Holographic SNARGs for P and Batch-NP from (Polynomially Hard) Learning with Errors

Susumu Kiyoshima

NTT Social Informatics Laboratories
susumu.kiyoshima@ntt.com

September 28, 2023

Abstract

A succinct non-interactive argument (SNARG) is called *holographic* if the verifier runs in time sub-linear in the input length when given oracle access to an encoding of the input. We present holographic SNARGs for P and Batch-NP under the learning with errors (LWE) assumption. Our holographic SNARG for P has a verifier that runs in time $\mathsf{poly}(\lambda, \log T, \log n)$ for T-time computations and n-bit inputs (λ is the security parameter), while our holographic SNARG for Batch-NP has a verifier that runs in time $\mathsf{poly}(\lambda, T, \log k)$ for k instances of T-time computations. Before this work, constructions with the same asymptotic efficiency were known in the designated-verifier setting or under the sub-exponential hardness of the LWE assumption. We obtain our holographic SNARGs (in the public-verification setting under the polynomial hardness of the LWE assumption) by constructing holographic SNARGs for certain hash computations and then applying known/trivial transformations.

As an application, we use our holographic SNARGs to weaken the assumption needed for a recent public-coin 3-round zero-knowledge (ZK) argument [Kiyoshima, CRYPTO 2022]. Specifically, we use our holographic SNARGs to show that a public-coin 3-round ZK argument exists under the same assumptions as the state-of-the-art private-coin 3-round ZK argument [Bitansky et al., STOC 2018].

This article is a full version of the following article: Holographic SNARGs for P and Batch-NP from (Polynomially Hard) Learning with Errors, TCC 2023, ©IACR 2023. This work was done while the author was a member of NTT Research.

Contents

1 Introduction

SNARGs. Informally speaking, a *succinct argument* [Kil92] (or *delegation scheme* [GKR15]) is an argument system with small communication complexity and fast verification time. In a typical setting, the statement to be proven contains a description of a computation and an input to the computation; the prover's task is to convince the verifier that the output of the computation is 1. A typical efficiency requirement is that the communication complexity and the verification time are polylogarithmic in the complexity of the computation.

When succinct arguments are non-interactive in the common random/reference string (CRS) model, they are commonly referred to as *succinct non-interactive arguments (SNARGs)*. Initially, the study of SNARGs was focused on constructing SNARGs for all NP computations (i.e., the goal was to design SNARGs that can prove the correctness of any NP computations). However, for such a large class of computations, positive results were only obtained in the idealized model (e.g., the random oracle model) or under non-standard cryptographic assumptions (e.g., extractability assumptions). Recently, a growing number of works showed that SNARGs for useful subclasses of NP computations, such as deterministic computations and Batch-NP computations,[1] can be obtained in the standard model under standard cryptographic assumptions. Specifically, Kalai, Raz, and Rothblum [KRR22] and subsequent works (e.g., [KP16, BHK17, BKK+18, HR18]) constructed *designated-verifier SNARGs* for such subclasses under the learning with errors (LWE) assumption. (Designated-verifier SNARGs are weaker than standard SNARGs in that they are not publicly verifiable, i.e., only the owner of a secret verification key can verify the correctness of proofs.) More recently, Choudhuri, Jain, and Jin [CJJ22] and other works (e.g., [JKKZ21, CJJ21a, KVZ21, HJKS22, WW22, DGKV22, PP22]) constructed (publicly verifiable) SNARGs for such subclasses under various standard cryptographic assumptions (e.g., the LWE assumption).

Holographic SNARGs. SNARGs are called *holographic* if the verifier runs in time sub-linear in the length of the input when given oracle access to an encoding of the input.[2] The holographic property is often naturally satisfied when SNARGs are constructed based on code-theoretic techniques. It also often comes with additional useful properties such as (i) input encoding having a simple algebraic structure (e.g., a low-degree polynomial) and (ii) verification that only makes non-adaptive queries to the encoding of the input.

