R}} \mathbb{Z}_p$. In other words, the SXDH assumption corresponds to DDH holding in *both* $\mathbb{G}_1$ and $\mathbb{G}_2$.

Tensor decisional Diffie-Hellman. When analyzing our constructions, it will be convenient to use the following variant of the DDH assumption. We show in Theorem 6.6 that the assumption follows generically from plain DDH.

Definition 6.5 (Tensor Diffie-Hellman Assumption)**.** Let $\mathsf{GroupGen}$ be a prime-order group generator and let $n_1, m_1, n_2, m_2 \in \mathbb{N}$ be dimensions. The tensor Diffie-Hellman assumption $\mathsf{TDDH}_{n_1,m_1,n_2,m_2}$ holds with respect to $\mathsf{GroupGen}$ if for all efficient adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$:

$$|\Pr[\mathcal{A}(\mathcal{G}, [\mathbf{A}], [\mathbf{B}], [\mathbf{A} \otimes \mathbf{B}]) = 1] - \Pr[\mathcal{A}(\mathcal{G}, [\mathbf{A}], [\mathbf{B}], [\mathbf{C}]) = 1]| = \mathsf{negl}(\lambda),$$

where $\mathcal{G} \leftarrow \mathsf{GroupGen}(1^\lambda)$, $\mathbf{A} \xleftarrow{\mathrm{R}} \mathbb{Z}_p^{n_1 \times m_1}$, $\mathbf{B} \xleftarrow{\mathrm{R}} \mathbb{Z}_p^{n_2 \times m_2}$, and $\mathbf{C} \xleftarrow{\mathrm{R}} \mathbb{Z}_p^{n_1 n_2 \times m_1 m_2}$. We define the $\mathsf{TDDH}_{n_1,m_1,n_2,m_2}$ assumption in $\mathbb{G}_1, \mathbb{G}_2$ with respect to a prime-order asymmetric bilinear group generator $\mathsf{BilinearGroupGen}$ in an analogous manner (to the SXDH assumption in Definition 6.4).

Theorem 6.6 (DDH implies TDDH)**.** *Let $\lambda \in \mathbb{N}$, and suppose $n_1, m_1, n_2, m_2 = \mathsf{poly}(\lambda)$. Suppose that* DDH *holds with respect to a prime-order group generator* GroupGen*. Then* $\mathsf{TDDH}_{n_1,m_1,n_2,m_2}$ *holds with respect to* GroupGen*. Analogously, if* SXDH *holds with respect to a prime-order asymmetric group generator* BilinearGroupGen*, then* $\mathsf{TDDH}_{n_1,m_1,n_2,m_2}$ *also holds with respect to* BilinearGroupGen*.*

Proof. We prove the claim for GroupGen. The claim for BilinearGroupGen follows analogously (using the fact that the SXDH assumption corresponds to the DDH assumption in $\mathbb{G}_1$ and $\mathbb{G}_2$). We now define a sequence of hybrid distributions.

- Hyb_0: $(\mathcal{G}, [\mathbf{A}], [\mathbf{B}], [\mathbf{C}])$ where $\mathbf{A} \xleftarrow{\mathrm{R}} \mathbb{Z}_p^{n_1 \times m_1}$, $\mathbf{B} \xleftarrow{\mathrm{R}} \mathbb{Z}_p^{n_2 \times m_2}$, and $\mathbf{C} = \mathbf{A} \otimes \mathbf{B}$. We index the components of $\mathbf{A}$ by a single index $i \in [n_1 m_1]$ (e.g., in row-major order). Specifically, we can write

$$\mathbf{A} = \begin{bmatrix} a_1 & \cdots & a_{m_1} \\ \vdots & \ddots & \vdots \\ a_{(n_1-1)\cdot m_1+1} & \cdots & a_{n_1 m_1} \end{bmatrix} \quad \text{and} \quad \mathbf{C} = \begin{bmatrix} a_1 \cdot \mathbf{B} & \cdots & a_{m_1} \cdot \mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{(n_1-1)\cdot m_1+1} \cdot \mathbf{B} & \cdots & a_{n_1 m_1} \cdot \mathbf{B} \end{bmatrix}.$$

  For $i \in [n_1 m_1]$, we write $\mathbf{C}_i \coloneqq a_i \mathbf{B}$.

- Hyb_i for $i \in [n_1 m_1]$: Same as Hyb_0 except for all $j \le i$, the challenger now samples $\mathbf{C}_j \xleftarrow{\mathrm{R}} \mathbb{Z}_p^{n_2 \times m_2}$.

By definition, in $\mathsf{Hyb}_{n_1 m_1}$, the challenger samples $\mathbf{C}_i \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n_2 \times m_2}$ for all $i \in [n_1 m_1]$, which corresponds to the uniform case in the $\mathsf{TDDH}_{n_1,m_1,n_2,m_2}$ assumption. To complete the proof, we use DDH to argue that for all $i \in [n_1 m_1]$, Hyb_i and Hyb_{i-1} are computationally indistinguishable.

Lemma 6.7. *Suppose* DDH *holds with respect to* GroupGen*. Then, for all $i \in [n_1 m_1]$ and all efficient adversaries $\mathcal{A}$, there exists a negligible function* negl *such that for all $\lambda \in \mathbb{N}$,*

$$\left|\Pr[\mathsf{Hyb}_{i-1}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_i(\mathcal{A}) = 1]\right| = \mathsf{negl}(\lambda).$$

Proof. We use the random self-reduction of DDH. Suppose there exists an efficient adversary $\mathcal{A}$ that distinguishes Hyb_i and Hyb_{i-1} with non-negligible advantage δ. We construct $\mathcal{B}$ that breaks DDH:

1. At the beginning of the game, algorithm $\mathcal{B}$ receives the DDH challenge $(\mathcal{G}, [x], [y], [z])$ from its challenger.

2. For all $j \in [n_2 m_2]$, algorithm $\mathcal{B}$ samples $\alpha_j, \beta_j \xleftarrow{R} \mathbb{Z}_p$. Algorithm $\mathcal{B}$ then defines $[\mathbf{B}] \in \mathbb{G}^{n_2 \times m_2}$ to be the matrix with components $[b_j] = \alpha_j [y] + \beta_j$ for all $j \in [n_2 m_2]$ (where j indexes the components of $[\mathbf{B}]$ in row-major order). Let $[\mathbf{C}_i]$ be the matrix with components $\alpha_j [z] + \beta_j [x]$. Finally, algorithm $\mathcal{B}$ sets $[a_i] = [x]$.
3. For $k \in [n_1 m_1]$ where $k \neq i$, algorithm $\mathcal{B}$ samples $a_k \xleftarrow{R} \mathbb{Z}_p$ itself. It defines the matrix $\mathbf{A}$ to be the matrix with components $[a_1], \ldots, [a_{n_1 m_1}]$.
4. Next, for $k > i$, algorithm $\mathcal{B}$ sets $[\mathbf{C}_k] = a_k [\mathbf{B}]$. For $k < i$, algorithm $\mathcal{B}$ samples $\mathbf{C}_k \xleftarrow{R} \mathbb{Z}_p^{n_2 m_2}$.
5. Finally, algorithm $\mathcal{B}$ gives $(\mathcal{G}, [\mathbf{A}], [\mathbf{B}], [\mathbf{C}])$ to $\mathcal{A}$ and outputs whatever $\mathcal{A}$ outputs.

Clearly $\mathcal{B}$ is efficient if $\mathcal{A}$ is since $n_1, m_1, n_2, m_2 = \mathsf{poly}(\lambda)$. If $z = xy$, then $\mathcal{B}$ perfectly simulates Hyb_{i-1} for $\mathcal{A}$. If $z \xleftarrow{R} \mathbb{Z}_p$, then algorithm $\mathcal{B}$ perfectly simulates Hyb_i for $\mathcal{A}$ since $\alpha_j, \beta_j \xleftarrow{R} \mathbb{Z}_p$. Thus $\mathcal{B}$ that breaks DDH with the same advantage δ. □

The theorem follows from Lemma 6.7 and a standard hybrid argument, since $n_1, m_1 = \mathsf{poly}(\lambda)$. □

6.1 Split Encoding from SXDH

In this section, we describe how to construct a split encoding from prime-order asymmetric bilinear groups and a pseudorandom function (Definition 2.11).

Construction 6.8 (Split Encoding from SXDH)**.** Let $\lambda \in \mathbb{N}$ be a security parameter and $\mathcal{T} = \{\mathcal{T}_\lambda\}_{\lambda \in \mathbb{N}}$ be the tag space. Our construction relies on the following primitives:

- Let BilinearGroupGen be a prime-order asymmetric bilinear group generator. Let $p = p(\lambda)$ be the order of the group output by BilinearGroupGen.
- Let $\mathsf{PRF}\colon \mathcal{K}_\lambda \times \mathcal{T}_\lambda \to (\mathbb{Z}_p^*)^3$ be a pseudorandom function with key space $\mathcal{K} = \{\mathcal{K}_\lambda\}_{\lambda \in \mathbb{N}}$.

We now construct our split encoding scheme $\Pi_{\mathsf{SE}} = (\mathsf{Setup}, \mathsf{SetupSF}, \mathsf{Encode}, \mathsf{EncodeSF}, \mathsf{Test})$ as follows:

- $\mathsf{Setup}(1^\lambda)$: On input the security parameter λ, the setup algorithm outputs the common reference string $\mathsf{crs} = \mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e) \leftarrow \mathsf{BilinearGroupGen}(1^\lambda)$.
- $\mathsf{SetupSF}(1^\lambda)$: On input the security parameter λ, the semi-functional setup algorithm outputs $\mathsf{crs} = \mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e) \leftarrow \mathsf{BilinearGroupGen}(1^\lambda)$ and $\mathsf{td} = K \xleftarrow{R} \mathcal{K}_\lambda$.
- $\mathsf{Encode}(\mathsf{crs}, \mathsf{type})$: On input the common reference string $\mathsf{crs} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e)$ and $\mathsf{type} \in \{0, 1\}$, the encode algorithm samples $\mathbf{x} \xleftarrow{R} (\mathbb{Z}_p^*)^2$. It then outputs $\mathsf{enc}_{\mathsf{type}}$ where $\mathsf{enc}_0 = [\mathbf{x}^{\mathsf{T}}]_1$ and $\mathsf{enc}_1 = [\mathbf{x}]_2$.
- $\mathsf{EncodeSF}(\mathsf{crs}, \mathsf{td}, \mathsf{tag}, \mathsf{type})$: On input the common reference string $\mathsf{crs} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e)$, a trapdoor $\mathsf{td} = K \in \mathcal{K}_\lambda$, $\mathsf{tag} \in \mathcal{T}_\lambda$ and $\mathsf{type} \in \{0, 1\}$, the semi-functional encode algorithm first computes $(x_1, x_2, y_1) = \mathsf{PRF}(K, \mathsf{tag})$, $y_2 = x_2^{-1}(-x_1 y_1)$. Let $\mathbf{x} = \left[\begin{smallmatrix} x_1 \\ x_2 \end{smallmatrix} \right]$ and $\mathbf{y} = \left[\begin{smallmatrix} y_1 \\ y_2 \end{smallmatrix} \right]$. Let $\mathsf{enc}_0 = [\mathbf{x}^{\mathsf{T}}]_1$, and $\mathsf{enc}_1 = [\mathbf{y}]_2$. Then, it samples $s \xleftarrow{R} \mathbb{Z}_p^*$ and outputs $s \cdot \mathsf{enc}_{\mathsf{type}}$.
- $\mathsf{Test}(\mathsf{crs}, \mathsf{enc}_0, \mathsf{enc}_1)$: On input the common reference string $\mathsf{crs} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e)$ and a pair of encodings $\mathsf{enc}_0 = [\mathbf{x}^{\mathsf{T}}]_1$ and $\mathsf{enc}_1 = [\mathbf{y}]_2$, the testing algorithm outputs 1 if $[\mathbf{x}^{\mathsf{T}}]_1 \cdot [\mathbf{y}]_2 = [0]_T$.

Theorem 6.9 (Tester Correctness)**.** *Construction 6.8 satisfies tester correctness.*

Proof. We show each property separately:

- Take any $\lambda \in \mathbb{N}$, $\mathsf{tag} \in \mathcal{T}_\lambda$, and $(\mathsf{crs}, \mathsf{td} = K)$ in the support of $\mathsf{SetupSF}(1^\lambda)$. Then $\mathsf{crs} = \mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e) \leftarrow \mathsf{BilinearGroupGen}(1^\lambda)$ and $\mathsf{td} = K \xleftarrow{\text{R}} \mathcal{K}_\lambda$. Suppose enc_0 is in the support of $\mathsf{EncodeSF}(\mathsf{crs}, K, \mathsf{tag}, 0)$ and enc_1 is in the support of $\mathsf{EncodeSF}(\mathsf{crs}, K, \mathsf{tag}, 1)$. In this case $\mathsf{enc}_0 = [s_0\mathbf{x}^\intercal]_1$ and $\mathsf{enc}_1 = [s_1\mathbf{y}]_2$ where $\mathbf{x}^\intercal\mathbf{y} = 0$. This means $[s_0\mathbf{x}^\intercal]_1 \cdot [s_1\mathbf{y}]_2 = [s_0s_1\mathbf{x}^\intercal\mathbf{y}]_T = [0]_T$, so $\mathsf{Test}(\mathsf{crs}, \mathsf{enc}_0, \mathsf{enc}_1)$ outputs 1.

- Now suppose $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda), \mathsf{enc}_0 \leftarrow \mathsf{Encode}(\mathsf{crs}, 0), \mathsf{enc}_1 \leftarrow \mathsf{Encode}(\mathsf{crs}, 1)$. In this case $\mathsf{enc}_0 = [\mathbf{x}^\intercal]_1$ and $\mathsf{enc}_1 = [\mathbf{y}]_2$ where $\mathbf{x}, \mathbf{y} \xleftarrow{\text{R}} (\mathbb{Z}_p^*)^2$. In this case, $[\mathbf{x}^\intercal]_1 \cdot [\mathbf{y}]_2 = [0]_T$ if and only if $\mathbf{x}^\intercal\mathbf{y} = 0$, which happens with probability at most $1/(p-1) = \mathsf{negl}(\lambda)$. □

Theorem 6.10 (Mode Indistinguishability)**.** *Suppose* SXDH *holds with respect to* BilinearGroupGen *and* PRF *is a secure pseudorandom function. Then, Construction 6.8 satisfies mode indistinguishability.*

Proof. Let $\mathcal{A}$ be an efficient adversary for the mode indistinguishability game, and let $Q = Q(\lambda)$ be a bound on the number of encoding queries algorithm $\mathcal{A}$ makes in the security game. We define a sequence of hybrid experiments:

- Hyb_0: This is the mode indistinguishability game with normal setup and encodings from Definition 5.1, which we recall in full below:

  1. The challenger samples $\mathsf{crs} = \mathcal{G} \leftarrow \mathsf{BilinearGroupGen}(1^\lambda)$ and gives crs to $\mathcal{A}$.
  2. When algorithm $\mathcal{A}$ makes an encoding query $(\mathsf{tag}, \mathsf{type})$, the challenger samples $\mathbf{x} \xleftarrow{\text{R}} (\mathbb{Z}_p^*)^2$ and replies with $\mathsf{enc}_{\mathsf{type}}$, where $\mathsf{enc}_0 = [\mathbf{x}^\intercal]_1$ and $\mathsf{enc}_1 = [\mathbf{x}]_2$.
  3. At the end of the game, $\mathcal{A}$ outputs $b' \in \{0,1\}$, which is the output of the experiment.

- Hyb_1: Same as Hyb_0 except at the beginning of the game, the challenger initializes an empty table T to keep track of the tags that the adversary has queried. Then, when the adversary makes an encoding query $(\mathsf{tag}, \mathsf{type})$, the challenger now responds as follows:

  – First, it checks if tag is present in the table T. If so, it sets $(\mathbf{x}, \mathbf{y}) = T[\mathsf{tag}]$. Otherwise, the challenger samples $\mathbf{x}, \mathbf{y} \xleftarrow{\text{R}} (\mathbb{Z}_p^*)^2$ and adds $T[\mathsf{tag}] = (\mathbf{x}, \mathbf{y})$.

  – Then the challenger samples $s \xleftarrow{\text{R}} \mathbb{Z}_p^*$ and sets $\mathsf{enc}_0 \leftarrow s[\mathbf{x}^\intercal]_1, \mathsf{enc}_1 \leftarrow s[\mathbf{y}]_2$. It responds with $\mathsf{enc}_{\mathsf{type}}$.

- Hyb_2: Same as Hyb_1 except whenever the challenger samples $\mathbf{y} \xleftarrow{\text{R}} (\mathbb{Z}_p^*)^2$ in respond to an encoding query, it now samples $\mathbf{x} \xleftarrow{\text{R}} (\mathbb{Z}_p^*)^2$, $y_1 \xleftarrow{\text{R}} \mathbb{Z}_p^*$, and sets $y_2 = x_2^{-1}(-x_1y_1)$.

- Hyb_3: Same as Hyb_2 except at the beginning of the experiment, the challenger samples $\mathsf{td} = K \xleftarrow{\text{R}} \mathcal{K}_\lambda$. Then, when answering encoding queries $(\mathsf{tag}, \mathsf{type})$, instead of sampling $x_1, x_2, y_1 \xleftarrow{\text{R}} \mathbb{Z}_p^*$, the challenger instead computes $(x_1, x_2, y_1) = \mathsf{PRF}(K, \mathsf{tag})$. This is the mode indistinguishability game with semi-functional setup and encodings.

We write $\mathsf{Hyb}_i(\mathcal{A})$ to denote the output of Hyb_i with adversary $\mathcal{A}$.

Lemma 6.11. *Suppose the* SXDH *holds with respect to* BilinearGroupGen*. Then, for all efficient adversaries $\mathcal{A}$, there exists a negligible function* negl *such that for all $\lambda \in \mathbb{N}$,*

$$\left|\Pr[\mathsf{Hyb}_0(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_1(\mathcal{A}) = 1]\right| = \mathsf{negl}(\lambda).$$

Proof. We define an intermediate hybrid Hyb_0' to be the same as Hyb_1 except only the Type-0 encoding queries are changed. We appeal to SXDH to show that both pairs $(\mathsf{Hyb}_0, \mathsf{Hyb}_0')$ and $(\mathsf{Hyb}_0', \mathsf{Hyb}_1)$ are computationally indistinguishable. Specifically, we use the tensor decisional Diffie-Hellman assumption (Definition 6.5), which is implied by SXDH (Theorem 6.6).

Claim 6.12. *Suppose the* $\mathsf{TDDH}_{Q,1,Q,2}$ *assumption holds in* $\mathbb{G}_1$ *with respect to* $\mathsf{BilinearGroupGen}$. *Then, for all efficient adversaries* $\mathcal{A}$, *there exists a negligible function* negl *such that for all* $\lambda \in \mathbb{N}$,

$$\left|\Pr[\mathsf{Hyb}_0(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_0'(\mathcal{A}) = 1]\right| = \mathsf{negl}(\lambda).$$

Proof. Suppose there exists an efficient adversary $\mathcal{A}$ that makes Q Type-0 encoding queries and distinguishes Hyb_0 and Hyb_0' with non-negligible probability δ. We use $\mathcal{A}$ to construct an algorithm $\mathcal{B}$ that breaks $\mathsf{TDDH}_{Q,1,Q,2}$ in $\mathbb{G}_1$:

1. At the beginning of the game, algorithm $\mathcal{B}$ gets a tuple $(\mathcal{G}, [\mathbf{A}]_1, [\mathbf{B}]_1, [\mathbf{C}]_1)$ from the TDDH challenger, where $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e)$ and $\mathbf{A} \in \mathbb{Z}_p^{Q\times 1}$, $\mathbf{B} \in \mathbb{Z}_p^{Q\times 2}$, and $\mathbf{C} \in \mathbb{Z}_p^{Q^2\times 2}$.
2. Algorithm $\mathcal{B}$ sets $\mathsf{crs} = \mathcal{G}$ and sends crs to $\mathcal{A}$. In addition, it sets $i = 1, j = 1$ and initializes a table T.
3. When algorithm $\mathcal{A}$ makes a Type-0 encoding query on tag, algorithm $\mathcal{B}$ does the following:
   - If $\mathsf{tag} \notin T$, algorithm algorithm $\mathcal{B}$ computes $\mathsf{enc} \leftarrow [\mathbf{c}_{i,j}^{\mathsf{T}}]_1$, where $\mathbf{c}_{i,j}$ is the $(i(Q-1)+j)^{\text{th}}$ row of $\mathbf{C}$. Algorithm $\mathcal{B}$ then sets $T[\mathsf{tag}] := j$ and $i = i+1$, $j = j+1$.
   - If $\mathsf{tag} \in T$, algorithm $\mathcal{B}$ computes $\mathsf{enc} \leftarrow [\mathbf{c}_{i,T[\mathsf{tag}]}^{\mathsf{T}}]_1$. Algorithm $\mathcal{B}$ then sets $i = i+1$.

   Algorithm $\mathcal{B}$ responds with enc.
4. Algorithm $\mathcal{B}$ answers all Type-1 queries with $[\mathbf{x}]_2$ where $\mathbf{x} \xleftarrow{\text{R}} (\mathbb{Z}_p^*)^2$. At the end of the game, algorithm $\mathcal{B}$ outputs what algorithm $\mathcal{A}$ outputs.

Let $a_1, \ldots, a_Q \in \mathbb{Z}_p$ be the entries of $\mathbf{A}$ and $\mathbf{b}_1^{\mathsf{T}}, \ldots, \mathbf{b}_Q^{\mathsf{T}} \in \mathbb{Z}_p^2$ be the rows of $\mathbf{B}$. If $\mathbf{C} = \mathbf{A} \otimes \mathbf{B}$, then $\mathbf{c}_{i,j}^{\mathsf{T}} = a_i \mathbf{b}_j^{\mathsf{T}}$. Since $p = 2^{\Theta(\lambda)}$, with overwhelming probability over the choice of $\mathbf{B}$, it holds that $\mathbf{A} \in (\mathbb{Z}_p^*)^{Q\times 1}$ and $\mathbf{B} \in (\mathbb{Z}_p^*)^{Q\times 2}$. Now, if $\mathbf{C} = \mathbf{A} \otimes \mathbf{B}$, algorithm $\mathcal{B}$ simulates the Hyb_0' challenger (where a_i is encoding randomness on the i^{th} query, and $\mathbf{b}_j^{\mathsf{T}}$ is the vector $\mathbf{x}$ associated with tag). If $\mathbf{C}$ is uniform, then algorithm $\mathcal{B}$ simulates the Hyb_0 challenger. Thus, algorithm $\mathcal{B}$ breaks $\mathsf{TDDH}_{Q,1,Q,2}$ with advantage that is negligibly close to δ. □

Claim 6.13. *Suppose the* $\mathsf{TDDH}_{Q,1,Q,2}$ *assumption holds in* $\mathbb{G}_2$ *with respect to* $\mathsf{BilinearGroupGen}$. *Then, for all efficient adversaries* $\mathcal{A}$, *there exists a negligible function* negl *such that for all* $\lambda \in \mathbb{N}$,

$$\left|\Pr[\mathsf{Hyb}_0'(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_1(\mathcal{A}) = 1]\right| = \mathsf{negl}(\lambda).$$

Proof. Suppose there exists an efficient adversary $\mathcal{A}$ that makes Q Type-1 encoding queries and distinguishes Hyb_0' and Hyb_1 with non-negligible probability δ. We use $\mathcal{A}$ to construct an algorithm $\mathcal{B}$ that breaks $\mathsf{TDDH}_{Q,1,Q,2}$:

1. At the beginning of the game, algorithm $\mathcal{B}$ gets a tuple $(\mathcal{G}, [\mathbf{A}]_2, [\mathbf{B}]_2, [\mathbf{C}]_2)$ from the TDDH challenger, where $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e)$ and and $\mathbf{A} \in \mathbb{Z}_p^{Q\times 1}$, $\mathbf{B} \in \mathbb{Z}_p^{Q\times 2}$, and $\mathbf{C} \in \mathbb{Z}_p^{Q^2\times 2}$.
2. Algorithm $\mathcal{B}$ sets $\mathsf{crs} = \mathcal{G}$ and sends crs to $\mathcal{A}$. In addition, it sets $i = 1, j = 1$ and initializes tables T_0, T_1.

3. When algorithm $\mathcal{A}$ makes a Type-1 encoding query on tag, algorithm $\mathcal{B}$ does the following:
   - If tag $\notin T_1$, algorithm algorithm $\mathcal{B}$ computes enc $\leftarrow [\mathbf{c}_{i,j}^{\mathsf{T}}]_2$, where $\mathbf{c}_{i,j}$ is the $(i(Q-1)+j)^{\text{th}}$ row of $\mathbf{C}$. Algorithm $\mathcal{B}$ then sets $T_1[\mathsf{tag}] := j$ and $i = i+1$, $j = j+1$.
   - If tag is in the table, algorithm $\mathcal{B}$ computes enc $\leftarrow [\mathbf{c}_{i,T[\mathsf{tag}]}^{\mathsf{T}}]_2$. Algorithm $\mathcal{B}$ then sets $i = i+1$.

   Algorithm $\mathcal{B}$ responds with enc.

4. When algorithm $\mathcal{A}$ makes a Type-0 encoding query on tag, algorithm $\mathcal{B}$ does the following:
   - If tag $\notin T_0$, algorithm $\mathcal{B}$ samples $\mathbf{x} \xleftarrow{R} (\mathbb{Z}_p^*)^2$, $s \xleftarrow{R} \mathbb{Z}_p^*$, sets $T_0[\mathsf{tag}] := [\mathbf{x}^{\mathsf{T}}]_1$, and responds with enc $\leftarrow s[\mathbf{x}^{\mathsf{T}}]_1$.
   - If tag is in the table, algorithm $\mathcal{B}$ samples $s \xleftarrow{R} \mathbb{Z}_p^*$ and outputs enc $\leftarrow s \cdot T_0[\mathsf{tag}]$.