The holographic property of existing SNARGs has been used crucially in some applications. For example, in the application to 2-message arguments of proximity [KR15], the holographic property of the underlying SNARG [KRR22] was used to reduce the task of proving the correctness of an arbitrary deterministic computation to a much simpler task of proving that the encoding of the computation input has certain values at certain positions. Other examples include applications to succinct probabilistically checkable arguments [BR22] and 3-round zero-knowledge argument [BKP18, Kiy22a], where the holographic property of the underlying SNARGs was used to have succinct verification in the setting where the verification cannot read the entire input.

Existing holographic SNARGs are, however, less powerful than state-of-the-art non-holographic SNARGs. Concretely, compared with the non-holographic SNARGs of Choudhuri et al. [CJJ22] and subsequent works [KVZ21, WW22, DGKV22, PP22] (which are publicly verifiable, can be used for deterministic polynomial-time computations and Batch-NP computations, and are based on standard polynomial hardness assumptions), existing holographic SNARGs are either (i) not publicly verifiable [KRR22, BHK17] or (ii) based on sub-exponential hardness assumptions [JKKZ21, Kiy22a].

Because of this gap, some applications of holographic SNARGs only obtained sub-optimal results. For example, in the application to 3-round zero-knowledge arguments [BKP18, Kiy22a], the existing constructions are either (i) private-coin (as the underlying holographic SNARG [KRR22] is not publicly verifiable) or (ii) based on a sub-exponential hardness assumption (as the underlying holographic SNARG [JKKZ21] is based on a sub-exponential hardness assumption).

[1] In SNARGs for Batch-NP computations, a statement consists of multiple instances of an NP language, and the prover tries to convince the verifier that all the instances belong to the language. The communication complexity and the verification time are required to be smaller than the naive check.

[2] The holographic property has also been considered for *interactive oracle proofs* (IOPs) [CHM+20, COS20] and interactive proofs/arguments [GR17, BR22]. The term "holography" was initially used in the context of *probabilistically checkable proofs* (PCPs) [BFLS91].

1.1 Our Results

We give holographic SNARGs for deterministic polynomial-time computations and Batch-NP computations under the polynomial hardness of the LWE assumption. The holographic verifier of our SNARGs makes non-adaptive queries to the *low-degree extension* (LDE) of the computation input.[3]

Theorem (informal, see Theorem 5 and Corollary 2)**.** *Under the LWE assumption, there exist holographic SNARGs for deterministic polynomial-time computations and Batch-NP computations.*

- *For security parameter λ and any deterministic T-time computation with input length n, the CRS generation algorithm runs in time $\mathsf{poly}(\lambda, \log T, \log n)$, the prover runs in time $\mathsf{poly}(\lambda, T, n)$, and the verifier runs in time $\mathsf{poly}(\lambda, \log T, \log n)$ given oracle access to the LDE of the computation input.*
- *For security parameter λ and any Batch-NP computation that consists of k instances of a T-time nondeterministic computation, the CRS generation algorithm runs in time $\mathsf{poly}(\lambda, T, \log k)$, the prover runs in time $\mathsf{poly}(\lambda, T, k)$, and the verifier runs in time $\mathsf{poly}(\lambda, T, \log k)$ given oracle access to the LDE of the concatenation of the k instances.*

Given the LDE of a long input, our SNARG verifiers run in time sub-linear in the input length (which is n in deterministic computations and is linear in k in Batch-NP computations).

At a high level, our result can be seen as a holographic version of the result of Choudhuri, Jain, and Jin [CJJ22] (where non-holographic SNARGs for deterministic computations and Batch-NP computations are given under the LWE assumption). However, in the adaptive-statement setting, our holographic soundness notions are weaker than the non-holographic counterparts since the input to be encoded is required to be fixed non-adaptively before the CRS is sampled (see, e.g., Definition 14 for the soundness notion that our holographic SNARG for Batch-NP satisfies).

As an application of the above result, we give the following result about 3-round zero-knowledge arguments.