5. At the end of the game, algorithm $\mathcal{B}$ outputs whatever algorithm $\mathcal{A}$ outputs.

Let $a_1, \ldots, a_Q \in \mathbb{Z}_p$ be the entries of $\mathbf{A}$ and $\mathbf{b}_1^{\mathsf{T}}, \ldots, \mathbf{b}_Q^{\mathsf{T}} \in \mathbb{Z}_p^2$ be the rows of $\mathbf{B}$. If $\mathbf{C} = \mathbf{A} \otimes \mathbf{B}$, then $\mathbf{c}_{i,j}^{\mathsf{T}} = a_i \mathbf{b}_j^{\mathsf{T}}$. Since $p = 2^{\Theta(\lambda)}$, with overwhelming probability over the choice of $\mathbf{B}$, it holds that $\mathbf{A} \in (\mathbb{Z}_p^*)^{Q \times 1}$ and $\mathbf{B} \in (\mathbb{Z}_p^*)^{Q \times 2}$. Now, if $\mathbf{C} = \mathbf{A} \otimes \mathbf{B}$, algorithm $\mathcal{B}$ simulates the Hyb_1 challenger and if $\mathbf{C}$ is uniform, algorithm $\mathcal{B}$ simulates the Hyb_0' challenger. Thus, algorithm $\mathcal{B}$ breaks $\mathsf{TDDH}_{Q,1,Q,2}$ with advantage negligibly close to δ. □

The lemma follows from Claims 6.12 and 6.13 and a standard hybrid argument. □

Lemma 6.14. *For all admissible adversaries $\mathcal{A}$,* $\Pr[\mathsf{Hyb}_1(\mathcal{A}) = 1] = \Pr[\mathsf{Hyb}_2(\mathcal{A}) = 1]$.

Proof. Since $\mathcal{A}$ is admissible, it does not make a Type-0 and a Type-1 encoding query on the same tag. Thus, for any tag, algorithm $\mathcal{A}$ either observes a function of $\mathbf{x}$ only or a function of $\mathbf{y}$ only (but never both), where $\mathbf{x}^{\mathsf{T}}\mathbf{y} = 0$. It remains to show that the *marginal* distribution of $\mathbf{x}$ and $\mathbf{y}$ individually is uniform over $(\mathbb{Z}_p^*)^2$:

- By construction $\mathbf{x} \xleftarrow{R} (\mathbb{Z}_p^*)^2$, so the marginal distribution of $\mathbf{x}$ is uniform.
- Consider the marginal distribution of $\mathbf{y}$. First, $y_1 \xleftarrow{R} \mathbb{Z}_p^*$. Next, $y_2 = x_2^{-1}(-x_1 y_1)$, where $x_2 \xleftarrow{R} \mathbb{Z}_p^*$. Since $x_1, y_1 \in \mathbb{Z}_p^*$, this means $x_1 y_1 \neq 0$, and so y_2 is uniform over $\mathbb{Z}_p^*$ and independent of y_1.

We conclude that Hyb_1 and Hyb_2 are identical distributions, as required. □

Lemma 6.15. *Suppose* PRF *is a secure pseudorandom function. Then, for all efficient adversaries $\mathcal{A}$, there exists a negligible function* negl *such that for all $\lambda \in \mathbb{N}$,*

$$\left|\Pr[\mathsf{Hyb}_2(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_3(\mathcal{A}) = 1]\right| = \mathsf{negl}(\lambda).$$

Proof. Suppose there exists an efficient adversary $\mathcal{A}$ that distinguishes Hyb_2 and Hyb_3 with non-negligible probability δ. We use $\mathcal{A}$ to construct $\mathcal{B}$ that breaks PRF security of PRF:

1. At the beginning of the game, algorithm $\mathcal{B}$ gets 1^λ from the PRF challenger. $\mathcal{B}$ samples crs $\leftarrow$ $\mathsf{Setup}(1^\lambda)$ and gives crs to $\mathcal{A}$.

2. When $\mathcal{A}$ makes an encoding query on (tag, type), $\mathcal{B}$ queries the PRF challenger on tag to get $(x_1, x_2, y_1) \in (\mathbb{Z}_p^*)^3$. Algorithm $\mathcal{B}$ sets $y_2 = x_2^{-1}(-x_1 y_1)$, $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, and $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$. Finally, it sets $\mathsf{enc}_0 = [\mathbf{x}^{\mathsf{T}}]_1$, and $\mathsf{enc}_1 = [\mathbf{y}]_2$. Finally, it samples $s \xleftarrow{R} \mathbb{Z}_p^*$ and outputs $s \cdot \mathsf{enc}_{\mathsf{type}}$.

3. At the end of the experiment, $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs.

If the challenger computes $(x_1, x_2, y_1) = f(\mathsf{tag})$ where f is a uniform random function from $\mathcal{T}_\lambda \to (\mathbb{Z}_p^*)^3$, then algorithm $\mathcal{B}$ simulates an execution of Hyb_2. If the challenger derives $(x_1, x_2, y_1) = \mathsf{PRF}(K, \mathsf{tag})$, then $\mathcal{B}$ simulates an execution of Hyb_3. Thus, $\mathcal{B}$ breaks security of PRF with advantage δ. □

Combining Lemmas 6.11, 6.14 and 6.15 yields the statement by a hybrid argument. □

6.2 Privately Testable Encoding from DDH

In this section, we describe how to construct a privately-testable encoding from prime-order pairing-free groups.

Construction 6.16 (Privately Testable Encoding from DDH)**.** Let $\lambda \in \mathbb{N}$ be a security parameter, $d = d(\lambda)$ be a dimension parameter, GroupGen be a prime-order group generator. Let $p = p(\lambda)$ be the order of the group output by GroupGen and let $\mathcal{X} = \{(\mathbb{Z}_{p(\lambda)}^*)^d\}_{\lambda \in \mathbb{N}}$ be the input space. We construct our privately testable encoding scheme $\Pi_{\mathsf{PTE}} = (\mathsf{Setup}, \mathsf{SetupSF}, \mathsf{Samp}, \mathsf{SampSF}, \mathsf{Encode}, \mathsf{EncodeSF}, \mathsf{Test})$ as follows:

- $\mathsf{Setup}(1^\lambda)$: On input the security parameter λ, the setup algorithm samples $\mathcal{G} = (\mathbb{G}, p, g) \leftarrow \mathsf{GroupGen}(1^\lambda)$ and $\mathbf{H} \xleftarrow{\mathrm{R}} (\mathbb{Z}_p^*)^{d \times d}$. It outputs $\mathsf{crs} = (\mathcal{G}, [\mathbf{H}])$.
- $\mathsf{SetupSF}(1^\lambda)$: On input the security parameter λ, the semi-functional setup algorithm samples $\mathcal{G} = (\mathbb{G}, p, g) \leftarrow \mathsf{GroupGen}(1^\lambda)$ and $\mathbf{H} \xleftarrow{\mathrm{R}} (\mathbb{Z}_p^*)^{d \times d}$. Then it outputs $\mathsf{crs} = (\mathcal{G}, [\mathbf{H}])$ and $\mathsf{td} = \mathbf{H}$.
- $\mathsf{Samp}(1^\lambda)$: On input the security parameter λ, the sampling algorithm outputs $\mathbf{u} \xleftarrow{\mathrm{R}} (\mathbb{Z}_p^*)^d$.
- $\mathsf{SampSF}(\mathsf{td}, \mathbf{v})$: On input a trapdoor td and a vector $\mathbf{v} \in (\mathbb{Z}_p^*)^d$, the trapdoor sampling algorithm outputs $\mathsf{td}_\mathbf{v} \xleftarrow{\mathrm{R}} \mathcal{V}_{\mathbf{Hv}}^\perp$, where $\mathcal{V}_{\mathbf{Hv}}^\perp$ is the space of vectors orthogonal to $\mathbf{Hv}$.
- $\mathsf{Encode}(\mathsf{crs}, \mathbf{v})$: On input the common reference string crs and a vector $\mathbf{v} \in (\mathbb{Z}_p^*)^d$, the encode algorithm samples $s \xleftarrow{\mathrm{R}} \mathbb{Z}_p^*$ and outputs $s[\mathbf{Hv}]$.
- $\mathsf{EncodeSF}(\mathsf{crs})$: On input the common reference string crs, the semi-functional encode algorithm outputs $[\mathbf{x}]$ where $\mathbf{x} \xleftarrow{\mathrm{R}} (\mathbb{Z}_p^*)^d$.
- $\mathsf{Test}(\mathsf{crs}, \mathsf{enc}, \mathsf{td}_\mathbf{v}) \to \{0, 1\}$: On input the common reference string crs, an encoding $\mathsf{enc} = [\mathbf{u}]$, and a vector trapdoor $\mathsf{td}_\mathbf{v} \in (\mathbb{Z}_p^*)^d$, the testing algorithm outputs 1 if $[\mathsf{td}_\mathbf{v}^\mathsf{T} \mathbf{u}] = [0]$ and 0 otherwise.

Theorem 6.17 (Tester Correctness)**.** *Construction 6.16 satisfies tester correctness.*

Proof. We show each property separately:

- Take any $\lambda \in \mathbb{N}$ and any $(\mathsf{crs}, \mathsf{td})$ in the support of $\mathsf{SetupSF}(1^\lambda)$. Then $\mathsf{crs} = (\mathcal{G}, [\mathbf{H}])$ where $\mathcal{G} = (\mathbb{G}, p, g)$. Take any input $\mathbf{v} \in (\mathbb{Z}_p^*)^d$ and any encoding enc in the support of $\mathsf{Encode}(\mathsf{crs}, \mathbf{v})$. This means $\mathsf{enc} = s[\mathbf{Hv}]$ for some $s \in \mathbb{Z}_p^*$. Let $\mathsf{td}_\mathbf{v} \leftarrow \mathsf{SampSF}(\mathsf{td}, \mathbf{v})$. By definition, $\mathsf{td}_\mathbf{v}$ is orthogonal to $\mathbf{Hv}$. This means $\mathsf{td}_\mathbf{v} \cdot \mathbf{Hv} = 0$, and correspondingly, that $\mathsf{td}_\mathbf{v} \cdot (s\mathbf{Hv}) = 0$. This means $\mathsf{Test}(\mathsf{crs}, \mathsf{enc}, \mathsf{td}_\mathbf{v})$ outputs 1, as required.
- For the second property, suppose $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$, $\mathsf{enc} \leftarrow \mathsf{Encode}(\mathsf{crs}, \mathbf{v})$, and $\mathbf{u} \leftarrow \mathsf{Samp}(1^\lambda)$. In this case, $\mathsf{crs} = (\mathcal{G}, [\mathbf{H}])$ and $\mathsf{enc} = s[\mathbf{Hv}]$ where $\mathbf{H} \xleftarrow{\mathrm{R}} (\mathbb{Z}_p^*)^{d \times d}$, $s \xleftarrow{\mathrm{R}} \mathbb{Z}_p^*$, and $\mathbf{u} \xleftarrow{\mathrm{R}} (\mathbb{Z}_p^*)^d$. Over the random choice of $\mathbf{u}$, the probability that $\mathbf{u}^\mathsf{T}\mathbf{Hv} = 0$ is at most $1/(p-1) = \mathsf{negl}(\lambda)$. Correspondingly, $\mathsf{Test}(\mathsf{crs}, \mathsf{enc}, \mathbf{u}) = 1$ with negligible probability. □

Theorem 6.18 (Mode Indistinguishability). *Suppose* DDH *holds with respect to* GroupGen. *Then, Construction 6.16 satisfies mode indistinguishability.*

Proof. Suppose there exists an efficient adversary $\mathcal{A}$ that makes Q encoding queries and wins the mode indistinguishability game with non-negligible advantage δ. We use $\mathcal{A}$ to construct an algorithm $\mathcal{B}$ that breaks $\mathsf{TDDH}_{Q,1,d,d}$:

1. At the beginning of the game, algorithm $\mathcal{B}$ gets a tuple $(\mathcal{G}, [\mathbf{A}], [\mathbf{B}], [\mathbf{C}])$ from the TDDH challenger, where $\mathcal{G} = (\mathbb{G}, p, g)$ and $\mathbf{A} \in \mathbb{Z}_p^{Q \times 1}$, $\mathbf{B} \in \mathbb{Z}_p^{d \times d}$, and $\mathbf{C} \in \mathbb{Z}_p^{dQ \times d}$. Parse $\mathbf{C}$ as the vertical concatenation of $\mathbf{C}_1, \ldots, \mathbf{C}_Q$ where $\mathbf{C}_i \in \mathbb{Z}_p^{d \times d}$.

2. Algorithm $\mathcal{B}$ gives $\mathsf{crs} = (\mathcal{G}, [\mathbf{B}])$ to $\mathcal{A}$.

3. When $\mathcal{A}$ makes its i^{th} encoding query on a vector $\mathbf{v}$, algorithm $\mathcal{B}$ replies with $\mathsf{enc} = [\mathbf{C}_i] \cdot \mathbf{v}$.