Theorem (informal, see Theorem 6)**.** *Assume the existence of (polynomially compressing) keyless hash functions that are multi-collision resistant against slightly super-polynomial-time adversaries, and additionally assume slightly super-polynomial hardness of the LWE assumption. Then, there exists a public-coin 3-round zero-knowledge argument for NP.*

We obtain this result by relying on a known transformation [BKP18, Kiy22a]. The assumptions in this result are the same as those that are needed for the state-of-the-art private-coin 3-round zero-knowledge argument [BKP18], and they are weaker than those that are needed for the state-of-the-art public-coin 3-round zero-knowledge argument [Kiy22a] (concretely, sub-exponential hardness is not necessary for the LWE assumption). In short, this result closes the gap between the assumptions needed for private-coin 3-round zero-knowledge arguments and those needed for public-coin ones.

1.2 Related Work

The following is a brief review of existing SNARGs that have verification time sub-linear in the input length. (We focus on those that are publicly verifiable and based on well-studied falsifiable assumptions.) For deterministic computations, the aforementioned work by Choudhuri et al. [CJJ22] constructed a *SNARG for RAM computations* [CKLR11, KP16], where the verifier runs in time sub-linear in the length of the initial memory when given a short digest of the initial memory. (The digest can be computed by the verifier in a one-time expensive preprocessing phase.) For Batch-NP computations, some of the existing constructions (e.g., [CJJ22, WW22]) have the verifier that first runs in an expensive offline phase and subsequently checks the proof in time sub-linear in the number of the instances. As a special case of SNARGs for Batch-NP, the work by Choudhuri et al. [CJJ22] defined and constructed a *batch argument (BARG) for the index language*, where the statement to be proven is restricted to the form $\forall i \in [k] \exists w_i \in \{0,1\}^*$ s.t. $C(i, w_i) = 1$ for a Boolean circuit C, and the verifier runs in time sub-linear in k if C is of size sub-linear in k. (A similar notion was also considered by Kalai, Vaikuntanathan, and Zhang [KVZ21].) As a related notion, Devadas, Goyal, Kalai, and Vaikuntanathan [DGKV22] considered *hashed BARGs*, where the verifier runs in sub-linear time when given a (*somewhere extractable* [HW15, CJJ22]) hash of the instances.

[3] For the definition of low-degree extensions, see Section 3.2.

Our holographic SNARGs are incomparable to the above-mentioned SNARGs. For example, the input is encoded information theoretically and does not require any setup from the verifier. This property is essential in the application to 3-round ZK arguments [BKP18, Kiy22a] since the zero-knowledge is proved using a simulator that commits to the encoded input of the holographic SNARG in the first round (without any setup from the verifier).

2 Technical Overviews

2.1 Holographic SNARG for P

Overall approach. Our starting point is the work by Choudhuri, Jain, and Jin (CJJ) [CJJ22], which gives a (non-holographic) SNARG for deterministic polynomial-time computation under the LWE assumption. In the CJJ SNARG, the computation to be proven is modeled as a RAM computation, and the input to the computation is viewed as the initial memory. Importantly, the verifier only uses the input to obtain the digest of the initial memory, and the digest is simply the Merkle tree-hash of the input. Thus, the CJJ SANRG is non-holographic only because the verifier needs to compute the Merkle tree-hash of the input.[4]

Recently, it was shown that the CJJ SNARG can be made holographic under the sub-exponential hardness of the LWE assumption [Kiy22a]. The main idea was to delegate the computation of the Merkle tree-hash to the prover. That is, the prover was modified to additionally create the Merkle tree-hash of the input along with a holographic proof about the correctness of the Merkle tree-hash. The key point was that the correctness of Merkle tree-hash computations can be proved in a holographic way using the SNARG by Jawale, Kalai, Khurana, and Zhang (JKKZ) [JKKZ21] (which is holographic and can be used for any log-uniform bounded-depth deterministic computations). This approach requires the sub-exponential hardness of the LWE assumption since the JKKZ SNARG is based on the sub-exponential hardness of the LWE assumption.