4. At the end of the game, algorithm $\mathcal{B}$ outputs whatever algorithm $\mathcal{A}$ outputs.

First, since $p = 2^{\Theta(\lambda)}$, the uniform distribution over $\mathbb{Z}_p^*$ is statistically indistinguishable from the uniform distribution over $\mathbb{Z}_p$. Thus, the components of the CRS are correctly distributed with overwhelming probability. Let $a_1, \ldots, a_Q$ be the entries of $\mathbf{A}$. If $\mathbf{C} = \mathbf{A} \otimes \mathbf{B}$, then $\mathbf{C}_i = a_i \mathbf{B}$. In this case, algorithm $\mathcal{B}$ responds to the encoding queries according to $\mathsf{Encode}(\mathsf{crs}, \mathbf{v})$, where a_i is the encoding randomness. If $\mathbf{C} \xleftarrow{\text{R}} \mathbb{Z}_p^{dQ \times d}$ is uniform, then the encoding queries are implemented according to $\mathsf{SetupSF}(\mathsf{crs})$. Thus, algorithm $\mathcal{B}$ breaks $\mathsf{TDDH}_{Q,1,d,d}$ with the same advantage δ. □

Theorem 6.19 (k-Trapdoor Indistinguishability). *Let $\lambda \in \mathbb{N}$ be the security parameter and $d = d(\lambda) > 2$ for all λ. Then, Construction 6.16 satisfies $(\omega(\log \lambda) + 2 \log p)$-trapdoor uniformity.*

Proof. We start by showing the following consequence of the leftover hash lemma (Corollary 2.9):

Lemma 6.20. *Let $\mathbb{F}$ be a finite field, $n > 2$ be an integer, $\varepsilon > 0$ be fixed, and $S \subseteq \mathbb{F} \setminus \{0\}$ be a set. Suppose X is a random variable over S^n such that $\mathbf{H}_\infty(X) \geq k \geq 2 \log |\mathbb{F}| + 2 \log(1/\varepsilon)$. Then, the statistical distance between the following distributions is at most $\varepsilon/2$:*

$$\left\{ \mathbf{y} \xleftarrow{\text{R}} \mathbb{F}^n \right\} \quad \textit{and} \quad \left\{ \mathbf{y}^\perp \xleftarrow{\text{R}} \mathcal{V}_{\mathbf{x}}^\perp : \mathbf{x} \leftarrow X \right\},$$

where $\mathcal{V}_{\mathbf{x}}^\perp$ is the set of vectors that are orthogonal to $\mathbf{x}$.

Proof. For any $\mathbf{x} \in S^n$, we can sample $\mathbf{y}^\perp \xleftarrow{\text{R}} \mathcal{V}_{\mathbf{x}}^\perp$ by sampling $y_1^\perp, \ldots, y_{n-1}^\perp \xleftarrow{\text{R}} \mathbb{F}$ and setting

$$y_n^\perp = -x_n^{-1} \sum_{i \in [n-1]} x_i y_i^\perp.$$

It suffices to show that $\sum_{i=1}^{n-1} x_i y_i^\perp$ is statistically close to uniform over $\mathbb{F}$. This follows by Corollary 2.9, with $n' = n - 1$ and min-entropy $k - \log |\mathbb{F}| \geq \log |\mathbb{F}| + 2 \log(1/\varepsilon)$ (specifically, we treat $y_1, \ldots, y_{n-1}$ as the seed for the extractor and $x_1, \ldots, x_n$ as the randomness source). □

Theorem 6.19 follows directly from Lemma 6.20 by taking $\log(1/\varepsilon) = \omega(\log \lambda)$ and $n = d$. Specifically, in Construction 5.3, the matrix $\mathbf{H} \xleftarrow{\text{R}} (\mathbb{Z}_p^*)^{d \times d}$ sampled by Setup and SetupSF is full-rank with overwhelming probability. In this case, if a random vector $\mathbf{v} \in (\mathbb{Z}_p^*)^d$ has k-bits of min-entropy, the vector $\mathbf{Hv}$ also has k-bits of min-entropy (over $\mathbb{Z}_p^d$). □

Acknowledgments

Brent Waters is supported by NSF CNS-1908611, CNS-2318701, and a Simons Investigator award. David J. Wu is supported by NSF CNS-2140975, CNS-2318701, a Microsoft Research Faculty Fellowship, and a Google Research Scholar award.

References

[ABSV15] Prabhanjan Ananth, Zvika Brakerski, Gil Segev, and Vinod Vaikuntanathan. From selective to adaptive security in functional encryption. In *CRYPTO*, pages 657–677, 2015.

[ADN+10] Joël Alwen, Yevgeniy Dodis, Moni Naor, Gil Segev, Shabsi Walfish, and Daniel Wichs. Public-key encryption in the bounded-retrieval model. In *EUROCRYPT*, pages 113–134, 2010.

[ADW09] Joël Alwen, Yevgeniy Dodis, and Daniel Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In *CRYPTO*, pages 36–54, 2009.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In *CRYPTO*, pages 213–229, 2001.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *STOC*, pages 103–112, 1988.

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *CRYPTO*, pages 1–18, 2001.

[BGI+12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6:1–6:48, 2012.

[BHK11] Mark Braverman, Avinatan Hassidim, and Yael Tauman Kalai. Leaky pseudo-entropy functions. In *Innovations in Computer Science*, pages 353–366, 2011.

[BKR16] Mihir Bellare, Daniel Kane, and Phillip Rogaway. Big-key symmetric encryption: Resisting key exfiltration. In *CRYPTO*, pages 373–402, 2016.

[BL20] Fabrice Benhamouda and Huijia Lin. Mr NISC: multiparty reusable non-interactive secure computation. In *TCC*, pages 349–378, 2020.

[CDD+07] David Cash, Yan Zong Ding, Yevgeniy Dodis, Wenke Lee, Richard J. Lipton, and Shabsi Walfish. Intrusion-resilient key exchange in the bounded retrieval model. In *TCC*, pages 479–498, 2007.

[CDG+17] Chongwon Cho, Nico Döttling, Sanjam Garg, Divya Gupta, Peihan Miao, and Antigoni Polychroniadou. Laconic oblivious transfer and its applications. In *CRYPTO*, pages 33–65, 2017.

[CLW06] Giovanni Di Crescenzo, Richard J. Lipton, and Shabsi Walfish. Perfectly secure password protocols in the bounded retrieval model. In *TCC*, pages 225–244, 2006.

[Coc01] Clifford C. Cocks. An identity based encryption scheme based on quadratic residues. In *Cryptography and Coding*, pages 360–363, 2001.

[DFR+07] Ivan Damgård, Serge Fehr, Renato Renner, Louis Salvail, and Christian Schaffner. A tight high-order entropic quantum uncertainty relation with applications. In *CRYPTO*, pages 360–378, 2007.

[DGSW22] Nico Döttling, Sanjam Garg, Sruthi Sekar, and Mingyuan Wang. IBE with incompressible master secret and small identity secrets. In *TCC*, pages 588–617, 2022.

[DN02] Ivan Damgård and Jesper Buus Nielsen. Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor. In *CRYPTO*, pages 581–596, 2002.

[DORS08] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam D. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008.

[Dzi06] Stefan Dziembowski. Intrusion-resilience via the bounded-storage model. In *TCC*, pages 207–224, 2006.

[EHK+13] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge L. Villar. An algebraic framework for diffie-hellman assumptions. In *CRYPTO*, pages 129–147, 2013.

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, pages 40–49, 2013.

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In *STOC*, pages 467–476, 2013.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *STOC*, pages 291–304, 1985.

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In *EUROCRYPT*, pages 415–432, 2008.

[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, 1999.

[Hoe63] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301), 1963.

[ILL89] Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudo-random generation from one-way functions (extended abstracts). In *STOC*, pages 12–24, 1989.

[LP09] Yehuda Lindell and Benny Pinkas. A proof of security of yao's protocol for two-party computation. *J. Cryptol.*, 22(2):161–188, 2009.

[LW10] Allison B. Lewko and Brent Waters. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In *TCC*, pages 455–479, 2010.

[MW20] Tal Moran and Daniel Wichs. Incompressible encodings. In *CRYPTO*, pages 494–523, 2020.

[Nao89] Moni Naor. Bit commitment using pseudo-randomness. In *CRYPTO*, pages 128–136, 1989.

[Par19] Noam Parzanchevski. *Dispersers with logarithmic entropy loss.* Msc thesis, Tel-Aviv University, 2019.

[QWW18] Willy Quach, Hoeteck Wee, and Daniel Wichs. Laconic function evaluation and applications. In *FOCS*, pages 859–870, 2018.

[Sha84] Adi Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO*, pages 47–53, 1984.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *STOC*, pages 475–484, 2014.

[TUZ07] Amnon Ta-Shma, Christopher Umans, and David Zuckerman. Lossless condensers, unbalanced expanders, and extractors. *Comb.*, 27(2):213–240, 2007.

[Wat05] Brent Waters. Efficient identity-based encryption without random oracles. In *EUROCRYPT*, pages 114–127, 2005.

[Wat09] Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In *CRYPTO*, pages 619–636, 2009.

[WW24] Brent Waters and Daniel Wichs. Adaptively secure attribute-based encryption from witness encryption. In *TCC*, pages 65–90, 2024.

[Yao82] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *FOCS*, pages 160–164, 1982.

A Transforming Adaptive Big-Key IBE to have Short Public Parameters

The work of [DGSW22] provide a generic transformation that takes any big-key IBE scheme with *long* public parameters (that scale with the leakage bound), but where encryption and decryption only requires local access to the public parameters, and transforms it into a big-key IBE scheme with short public parameters. Their approach relies on a non-interactive secure computation (NISC) scheme. In this section, we show that the same transformation preserves adaptive security. To do so, we first modify the NISC privacy definition to a somewhat adaptive version. Then, in Appendix B, we show that the NISC construction of [CDG+17] indeed satisfies the stronger variant (in fact, this fact is implicit in their existing analysis).

Definition A.1 (Non-Interactive Secure Computation in the RAM Model [CDG+17, adapted])**.** A non-interactive secure computation scheme in the RAM model is a tuple of efficient algorithms $\Pi_{\mathsf{NISC}} = (\mathsf{Setup}, \mathsf{EncData}, \mathsf{EncProg}, \mathsf{Decrypt})$ with the following syntax:

- $\mathsf{Setup}(1^\lambda) \to \mathsf{crs}$: On input the security parameter λ, the setup algorithm outputs a common reference string crs.
- $\mathsf{EncData}(\mathsf{crs}, D) \to (\mathsf{dig}, \widetilde{D})$: On input the common reference string crs and the database D, the data encryption algorithm outputs a digest dig and a database state $\widetilde{D}$.
- $\mathsf{EncProg}(\mathsf{crs}, \mathsf{dig}, (P, x, t)) \to \mathsf{ct}$: On input the common reference string crs, a digest dig, and a program P with input x and maximum run-time t, the program encryption algorithm outputs a ciphertext ct.

- $\mathsf{Decrypt}^{\widetilde{D}}(\mathsf{crs}, \mathsf{ct}) \to y$: On input the common reference string crs and a ciphertext ct, the decryption algorithm outputs a string y. Additionally, the decryption algorithm has RAM access to a database state $\widetilde{D}$.

Moreover, Π_{NISC} should satisfy the following properties:

- **Correctness:** For all polynomials $M = M(\lambda)$, all security parameters $\lambda \in \mathbb{N}$, all databases $D \in \{0,1\}^M$, and all RAM program tuples (P, x, t), it holds that

$$\Pr[\mathsf{Decrypt}^{\widetilde{D}}(\mathsf{crs}, \mathsf{ct}) = P^D(x)] = 1,$$

  where $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$, $(\mathsf{dig}, \widetilde{D}) \leftarrow \mathsf{EncData}(\mathsf{crs}, D)$, and $\mathsf{ct} \leftarrow \mathsf{EncProg}(\mathsf{crs}, \mathsf{dig}, (P, x, t))$.

- **Privacy:** There exists an efficient algorithm SimEnc that takes as input a common reference string crs, a digest dig, a database D, an output string y, and a memory access pattern MemAccess, and outputs a ciphertext ct. For a security parameter $\lambda \in \mathbb{N}$ and a bit $b \in \{0,1\}$, we define the privacy game between an adversary $\mathcal{A}$ and a challenger as follows:

  1. On input the security parameter 1^λ, algorithm $\mathcal{A}$ chooses a database $D \in \{0,1\}^M$ and a program P and a bound on the running time 1^t to the challenger.
  2. The challenger samples $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$ and $(\mathsf{dig}, \widetilde{D}) \leftarrow \mathsf{EncData}(\mathsf{crs}, D)$. It gives $(\mathsf{crs}, \mathsf{dig}, \widetilde{D})$ to $\mathcal{A}$.
  3. $\mathcal{A}$ sends an input x to the challenger.
  4. If $b = 0$, the challenger computes $\mathsf{ct} \leftarrow \mathsf{EncProg}(\mathsf{crs}, \mathsf{dig}, (P, x, t))$. If $b = 1$ the challenger computes $\mathsf{ct} \leftarrow \mathsf{SimEnc}(\mathsf{crs}, \mathsf{dig}, D, y, \mathsf{MemAccess})$, where $y = P^D(x)$ and MemAccess is the memory-access pattern of $P^D(x)$. The challenger sends ct to $\mathcal{A}$.
  5. Algorithm $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$, which is the output of the experiment.

  We say Π_{NISC} satisfies privacy if there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all efficient adversaries $\mathcal{A}$ in the above privacy game,

$$|\Pr[b' = 1 \mid b = 0] - \Pr[b' = 1 \mid b = 1]| = \mathsf{negl}(\lambda).$$

  In this definition, the database D, the program P, and the running time t are committed to ahead of time. However, the input x is chosen *adaptively*.

- **Efficiency:** The length of dig output by $\mathsf{EncData}(\mathsf{crs}, D)$ is a fixed polynomial in λ (independent of $|D|$). The algorithm EncData runs in time $M \cdot \mathsf{poly}(\lambda, \log M)$. The algorithms EncProg, Decrypt run in time $t \cdot \mathsf{poly}(\lambda, \log M)$.

The [DGSW22] transformation. We now recall the the transformation from [DGSW22], which takes as input a big-key IBE scheme $\Pi_{\mathsf{bkIBE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ with large public parameters $\mathsf{pp} \in \{0,1\}^N$ and compiles it into one with short public parameters:

Construction A.2 (Big-Key IBE with Small Public Parameters [DGSW22])**.** Let $\lambda \in \mathbb{N}$ be a security parameter, ℓ be a fixed leakage parameter, and $N = N(\lambda, \ell)$ be a key size parameter. Let $\mathcal{ID} = \{\mathcal{ID}_\lambda\}_{\lambda \in \mathbb{N}}$ be the identity space. The construction relies on the following primitives:

- Let $\Pi_{\mathsf{IBE}} = (\mathsf{IBE.Setup}, \mathsf{IBE.KeyGen}, \mathsf{IBE.Encrypt}, \mathsf{IBE.Decrypt})$ be a big-key IBE scheme.
- Let $\Pi_{\mathsf{NISC}} = (\mathsf{NISC.Setup}, \mathsf{NISC.EncData}, \mathsf{NISC.EncProg}, \mathsf{NISC.Decrypt})$ be a NISC scheme in the RAM model.

We construct the big-key IBE scheme $\Pi_{\mathsf{bkIBE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ as follows:

- $\mathsf{Setup}(1^\lambda, 1^\ell)$: On input the security parameter λ, the setup algorithm proceeds as follows:
  1. Sample $(\mathsf{pp}_{\mathsf{IBE}}, \mathsf{msk}_{\mathsf{IBE}}) \leftarrow \mathsf{IBE.Setup}(1^\lambda, 1^\ell)$ and $\mathsf{crs}_{\mathsf{NISC}} \leftarrow \mathsf{NISC.Setup}(1^\lambda)$.
  2. Compute $(\mathsf{dig}, \widetilde{\mathsf{pp}}) \leftarrow \mathsf{NISC.EncData}(\mathsf{crs}_{\mathsf{NISC}}, \mathsf{pp}_{\mathsf{IBE}})$.

  Output $\mathsf{pp} = (\mathsf{crs}_{\mathsf{NISC}}, \mathsf{dig})$ and $\mathsf{msk} = (\mathsf{msk}_{\mathsf{IBE}}, \widetilde{\mathsf{pp}})$.
- $\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$: On input the master secret key msk and an identity $\mathsf{id} \in \mathcal{ID}_\lambda$, the key generation algorithm computes $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{IBE.KeyGen}(\mathsf{msk}_{\mathsf{IBE}}, \mathsf{id})$ and outputs $(\mathsf{sk}_{\mathsf{id}}, \widetilde{\mathsf{pp}}_{\mathsf{id}})$, where $\widetilde{\mathsf{pp}}_{\mathsf{id}}$ is the part of $\widetilde{\mathsf{pp}}$ accessed by $\mathsf{NISC.Decrypt}$, which depends on id.
- $\mathsf{Encrypt}(\mathsf{pp}, \mathsf{id}, m)$: On input the public parameters pp, an identity $\mathsf{id} \in \mathcal{ID}_\lambda$, and a message m, the encryption algorithm outputs $\mathsf{NISC.EncProg}(\mathsf{crs}_{\mathsf{NISC}}, \mathsf{dig}, (\mathsf{IBE.Encrypt}, (\mathsf{id}, m), t))$, where $\mathsf{IBE.Encrypt}$ is considered a RAM program that accesses $D = \mathsf{pp}_{\mathsf{IBE}}$ and t is a bound on its run-time.
- $\mathsf{Decrypt}(\mathsf{sk}_{\mathsf{id}}, \mathsf{id}, \mathsf{ct})$: On input an identity secret key $\mathsf{sk}_{\mathsf{id}} = (\mathsf{sk}'_{\mathsf{id}}, \widetilde{\mathsf{pp}}_{\mathsf{id}})$, an identity $\mathsf{id} \in \mathcal{ID}_\lambda$, and a ciphertext ct, the decryption algorithm outputs $\mathsf{IBE.Decrypt}(\mathsf{sk}'_{\mathsf{id}}, \mathsf{id}, \mathsf{NISC.Decrypt}(\mathsf{crs}_{\mathsf{NISC}}, \widetilde{\mathsf{pp}}_{\mathsf{id}}, \mathsf{ct}))$, where the extra input $\widetilde{\mathsf{pp}}_{\mathsf{id}}$ replaces the RAM access to $\widetilde{\mathsf{pp}}$.

Theorem A.3 (Correctness). *Suppose $\Pi_{\mathsf{IBE}}, \Pi_{\mathsf{NISC}}$ are correct. Then, Construction A.2 is correct.*

Proof. Take any security parameter λ, identity $\mathsf{id} \in \mathcal{ID}_\lambda$, and message m. Let $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$, where $\mathsf{pp} = (\mathsf{crs}_{\mathsf{NISC}}, \mathsf{dig})$, $\mathsf{msk} = (\mathsf{msk}_{\mathsf{IBE}}, \widetilde{\mathsf{pp}})$, $(\mathsf{sk}_{\mathsf{id}}, \widetilde{\mathsf{pp}}_{\mathsf{id}}) \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$, $\mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{pp}, \mathsf{id}, m)$. Consider the output of $\mathsf{Decrypt}(\mathsf{sk}_{\mathsf{id}}, \mathsf{id}, \mathsf{ct})$:

- By construction of KeyGen, $\mathsf{sk}_{\mathsf{id}}$ is an honestly generated identity key and $\widetilde{\mathsf{pp}}_{\mathsf{id}}$ can replace RAM access to $\widetilde{\mathsf{pp}}$.
- By construction of $\mathsf{Encrypt}$, ct is an honest encryption of $\mathsf{IBE.Encrypt}$ on message m.
- By correctness of Π_{NISC}, $\mathsf{NISC.Decrypt}(\mathsf{crs}_{\mathsf{NISC}}, \widetilde{\mathsf{pp}}_{\mathsf{id}}, \mathsf{ct})$ yields the output of $\mathsf{IBE.Encrypt}(\mathsf{pp}, \mathsf{id}, m)$, and by correctness of Π_{IBE}, $\mathsf{IBE.Decrypt}(\mathsf{sk}_{\mathsf{id}}, \mathsf{id}, \mathsf{IBE.Encrypt}(\mathsf{pp}, \mathsf{id}, m)) = m$ with overwhelming probability, as desired. □

Theorem A.4 (Adaptive Advantage-Checker Security under Bounded Leakage). *Suppose for all polynomially-bounded $\ell = \ell(\lambda)$, Π_{IBE} is adaptively advantage-checker secure under bounded leakage with challenge parameter k. Suppose also that Π_{NISC} satisfies privacy. Then, Construction A.2 is adaptively advantage-checker secure under bounded leakage with challenge parameter k.*

Proof. We define a sequence of hybrid experiments, each parameterized (implicitly) by an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ and an advantage threshold function $\varepsilon = \varepsilon(\lambda)$:

- Hyb_0: This is the adaptive advantage-checker security game from Definition 3.1, which we recall in full below:

- **Setup:** The challenger samples the components $(\mathsf{pp}_{\mathsf{IBE}}, \mathsf{msk}_{\mathsf{IBE}}) \leftarrow \mathsf{IBE.Setup}(1^\lambda, 1^\ell)$, $\mathsf{crs}_{\mathsf{NISC}} \leftarrow \mathsf{NISC.Setup}(1^\lambda)$, and It then computes $(\mathsf{dig}, \widetilde{\mathsf{pp}}) \leftarrow \mathsf{NISC.EncData}(\mathsf{crs}, \mathsf{pp}_{\mathsf{IBE}})$. It sets

$$\mathsf{pp} = (\mathsf{crs}_{\mathsf{NISC}}, \mathsf{dig}) \quad \text{and} \quad \mathsf{msk} = (\mathsf{msk}_{\mathsf{IBE}}, \widetilde{\mathsf{pp}})$$

  and gives pp to $\mathcal{A}$.
- **Pre-leakage queries:** When algorithm $\mathcal{A}_1$ makes a query on $\mathsf{id} \in \mathcal{ID}_\lambda$, the challenger computes $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{IBE.KeyGen}(\mathsf{msk}_{\mathsf{IBE}}, \mathsf{id})$ and replies with $(\mathsf{sk}_{\mathsf{id}}, \widetilde{\mathsf{pp}}_{\mathsf{id}})$.
- **Leakage:** After $\mathcal{A}_1$ outputs the description of an efficiently-computable leakage function f, the challenger replies with $\mathsf{leak} \leftarrow f(\mathsf{msk})$.
- **Post-leakage queries:** The challenger responds to post-leakage key queries exactly as in the pre-leakage phase.
- **Challenge:** Algorithm $\mathcal{A}_1$ outputs a set $\mathcal{J} \subseteq \mathcal{ID}_\lambda$ of size $\geq k$, two messages m_0, m_1, and a state st.
- **Output:** The output of Hyb_0 is $b' = 1$ if $\mathcal{A}$ is admissible and

$$\forall \mathsf{id} \in \mathcal{J} : \mathsf{AdvCheck}^{\mathcal{A}_2}(1^\lambda, 1^{1/\varepsilon}, \mathsf{id}, \mathsf{msk}, \mathsf{pp}, \mathsf{st}, \mathsf{leak}) = 1, \tag{A.1}$$

  and $b' = 0$ otherwise. The advantage-checker algorithm AdvCheck is defined as follows:

Inputs: security parameter λ, threshold $\varepsilon \in (0, 1)$, identity $\mathsf{id} \in \mathcal{ID}_\lambda$, master secret key $\mathsf{msk} = (\mathsf{msk}_{\mathsf{IBE}}, \widetilde{\mathsf{pp}})$, public parameters $\mathsf{pp} = (\mathsf{crs}_{\mathsf{NISC}}, \mathsf{dig})$, state st, string leak, and (oracle) access to an algorithm $\mathcal{A}$

- Let $T = \lambda/\varepsilon^2$ and initialize a counter $\mathsf{WINS} \leftarrow 0$.
- The advantage-checker algorithm now simulates T independent executions of experiment $\mathsf{Exp}^{\mathsf{id}}(\mathsf{msk}, \mathsf{pp}, \mathsf{st}, \mathsf{leak})$ for algorithm $\mathcal{A}$.
  1. Sample $\beta \xleftarrow{\text{R}} \{0, 1\}$.
  2. The challenger computes

$$\mathsf{ct} \leftarrow \mathsf{NISC.EncProg}(\mathsf{crs}_{\mathsf{NISC}}, \mathsf{dig}, (\mathsf{IBE.Encrypt}, (\mathsf{id}, m_\beta), t)).$$

     Run $\mathcal{A}$ on input $(\mathsf{st}, \mathsf{id}, \mathsf{ct})$.
  3. Whenever algorithm $\mathcal{A}$ makes a key-generation query on an identity $\mathsf{id} \in \mathcal{ID}_\lambda$, the challenger computes $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{IBE.KeyGen}(\mathsf{msk}_{\mathsf{IBE}}, \mathsf{id})$ and replies to $\mathcal{A}$ with $(\mathsf{sk}_{\mathsf{id}}, \widetilde{\mathsf{pp}}_{\mathsf{id}})$.
  4. After $\mathcal{A}$ has finished making key-generation queries, it outputs a bit $\beta' \in \{0, 1\}$.
  5. If $\beta = \beta'$, then increment $\mathsf{WINS} \leftarrow \mathsf{WINS} + 1$.
- Output 1 if $\mathsf{WINS} \geq \frac{T}{2} + \frac{\varepsilon T}{2}$ and 0 otherwise.

Figure 9: Function $\mathsf{AdvCheck}^{\mathcal{A}}(1^\lambda, 1^{1/\varepsilon}, \mathsf{id}, \mathsf{msk}, \mathsf{pp}, \mathsf{st}, \mathsf{leak})$ in Construction A.2

- Hyb_1: Same as Hyb_0, except the challenger constructs the challenge ciphertext ct in the procedure $\mathsf{AdvCheck}^{\mathcal{A}_2}$ as $\mathsf{ct} \leftarrow \mathsf{SimEnc}(\mathsf{crs}_{\mathsf{NISC}}, \mathsf{dig}, \mathsf{pp}_{\mathsf{IBE}}, y, \mathsf{MemAccess})$, where the output y is computed as $y \leftarrow \mathsf{IBE.Encrypt}(\mathsf{pp}_{\mathsf{IBE}}, \mathsf{id}, m_\beta)$ and $\mathsf{MemAccess}$ is the memory access pattern for $\mathsf{IBE.Encrypt}$.