We make the CJJ SNARG holographic under the polynomial hardness of the LWE assumption. Our approach is to design a holographic SNARG that is tailored to Merkle tree-hash computations, expecting that such a restricted SNARG is easier to construct under the polynomial hardness assumption. Concretely, our target is a SNARG such that (i) for a statement of the form (x, h, rt), the prover can prove that $\mathsf{rt} \in \{0,1\}^\lambda$ is the Merkle tree-hash of $x \in \{0,1\}^*$ under the hash function $h : \{0,1\}^{2\lambda} \to \{0,1\}^\lambda$, (ii) the soundness holds even against cheating provers that adaptively choose rt, and (iii) the verifier runs in time sub-linear in $|x|$ when it is given oracle access to an encoding of x.

We achieve our goal in two steps. First, we obtain a holographic SNARG for tree-hash computations under the LWE assumption while assuming the existence of a holographic SNARG for another specific computation with a weak soundness guarantee. Next, we construct the required holographic SNARG under the LWE assumption. We use the following tools in both steps.

1. **Somewhere extractable (SE) hash functions [HW15, CJJ22]:** Like the Merkle tree-hash scheme, an SE hash function $\mathsf{SEH.Hash}$ with a random public key h can create a hash value $\mathsf{rt} = \mathsf{SEH.Hash}(h, x)$ and short certificates $\{\mathsf{cert}_i\}_{i\in[N]}$ for a long message $x = (x_1, \ldots, x_N)$. Moreover, for any $i^* \in [N]$, we can sample h with trapdoor information td so that the following hold.

   - *Somewhere extractability.* With overwhelming probability over the sampling of h, any hash value rt uniquely determines its pre-image in position i^*. Furthermore, a PPT extractor $\mathsf{SEH.Extract}$ can extract the unique i^*-th position value v given $(\mathsf{td}, \mathsf{rt})$.
   - *Index hiding.* The public key h computationally hides the binding index i^*.

   An SE hash function can be obtained under the LWE assumption [HW15]. It can be generalized to support multiple binding indices naturally, and its complexity (such as the description size of the hash function) scales linearly in the number of binding indices.

2. **Batch arguments (BARGs) for the index language [CJJ22]:** BARGs for the index language are a special case of SNARGs for Batch-NP computations: for a Boolean circuit C and an integer $k \in \mathbb{N}$, the statement

[4] Most of the existing schemes (e.g., [KVZ21, HJKS22, WW22, PP22]) are also (implicitly) designed for RAM computations and are non-holographic for the same reason.

Building blocks: (i) $\mathsf{SEH.Hash}$—an SE hash function. (ii) $\mathsf{BARG}_{\mathrm{idx}}$—a BARG for the index language.

CRS generation: The CRS generation algorithm samples λ public keys $h_0^{\mathsf{SEH}}, \ldots, h_{\lambda-1}^{\mathsf{SEH}}$ of $\mathsf{SEH.Hash}$ and a CRS of $\mathsf{BARG}_{\mathrm{idx}}$.

Prover P: The prover P is given a binary string $x \in \{0,1\}^*$, a hash function $h : \{0,1\}^{2\lambda} \to \{0,1\}^\lambda$, and a hash value $\mathsf{rt} \in \{0,1\}^\lambda$.