For an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, we write $\mathsf{Hyb}_i(\mathcal{A}, \varepsilon)$ to denote the output of Hyb_i with adversary $\mathcal{A}$ and inner threshold function ε. Our goal is to show that for all efficient adversaries $\mathcal{A}$ and all inverse polynomial functions $\varepsilon = 1/\mathsf{poly}(\lambda)$, $\Pr[\mathsf{Hyb}_0(\mathcal{A}, \varepsilon) = 1] = \mathsf{negl}(\lambda)$. We proceed via a hybrid argument.

Lemma A.5. *Suppose that Π_{NISC} satisfies privacy. Then, for all efficient and admissible adversaries $\mathcal{A}$ and inverse polynomial functions $\varepsilon = 1/\mathsf{poly}(\lambda)$, there exists a negligible function* negl *such that for all $\lambda \in \mathbb{N}$,*

$$\Pr[\mathsf{Hyb}_1(\mathcal{A}, \varepsilon) = 1] \geq \Pr[\mathsf{Hyb}_0(\mathcal{A}, \varepsilon) = 1] - \mathsf{negl}(\lambda).$$

Proof. We define a sequence of intermediate hybrids:

- $\mathsf{Hyb}_{0,1,0}$: Same as Hyb_0. Notably, the challenge ciphertext is sampled as

$$\mathsf{ct} \leftarrow \mathsf{NISC.EncProg}(\mathsf{crs}_{\mathsf{NISC}}, \mathsf{dig}, (\mathsf{IBE.Encrypt}, (\mathsf{id}, m), t)),$$

  where components are defined as in Hyb_0 in $\mathsf{AdvCheck}$.

- $\mathsf{Hyb}_{0,i,j}$: Same as Hyb_0 except for all (i', j') such that $i' < i$ or $i' = i, j' \leq j$, we sample the challenge ciphertext in the j'^{th} execution of $\mathsf{Exp}^{\mathsf{id}=\mathcal{I}[i']}$ in $\mathsf{AdvCheck}$ as

$$\mathsf{ct} \leftarrow \mathsf{SimEnc}(\mathsf{crs}_{\mathsf{NISC}}, \mathsf{dig}, \mathsf{pp}_{\mathsf{IBE}}, y, \mathsf{MemAccess}),$$

  where $y \leftarrow \mathsf{IBE.Encrypt}(\mathsf{pp}_{\mathsf{IBE}}, \mathsf{id}, m_\beta)$ and $\mathsf{MemAccess}$ is the memory access pattern for the program $\mathsf{IBE.Encrypt}$ which can be computed given id. Note that $\mathsf{Hyb}_{0,k,T}$ is the same as Hyb_1 and that $\mathsf{Hyb}_{0,i,T}$ is the same as $\mathsf{Hyb}_{0,i+1,0}$ for $i \in [k-1]$.

We now appeal to privacy of Π_{NISC} to show that $\mathsf{Hyb}_{0,i,j}$ and $\mathsf{Hyb}_{0,i,j-1}$ are computationally indistinguishable for all $i \in [k], j \in [T]$.

Claim A.6. *Suppose the conditions in Lemma A.5 hold. Then for all $i \in [k], j \in [T]$, all efficient and admissible adversaries $\mathcal{A}$ and inverse polynomial functions $\varepsilon = 1/\mathsf{poly}(\lambda)$, there exists a negligible function* negl *such that for all $\lambda \in \mathbb{N}$,*

$$\Pr[\mathsf{Hyb}_{0,i,j}(\mathcal{A}, \varepsilon) = 1] \geq \Pr[\mathsf{Hyb}_{0,i,j-1}(\mathcal{A}, \varepsilon) = 1] - \mathsf{negl}(\lambda).$$

Proof. Suppose there exists an efficient adversary $\mathcal{A}$ that distinguishes Hyb_0 and Hyb_1 with non-negligible probability δ. We use $\mathcal{A}$ to construct algorithm $\mathcal{B}$ that breaks privacy of Π_{NISC}:

1. Algorithm $\mathcal{B}$ samples $(\mathsf{pp}_{\mathsf{IBE}}, \mathsf{msk}_{\mathsf{IBE}}) \leftarrow \mathsf{IBE.Setup}(1^\lambda, 1^\ell)$ and gives $(\mathsf{pp}_{\mathsf{IBE}}, \mathsf{IBE.Encrypt}, t)$ to the privacy challenger, where t is a bound on the run-time of $\mathsf{IBE.Encrypt}$. The challenger replies with a tuple $(\mathsf{crs}_{\mathsf{NISC}}, \mathsf{dig}, \widetilde{\mathsf{pp}})$.

2. Algorithm $\mathcal{B}$ sets $\mathsf{pp} = (\mathsf{crs}_{\mathsf{NISC}}, \mathsf{dig})$ and $\mathsf{msk} = (\mathsf{msk}_{\mathsf{IBE}}, \widetilde{\mathsf{pp}})$ and gives pp to $\mathcal{A}$.

3. When algorithm $\mathcal{A}$ issues a key-generation query on an identity id, algorithm $\mathcal{B}$ queries its challenger to get $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{IBE.KeyGen}(\mathsf{msk}_{\mathsf{IBE}}, \mathsf{id})$. Algorithm $\mathcal{B}$ replies to $\mathcal{A}$ with $(\mathsf{sk}_{\mathsf{id}}, \widetilde{\mathsf{pp}}_{\mathsf{id}})$, where $\widetilde{\mathsf{pp}}_{\mathsf{id}}$ is defined as in Construction A.2.

4. When algorithm $\mathcal{A}$ outputs an efficiently-computable leakage function f, algorithm $\mathcal{B}$ replies with $\mathsf{leak} = f(\mathsf{msk})$. In the challenge phase, algorithm $\mathcal{A}$ outputs $\mathcal{J} \subseteq \mathcal{ID}_\lambda$ of size at least k, messages m_0, m_1, and a state st.

5. For all (i', j') such that $i' < i$ or $i' = i, j' < j$, algorithm $\mathcal{B}$ samples the challenge ciphertext in the j'^{th} execution of $\mathsf{Exp}^{\mathcal{J}[i']}$ in AdvCheck as $\mathsf{ct} \leftarrow \mathsf{SimEnc}(\mathsf{crs}_{\mathsf{NISC}}, \mathsf{dig}, \mathsf{pp}_{\mathsf{IBE}}, y, \mathsf{MemAccess})$, where $\beta \xleftarrow{\text{R}} \{0,1\}$, $y \leftarrow \mathsf{IBE.Encrypt}(\mathsf{pp}_{\mathsf{IBE}}, \mathcal{J}[i'], m_\beta)$, and MemAccess is the memory access pattern for the program IBE.Encrypt.

6. For the j^{th} execution of $\mathsf{Exp}^{\mathcal{J}[i]}$, algorithm $\mathcal{B}$ samples $\beta \xleftarrow{\text{R}} \{0,1\}$ and sends $x = (\mathcal{J}[i], m_\beta)$ to the privacy challenger. Algorithm $\mathcal{B}$ uses the response ct from the privacy challenger as the challenge ciphertext in this execution.

7. In the remaining executions of $\mathsf{Exp}^{\mathcal{J}[i']}$ in iterations of AdvCheck, algorithm $\mathcal{B}$ computes challenge ciphertexts as $\mathsf{ct} \leftarrow \mathsf{NISC.EncProg}(\mathsf{crs}_{\mathsf{NISC}}, \mathsf{dig}, (\mathsf{IBE.Encrypt}, (\mathcal{J}[i'], m_\beta), t))$, where $\beta \xleftarrow{\text{R}} \{0,1\}$. Algorithm $\mathcal{B}$ outputs the output of the experiment.

If ct from the privacy challenger is constructed using NISC.EncProg, algorithm $\mathcal{B}$ simulates $\mathsf{Hyb}_{0,i,j-1}$ for $\mathcal{A}$. If ct from the privacy challenger is constructed using SimEnc, algorithm $\mathcal{B}$ simulates $\mathsf{Hyb}_{0,i,j}$ for $\mathcal{A}$. Thus, algorithm $\mathcal{B}$ breaks privacy of Π_{NISC} with advantage δ, as desired. ◻

The lemma now follows from Claim A.6 and a standard hybrid argument. ◻

Lemma A.7. *Suppose for all polynomially-bounded $\ell = \ell(\lambda)$, Π_{IBE} is k-adaptively advantage-checker secure under bounded leakage. Then, for all efficient and admissible adversaries $\mathcal{A}$ and inverse polynomial functions $\varepsilon = 1/\mathsf{poly}(\lambda)$, there exists a negligible function* negl *such that for all $\lambda \in \mathbb{N}$,*

$$\Pr[\mathsf{Hyb}_1(\mathcal{A}, \varepsilon) = 1] = \mathsf{negl}(\lambda).$$

Proof. Suppose there exists an efficient and admissible adversary $\mathcal{A}$ such that $\Pr[\mathsf{Hyb}_1(\mathcal{A}, \varepsilon) = 1] = \delta$ for some non-negligible δ. We use $\mathcal{A}$ to construct algorithm $\mathcal{B}$ that breaks the k-adaptive advantage-checker security under bounded leakage of Π_{IBE}:

- **Setup:** The challenger for Π_{IBE} starts by sampling $(\mathsf{pp}_{\mathsf{IBE}}, \mathsf{msk}_{\mathsf{IBE}}) \leftarrow \mathsf{IBE.Setup}(1^\lambda, 1^\ell)$ and gives $\mathsf{pp}_{\mathsf{IBE}}$ to algorithm $\mathcal{B}$. Algorithm $\mathcal{B}$ samples $\mathsf{crs}_{\mathsf{NISC}} \leftarrow \mathsf{NISC.Setup}(1^\lambda)$ and computes $(\mathsf{dig}, \widetilde{\mathsf{pp}}) \leftarrow \mathsf{NISC.EncData}(\mathsf{crs}_{\mathsf{NISC}}, \mathsf{pp}_{\mathsf{IBE}})$. Algorithm $\mathcal{B}$ gives $\mathsf{pp} = (\mathsf{crs}_{\mathsf{NISC}}, \mathsf{dig})$ to $\mathcal{A}$.

- **Key-generation queries:** When algorithm $\mathcal{A}$ issues a key-generation query on an identity id, algorithm $\mathcal{B}$ queries its challenger to get $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{IBE.KeyGen}(\mathsf{msk}_{\mathsf{IBE}}, \mathsf{id})$. Algorithm $\mathcal{B}$ replies to $\mathcal{A}$ with $(\mathsf{sk}_{\mathsf{id}}, \widetilde{\mathsf{pp}}_{\mathsf{id}})$, where $\widetilde{\mathsf{pp}}_{\mathsf{id}}$ is defined as in Construction A.2.

- **Leakage:** When algorithm $\mathcal{A}$ outputs the description of an efficiently-computable function f with output length at most ℓ, algorithm $\mathcal{B}$ gives the same function to its challenger to get $\mathsf{leak} := f(\mathsf{msk})$. Algorithm $\mathcal{B}$ gives leak to $\mathcal{A}$.

- **Challenge:** Algorithm $\mathcal{B}$ outputs the same set $\mathcal{J} \subseteq \mathcal{ID}_\lambda$ and messages (m_0, m_1) that $\mathcal{A}$ outputs.

- **Output:** When constructing a challenge ciphertext for identity $\mathsf{id} \in \mathcal{J}$ in some iteration of AdvCheck, algorithm $\mathcal{B}$ takes the challenge ciphertext ct from its challenger and computes

$$\mathsf{ct}' \leftarrow \mathsf{SimEnc}(\mathsf{crs}_{\mathsf{NISC}}, \mathsf{dig}, \mathsf{pp}_{\mathsf{IBE}}, \mathsf{ct}, \mathsf{MemAccess}),$$

where MemAccess can be computed given id. Algorithm $\mathcal{B}$ gives ct' to $\mathcal{A}$ and outputs the same bit β' as $\mathcal{A}$.

Since algorithm $\mathcal{B}$ perfectly simulates the experiment for $\mathcal{A}$ and answers the same way, the probability of the k-adaptive advantage-checker experiment outputting 1 is δ, as desired. □

Combining Lemmas A.5 and A.7 yields the statement by a hybrid argument. □

Corollary A.8 (Adaptive Security under Bounded Leakage)**.** *Suppose the conditions in Theorem A.4 hold. Then, Construction A.2 is adaptively secure under bounded leakage for the same k as in Theorem A.4.*

Proof. Follows immediately from Theorem 3.4. □

NISC instantiations. As seen in Appendix B, the core primitive needed to build a NISC for our purposes is a laconic oblivious transfer scheme. We remark that such a primitive can be constructed from DDH [CDG+17] or LWE [QWW18], so this transform retains our instantiation statements from Section 6.

B NISC in the RAM Model with Adaptive Privacy

We show how to achieve Definition A.1 via the NISC construction of [CDG+17]. To do so, we define laconic oblivious transfer and garbled circuits for convenience below.

Definition B.1 (Laconic Oblivious-Transfer [CDG+17])**.** A laconic oblivious transfer scheme is a tuple of efficient algorithms $\Pi_{\mathsf{OT}} = (\mathsf{Setup}, \mathsf{Hash}, \mathsf{Send}, \mathsf{Receive})$ with the following syntax:

- $\mathsf{Setup}(1^\lambda) \to \mathsf{crs}$: On input the security parameter λ, the setup algorithm outputs a common reference string crs.
- $\mathsf{Hash}(\mathsf{crs}, D) \to (\mathsf{dig}, \widehat{D})$: On input the common reference string crs and a database $D \in \{0,1\}^*$, the hashing algorithm outputs a digest dig and a database state $\widehat{D}$.
- $\mathsf{Send}(\mathsf{crs}, \mathsf{dig}, L, m_0, m_1) \to \mathsf{ct}$: On input the common reference string crs, a digest dig, a database location $L \in \mathbb{N}$, and a pair of messages (m_0, m_1) each of length λ, the send algorithm outputs a ciphertext ct.
- $\mathsf{Receive}^{\widehat{D}}(\mathsf{crs}, \mathsf{ct}, L) \to m$: On input the common reference string crs, a ciphertext ct, and a database location $L \in \mathbb{N}$, the receive algorithm outputs a message m. Additionally, the receive algorithm has RAM access to a database state $\widehat{D}$.

Moreover, Π_{OT} should satisfy the following properties:

- **Correctness:** For all security parameters $\lambda \in \mathbb{N}$, all databases $D \in \{0,1\}^M$ (where $M = \mathsf{poly}(\lambda)$ for any polynomial $\mathsf{poly}(\cdot)$), any memory location $L \in [M]$, and any pair of messages $(m_0, m_1) \in \{0,1\}^\lambda \times \{0,1\}^\lambda$, it holds that

$$\Pr[\mathsf{Receive}^{\widehat{D}}(\mathsf{crs}, \mathsf{Send}(\mathsf{crs}, \mathsf{dig}, L, m_0, m_1), L) = m_{D[L]}] = 1,$$

  where $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$ and $(\mathsf{dig}, \widehat{D}) \leftarrow \mathsf{Hash}(\mathsf{crs}, D)$.

- **Sender privacy against semi-honest receivers:** There exists an efficient algorithm OTSim that takes as input a common reference string crs, a database D, a location L, and a message m, and outputs a ciphertext ct. A laconic OT scheme satisfies sender privacy if for all security parameters $\lambda \in \mathbb{N}$, all databases $D \in \{0,1\}^M$ with $M = \mathsf{poly}(\lambda)$, all locations $L \in [M]$, and any pair of messages $(m_0, m_1) \in \{0,1\}^\lambda \times \{0,1\}^\lambda$, it holds that the distributions

$$(\mathsf{crs}, \mathsf{Send}(\mathsf{crs}, \mathsf{dig}, L, m_0, m_1)) \text{ and } (\mathsf{crs}, \mathsf{OTSim}(\mathsf{crs}, D, L, m_{D[L]}))$$

are computationally indistinguishable, where $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$ and $(\mathsf{dig}, \widehat{D}) \leftarrow \mathsf{Hash}(\mathsf{crs}, D)$.

- **Efficiency:** The length of dig output by $\mathsf{Hash}(\mathsf{crs}, D)$ is a fixed polynomial in λ, independent of $|D|$.

Definition B.2 (Garbled Circuits [Yao82, LP09]). A circuit garbling scheme is a tuple of efficient algorithms $\Pi_{\mathsf{GC}} = (\mathsf{Garble}, \mathsf{Eval})$ with the following syntax:

- $\mathsf{Garble}(1^\lambda, C, \{\mathsf{key}_{w,b}\}_{w \in \mathsf{inp}(C), b \in \{0,1\}}) \rightarrow \widetilde{C}$: On input the security parameter λ, a circuit C, and a set of labels $\mathsf{key}_{w,b}$ for all the input wires $w \in \mathsf{inp}(C)$ and $b \in \{0,1\}$, the garbling algorithm outputs a garbled circuit $\widetilde{C}$. Here, $\mathsf{inp}(C)$ denotes the indices corresponding to the input wires of C.

- $\mathsf{Eval}(\widetilde{C}, \{\mathsf{key}_{w,x_w}\}_{w \in \mathsf{inp}(C)}) \rightarrow y$: On input a garbled circuit $\widetilde{C}$ and a garbled input $\{\mathsf{key}_{w,x_w}\}_{w \in \mathsf{inp}(C))}$, the evaluation algorithm outputs a string y.

Moreover, Π_{GC} should satisfy the following properties:

- **Correctness:** For all security parameters $\lambda \in \mathbb{N}$, all circuits C with input length $m = \mathsf{poly}(\lambda)$, all inputs $x \in \{0,1\}^m$, it holds that

$$\Pr[C(x) = \mathsf{Eval}(\widetilde{C}, \{\mathsf{key}_{w,x_w}\}_{w \in \mathsf{inp}(C)})] = 1,$$

where $\widetilde{C} \leftarrow \mathsf{Garble}(1^\lambda, C, \{\mathsf{key}_{w,b}\}_{w \in \mathsf{inp}(C), b \in \{0,1\}})$.

- **Security:** There exists an efficient algorithm GCSim such that for all circuits C with input length $m = \mathsf{poly}(\lambda)$, all inputs $x \in \{0,1\}^m$, and uniformly-random keys $\{\mathsf{key}_{w,b}\}_{w \in \mathsf{inp}(C), b \in \{0,1\}}$, it holds that the distributions

$$(\widetilde{C}, \{\mathsf{key}_{w,x_w}\}_{w \in \mathsf{inp}(C))}) \text{ and } \mathsf{GCSim}(1^\lambda, C, y)$$

are computationally indistinguishable, where $\widetilde{C} \leftarrow \mathsf{Garble}(1^\lambda, C, \{\mathsf{key}_{w,b}\}_{w \in \mathsf{inp}(C), b \in \{0,1\}})$ and $y = C(x)$.

Simplified garbled circuit notation. For simplicity in the following construction, we will write Keys to denote the list of all input labels $\{\mathsf{key}_{w,b}\}_{w \in \mathsf{inp}(C), b \in \{0,1\}}$ and Keys_x to denote the labels $\{\mathsf{key}_{w,x_w}\}_{w \in \mathsf{inp}(C))}$ associated with the input x.

B.1 RAM Model of Computation

We briefly define the RAM model of computation. Parts of this subsection are taken from [CDG+17]. For our purposes, we only need to support read operations.

Notation. The RAM model consists of a CPU and a memory storage of size M. The CPU executes a program that can access the memory by using read operations. In particular, for a program P with memory of size M we denote the contents of the memory data by $D \in \{0,1\}^M$. Additionally, the program gets a "short" input $x \in \{0,1\}^m$, which is also considered the initial state of the program. We use $P^D(x)$ to denote the execution of P with memory contents D and input x. The program P can read from various locations in D throughout its execution.

CPU-step circuit. We represent a RAM program P via t CPU-step circuits, each of which executes a single CPU step. Each CPU step is denoted by:

$$C^P_{\mathsf{CPU}}(\mathsf{st}, \mathsf{rData}) \longrightarrow (\mathsf{st}', L)$$

The circuit takes as input the current CPU state st and a bit rData. The bit rData will be read from the memory location that was requested by the previous CPU step. The circuit outputs an updated state st′ and the next location to read from $L \in [M]$. The sequence of locations form the memory access pattern $\mathsf{MemAccess} = \{L^1, \ldots, L^t\}$.

Representing RAM computation by CPU-step circuits. The computation $P^D(x)$ starts with an initial state $\mathsf{st}_1 = x$. In each step $\tau \in \{1, \ldots, t\}$, the computation proceeds as follows: If $\tau = 1$, then $\mathsf{rData}^\tau := 0$; otherwise $\mathsf{rData}^\tau := D[L^{\tau-1}]$. Next it executes the CPU-step circuit $C^P_{\mathsf{CPU}}(\mathsf{st}^\tau, \mathsf{rData}^\tau) = (\mathsf{st}^{\tau+1}, L^\tau)$. When $\tau = t$, $\mathsf{st}^{\tau+1}$ is the output of the program.

B.2 Construction of NISC in the RAM Model

In this section, we now recall the NISC construction from [CDG+17]. As noted above, we only need to support read-only RAM machines.

Construction B.3 (NISC in the RAM Model [CDG+17, adapted])**.** The construction relies on the following primitives:

- Let $\Pi_{\mathsf{OT}} = (\mathsf{OT.Setup}, \mathsf{OT.Hash}, \mathsf{OT.Send}, \mathsf{OT.Receive})$ be a laconic OT scheme.
- Let $\Pi_{\mathsf{GC}} = (\mathsf{Garble}, \mathsf{Eval})$ be a circuit garbling scheme with key length $k = k(\lambda)$.

We construct the NISC scheme $\Pi_{\mathsf{NISC}} = (\mathsf{Setup}, \mathsf{EncData}, \mathsf{EncProg}, \mathsf{Decrypt})$ as follows:

- $\mathsf{Setup}(1^\lambda)$: On input the security parameter λ, the setup algorithm samples $\mathsf{crs} \leftarrow \mathsf{OT.Setup}(1^\lambda)$ and outputs crs.
- $\mathsf{EncData}(\mathsf{crs}, D)$: On input the common reference string crs and a database D, the data encryption algorithm computes $(\mathsf{dig}, \widehat{D}) \leftarrow \mathsf{OT.Hash}(\mathsf{crs}, D)$ and outputs $(\mathsf{dig}, \widehat{D})$.
- $\mathsf{EncProg}(\mathsf{crs}, \mathsf{dig}, (P, x, t))$: On input the common reference string crs, a digest dig, and a program P with input $x \in \{0,1\}^m$ and maximum run-time t, the program encryption algorithm does the following:
  1. For all $\tau \in [t+1]$, sample $(\mathsf{stKeys}^\tau, \mathsf{dataKeys}^\tau, \mathsf{digKeys}^\tau) \xleftarrow{\mathrm{R}} \{0,1\}^{(2m+2+2|\mathsf{dig}|)k}$.

2. For all $\tau \in [t]$, compute $\widetilde{C}^{\tau}_{\mathsf{step}} \leftarrow \mathsf{Garble}(1^\lambda, C_{\mathsf{step}}[\mathsf{crs}, P, \mathsf{Keys}^{\tau+1}], \mathsf{Keys}^{\tau})$, where we have $\mathsf{Keys}^{\tau} = (\mathsf{stKeys}^{\tau}, \mathsf{dataKeys}^{\tau}, \mathsf{digKeys}^{\tau})$ and $C_{\mathsf{step}}[\mathsf{crs}, P, \mathsf{nextKeys}]$ is defined as follows:

Hard-wired: CRS crs, program P, keys $\mathsf{nextKeys} = (\mathsf{stKeys}, \mathsf{dataKeys}, \mathsf{digKeys})$
Input: state st, data rData, digest dig

(a) Compute $(\mathsf{st}', L) \leftarrow C^{P}_{\mathsf{CPU}}(\mathsf{st}, \mathsf{rData})$.

(b) Compute $\mathsf{ct}_L \leftarrow \mathsf{OT.Send}(\mathsf{crs}, \mathsf{dig}, L, \mathsf{dataKeys})$.

Output $((\mathsf{stKeys}_{\mathsf{st}'}, \mathsf{ct}_L, \mathsf{digKeys}_{\mathsf{dig}}), L)$

Figure 10: Description of step circuit $C_{\mathsf{step}}[\mathsf{crs}, P, \mathsf{nextKeys}]$.

For $\tau = 1$, embed labels $\mathsf{digKeys}^1_{\mathsf{dig}}$,$\mathsf{dataKeys}^1_0$, and stKeys^1_x in $\widetilde{C}^1_{\mathsf{step}}$.

Output $\mathsf{ct} = (\{\widetilde{C}^{\tau}_{\mathsf{step}}\}_{\tau \in [t]}, \mathsf{stKeys}^{t+1})$.

- $\mathsf{Decrypt}^{\widetilde{D}}(\mathsf{crs}, \mathsf{ct})$: On input the CRS crs and a ciphertext $\mathsf{ct} = (\{\widetilde{C}^{\tau}_{\mathsf{step}}\}_{\tau \in [t]}, \mathsf{stKeys}^{t+1})$, the decryption algorithm does the following:
  1. Parse $\widetilde{C}^1_{\mathsf{step}} = (\widetilde{C}^1_{\mathsf{step}}, \mathsf{digLabels}^1, \mathsf{dataLabels}^1, \mathsf{stLabels}^1)$.
  2. For each $\tau \in [t]$ in ascending order do the following:
     (a) Compute $(X, L) \leftarrow \mathsf{Eval}(\widetilde{C}^{\tau}_{\mathsf{step}}, (\mathsf{stLabels}^{\tau}, \mathsf{dataLabels}^{\tau}, \mathsf{digLabels}^{\tau}))$.
     (b) Parse $X = (\mathsf{stLabels}^{\tau+1}, \mathsf{ct}_L, \mathsf{digLabels}^{\tau+1})$ and compute
     $$\mathsf{dataLabels}^{\tau+1} \leftarrow \mathsf{OT.Receive}^{\widetilde{D}}(\mathsf{crs}, \mathsf{ct}_L, L).$$

  Output y by using stKeys^{t+1} to decode $\mathsf{stLabels}^{t+1}$.

Theorem B.4 (Correctness)**.** *Suppose Π_{GC} and Π_{OT} satisfy correctness. Then, Construction B.3 is correct.*

Proof. Take any security parameter $\lambda \in \mathbb{N}$, database $D \in \{0,1\}^M$, and RAM program tuple (P, x, t). Let $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$, $(\mathsf{dig}, \widehat{D}) \leftarrow \mathsf{EncData}(\mathsf{crs}, D)$, and $\mathsf{ct} \leftarrow \mathsf{EncProg}(\mathsf{crs}, \mathsf{dig}, (P, x, t))$. When evaluating $\mathsf{Decrypt}^{\widehat{D}}(\mathsf{crs}, \mathsf{ct})$, Π_{GC} correctness ensures that each step outputs the correct labels for the next step, while Π_{OT} correctness ensures that the correct data labels are retrieved. At the end of this evaluation process, the keys given in ct can decode the final state of the CPU-step circuit, which is $y = P^D(x)$. □

Theorem B.5 (Privacy)**.** *Suppose Π_{OT} satisfies sender privacy and Π_{GC} is secure. Then, Construction B.3 is private.*

Proof. We first define the simulator SimEnc. On input the common reference string crs, a digest dig, a database D, an output string y, and a memory access pattern $\mathsf{MemAccess} = \{L^1, \ldots, L^t\}$, the simulator does the following:

1. Sample hard-wired keys $(\mathsf{stKeys}^{t+1}, \mathsf{dataKeys}^{t+1}, \mathsf{digKeys}^{t+1}) \xleftarrow{\mathrm{R}} \{0,1\}^{(2m+2+2|\mathsf{dig}|)k}$ for C_{step} and set output labels to $\mathsf{stLabels}^{t+1} \leftarrow \mathsf{stKeys}^{t+1}_y$, $\mathsf{dataLabels}^{t+1} \leftarrow \mathsf{dataKeys}^{t+1}_{D[L^t]}$, $\mathsf{digLabels}^{t+1} \leftarrow \mathsf{digKeys}^{t+1}_{\mathsf{dig}}$.

2. For $\tau = t, t-1, \ldots, 1$, proceed as follows:
   (a) Compute $\mathsf{ct}_{L^\tau} \leftarrow \mathsf{OTSim}(\mathsf{crs}, D, L^\tau, \mathsf{dataLabels}^{\tau+1})$ and set $X \leftarrow (\mathsf{stLabels}^{\tau+1}, \mathsf{ct}_{L^\tau}, \mathsf{digLabels}^{\tau+1})$.
   (b) Compute $(\widetilde{C}^\tau_{\mathsf{step}}, \mathsf{stLabels}^\tau, \mathsf{dataLabels}^\tau, \mathsf{digLabels}^\tau) \leftarrow \mathsf{GCSim}(1^\lambda, C_{\mathsf{step}}, (X, L^\tau))$.

   Output $\mathsf{ct} = (\{\widetilde{C}^\tau_{\mathsf{step}}\}_{\tau\in[t]}, \mathsf{stKeys}^{t+1})$.

To show SimEnc indeed satisfies privacy, we define a sequence of hybrid experiments:

- Hyb_{2i}: This is the privacy game from Definition A.1, with the following modified ct generation procedure in Step 4:
  1. Execute $P^D(x)$ to obtain $\mathsf{MemAccess} = \{L^1, \ldots, L^t\}$ and set $y \leftarrow \mathsf{st}^{t+1}$. Additionally, compute rData^τ at the beginning of step τ for $\tau \in [t+1]$.
  2. For $\tau = t+1$ down to $i+1$, sample $(\mathsf{stKeys}^\tau, \mathsf{dataKeys}^\tau, \mathsf{digKeys}^\tau) \xleftarrow{\mathrm{R}} \{0,1\}^{(2m+2+2|\mathsf{dig}|)k}$.
  3. For $\tau = t$ down to $i+1$, compute $\widetilde{C}^\tau_{\mathsf{step}} \leftarrow \mathsf{Garble}(1^\lambda, C_{\mathsf{step}}[\mathsf{crs}, P, \mathsf{Keys}^{\tau+1}], \mathsf{Keys}^\tau)$, where we have $\mathsf{Keys}^\tau = (\mathsf{stKeys}^\tau, \mathsf{dataKeys}^\tau, \mathsf{digKeys}^\tau)$ and C_{step} is defined as in Fig. 10. Set $\mathsf{stLabels}^{i+1} \leftarrow \mathsf{stKeys}^{i+1}_{\mathsf{st}^{i+1}}$, $\mathsf{dataLabels}^{i+1} \leftarrow \mathsf{dataKeys}^{i+1}_{\mathsf{rData}^{i+1}}$, and $\mathsf{digLabels}^{i+1} \leftarrow \mathsf{digKeys}^{i+1}_{\mathsf{dig}}$.
  4. For $\tau = i$ down to 1, proceed as follows:
     (a) Compute the ciphertext $\mathsf{ct}_{L^\tau} \leftarrow \mathsf{OTSim}(\mathsf{crs}, D, L^\tau, \mathsf{dataLabels}^{\tau+1})$ and set the tuple $X \leftarrow (\mathsf{stLabels}^{\tau+1}, \mathsf{ct}_{L^\tau}, \mathsf{digLabels}^{\tau+1})$.
     (b) Compute $(\widetilde{C}^\tau_{\mathsf{step}}, \mathsf{stLabels}^\tau, \mathsf{dataLabels}^\tau, \mathsf{digLabels}^\tau) \leftarrow \mathsf{GCSim}(1^\lambda, C_{\mathsf{step}}, (X, L^\tau))$.

     Embed $\mathsf{digLabels}^1, \mathsf{dataLabels}^1, \mathsf{stLabels}^1$ in $\widetilde{C}^1_{\mathsf{step}}$.

  Output $\mathsf{ct} = (\{\widetilde{C}^\tau_{\mathsf{step}}\}_{\tau\in[t]}, \mathsf{stKeys}^{t+1})$.

- Hyb_{2i+1}: Same as Hyb_{2i} except

$$(\widetilde{C}^{i+1}_{\mathsf{step}}, \mathsf{stLabels}^{i+1}, \mathsf{dataLabels}^{i+1}, \mathsf{digLabels}^{i+1}) \leftarrow \mathsf{GCSim}(1^\lambda, C_{\mathsf{step}}, ((\mathsf{stKeys}^{i+2}_{\mathsf{st}^{i+2}}, \mathsf{ct}_{L^{i+1}}, \mathsf{digKeys}^{i+2}_{\mathsf{dig}}), L^{i+1})),$$

  where $\mathsf{ct}_{L^{i+1}} \leftarrow \mathsf{OT.Send}(\mathsf{crs}, \mathsf{dig}, L^{i+1}, \mathsf{dataKeys}^{i+2})$ and $(\mathsf{stKeys}^{i+2}, \mathsf{dataKeys}^{i+2}, \mathsf{digKeys}^{i+2})$ are the input keys to $\widetilde{C}^{i+2}_{\mathsf{step}}$.

Note that Hyb_0 is the privacy game with EncProg used to generate the challenge ciphertext, and Hyb_{2t} is the privacy game with SimEnc used to generate the challenge ciphertext. We will appeal to security of Π_{GC} to show that Hyb_{2i} and Hyb_{2i+1} are computationally indistinguishable for all $i \in [0, t-1]$. We will then appeal to sender privacy of Π_{OT} to show that Hyb_{2i+1} and Hyb_{2i+2} are computationally indistinguishable for all $i \in [0, t-1]$.

Lemma B.6. *Suppose Π_{GC} satisfies security. Then, for all $i \in [0, t-1]$, Hyb_{2i} and Hyb_{2i+1} are computationally indistinguishable.*

Proof. Suppose there exists an efficient adversary $\mathcal{A}$ that distinguishes Hyb_{2i} and Hyb_{2i+1} with non-negligible probability δ. We use $\mathcal{A}$ to construct $\mathcal{B}$ that breaks security of Π_{GC}:

1. At the beginning of the game, $\mathcal{B}$ gets $D \in \{0,1\}^M$ and (P, t) from $\mathcal{A}$. Algorithm $\mathcal{B}$ samples $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$, computes $(\mathsf{dig}, \widetilde{D}) \leftarrow \mathsf{EncData}(\mathsf{crs}, D)$, and gives $(\mathsf{crs}, \mathsf{dig}, \widetilde{D})$ to $\mathcal{A}$.

2. Given input x from $\mathcal{A}$, algorithm $\mathcal{B}$ runs the ct generation procedure as in Hyb_{2i}, except the components $(\widetilde{C}^{i+1}_{\mathsf{step}}, \mathsf{stLabels}^{i+1}, \mathsf{dataLabels}^{i+1}, \mathsf{digLabels}^{i+1})$ are generated by sending $C_{\mathsf{step}}[\mathsf{crs}, P, \mathsf{Keys}^{i+2}]$ and input $(\mathsf{st}^{i+1}, \mathsf{rData}^{i+1}, \mathsf{dig})$ to the garbled circuit challenger and using the response.

3. Algorithm $\mathcal{B}$ gives ct to $\mathcal{A}$ and outputs whatever $\mathcal{A}$ outputs.

If $(\widetilde{C}^{i+1}_{\mathsf{step}}, \mathsf{stLabels}^{i+1}, \mathsf{dataLabels}^{i+1}, \mathsf{digLabels}^{i+1})$ from the garbled circuit challenger are generated honestly, $\mathcal{B}$ simulates Hyb_{2i} for $\mathcal{A}$. If the components from the garbled circuit challenger are generated via GCSim, $\mathcal{B}$ simulates Hyb_{2i+1} for $\mathcal{A}$. Thus, $\mathcal{B}$ breaks security of Π_{GC} with advantage δ, as desired. □

Lemma B.7. *Suppose Π_{OT} satisfies sender privacy. Then, for all $i \in [0, t-1]$, Hyb_{2i+1} and Hyb_{2i+2} are computationally indistinguishable.*

Proof. Suppose there exists an efficient adversary $\mathcal{A}$ that distinguishes Hyb_{2i+1} and Hyb_{2i+2} with non-negligible probability δ. We use $\mathcal{A}$ to construct $\mathcal{B}$ that breaks sender privacy of Π_{OT}:

1. At the beginning of the game, algorithm $\mathcal{B}$ gets $D \in \{0,1\}^M$ and (P, t) from $\mathcal{A}$.

2. For $\tau = t+1$ down to $i+2$, algorithm $\mathcal{B}$ samples $(\mathsf{stKeys}^{\tau}, \mathsf{dataKeys}^{\tau}, \mathsf{digKeys}^{\tau}) \xleftarrow{\mathrm{R}} \{0,1\}^{(2m+2+2|\mathsf{dig}|)k}$. Algorithm $\mathcal{B}$ also samples $L^{i+1} \xleftarrow{\mathrm{R}} [M]$ and sends $(D, L^{i+1}, (m_0, m_1) = \mathsf{dataKeys}^{i+2})$ to the sender privacy challenger to get back $(\mathsf{crs}, \mathsf{ct}_{L^{i+1}})$. Algorithm $\mathcal{B}$ computes $(\mathsf{dig}, \widetilde{D}) \leftarrow \mathsf{EncData}(\mathsf{crs}, D)$ and sends $(\mathsf{crs}, \mathsf{dig}, \widetilde{D})$ to $\mathcal{A}$.

3. Given input x from algorithm $\mathcal{A}$, algorithm $\mathcal{B}$ executes $P^D(x)$ to obtain MemAccess and check if L^{i+1} matches the $i+1$th entry of MemAccess. If the entries do not match, $\mathcal{B}$ outputs $\perp$. Otherwise, $\mathcal{B}$ runs the ct generation as in Hyb_{2i+2} with the crs, $\mathsf{ct}_{L^{i+1}}$, and $(\mathsf{stKeys}^{\tau}, \mathsf{dataKeys}^{\tau}, \mathsf{digKeys}^{\tau})_{\tau \in [i+2, t+1]}$ components as above.

4. $\mathcal{B}$ gives ct to $\mathcal{A}$ and outputs whatever $\mathcal{A}$ outputs.

If the tuple $(\mathsf{crs}, \mathsf{ct}_{L^{i+1}})$ from the sender privacy challenger is generated honestly, $\mathcal{B}$ simulates Hyb_{2i+1} for $\mathcal{A}$ when it does not abort. If $(\mathsf{crs}, \mathsf{ct}_{L^{i+1}})$ is generated with OTSim, $\mathcal{B}$ simulates Hyb_{2i+2} for $\mathcal{A}$ when it does not abort. Thus, $\mathcal{B}$ breaks sender privacy of Π_{OT} with advantage δ conditioned on the guess for L^{i+1} being correct. Since L^{i+1} is sampled uniformly and is independent of $\mathcal{A}$'s view, $\mathcal{B}$ breaks sender privacy with advantage at least δ/M, which is still non-negligible since $M = \mathrm{poly}(\lambda)$. □

Combining Lemmas B.6 and B.7 yields the statement by a hybrid argument. □

Theorem B.8 (Efficiency). *Suppose Π_{OT} satisfies efficiency. Then, Construction B.3 is efficient.*

Proof. Follows immediately from the efficiency of Π_{OT}. □