1. Compute the nodes $\{\mathsf{node}_{i,\sigma}\}_{i\in\{0,\ldots,\ell\},\sigma\in\{0,\ldots,2^i-1\}}$ of the tree-hash of x using h. That is, do the following.
   (a) Partition x into 2^ℓ blocks $\mathsf{blk}_0, \ldots, \mathsf{blk}_{2^\ell-1}$ such that $|\mathsf{blk}_0| = \cdots = |\mathsf{blk}_{2^\ell-1}| = \lambda$. (We assume for simplicity that $|x| = 2^\ell\lambda$ for $\ell \in \mathbb{N}$.)
   (b) Let $\mathsf{node}_{\ell,\sigma} := \mathsf{blk}_\sigma$ for $\forall\sigma \in \{0,\ldots,2^\ell-1\}$, and let $\mathsf{node}_{i,\sigma} := h(\mathsf{node}_{i+1,2\sigma} \| \mathsf{node}_{i+1,2\sigma+1})$ for $\forall i \in \{\ell-1,\ldots,0\}, \sigma \in \{0,\ldots,2^i-1\}$.
2. Compute $\mathsf{rt}_i := \mathsf{SEH.Hash}(h_i^{\mathsf{SEH}}, x_i)$ for $\forall i \in \{0,\ldots,\ell\}$, where $x_i := \mathsf{node}_{i,0} \| \cdots \| \mathsf{node}_{i,2^i-1}$ is the concatenation of the nodes at the i-th level.
3. For $\forall i \in \{0,\ldots,\ell-1\}$, use $\mathsf{BARG}_{\mathrm{idx}}$ to compute a proof π_i about the consistency between rt_i and rt_{i+1}, i.e., prove that for $\forall\sigma \in \{0,\ldots,2^i-1\}$, there exists a pair of triples $(\mathsf{node}^{(\mathrm{P})}, \mathsf{node}^{(\mathrm{L})}, \mathsf{node}^{(\mathrm{R})})$, $(\mathsf{cert}^{(\mathrm{P})}, \mathsf{cert}^{(\mathrm{L})}, \mathsf{cert}^{(\mathrm{R})})$ s.t. (i) $\mathsf{cert}^{(\mathrm{P})}$ certifies that the σ-th position of the pre-image of rt_i is $\mathsf{node}^{(\mathrm{P})}$, (ii) $(\mathsf{cert}^{(\mathrm{L})}, \mathsf{cert}^{(\mathrm{R})})$ certifies that the 2σ-th and $2\sigma+1$-st positions of the pre-image of rt_{i+1} are $(\mathsf{node}^{(\mathrm{L})}, \mathsf{node}^{(\mathrm{R})})$, and (iii) $h(\mathsf{node}^{(\mathrm{L})} \| \mathsf{node}^{(\mathrm{R})}) = \mathsf{node}^{(\mathrm{P})}$.
4. Output $\pi := (\{\mathsf{rt}_i\}_{i\in\{0,\ldots,\ell\}}, \{\pi_i\}_{i\in\{0,\ldots,\ell-1\}})$ as a proof.

Verifier V: The verifier V is given (h, rt) and $\pi = (\{\mathsf{rt}_i\}_{i\in\{0,\ldots,\ell\}}, \{\pi_i\}_{i\in\{0,\ldots,\ell-1\}})$ along with oracle access to an encoding of x. Then, V outputs 1 iff (i) each π_i is an accepting w.r.t. $(\mathsf{rt}_i, \mathsf{rt}_{i+1})$ and (ii) $\mathsf{SEH.Hash}(h_0^{\mathsf{SEH}}, \mathsf{rt}) = \mathsf{rt}_0$.

Figure 1: A candidate construction of holographic tree-hash SNARGs

to be proven is that $\forall i \in [k] \exists w_i \in \{0,1\}^*$ s.t. $C(i, w_i) = 1$. (Unlike the general case, BARGs for the index language can have verification time sub-linear in k since the verifier no longer needs to take k instances explicitly.) A BARG for the index language can be obtained under the LWE assumption [CJJ22]. Its CRS generation and verifier run in time polylogarithmic in k, and it satisfies a stronger notion of soundness, *semi-adaptive somewhere soundness*, guaranteeing that for any $i^* \in [k]$, the CRS can be sampled so that giving an accepting proof for an adaptively chosen circuit C is infeasible as long as $\nexists w$ s.t. $C(i^*, w) = 1$.

Step 1. Holographic SNARG for tree-hashing from somewhere-sound holographic SNARG for SE-hashing. We construct a holographic SNARG for tree-hashing computations by using what we call a *somewhere-sound holographic SNARG for SE-hash computations*.

We start by giving an insecure candidate construction as a motivating example. Recall that in SNARGs for tree-hash computations, for a statement (x, h, rt), the prover proves that $\mathsf{rt} \in \{0,1\}^\lambda$ is the Merkle tree-hash of $x \in \{0,1\}^*$ under the hash function $h : \{0,1\}^{2\lambda} \to \{0,1\}^\lambda$, where the verifier is given oracle access to an encoding of x. At a high level, we consider a construction that follows the commit-and-prove paradigm: the prover (i) hashes all the nodes of the tree-hash of x in the layer-by-layer basis and (ii) uses a BARG for the index language to prove that the hashed nodes constitute a Merkle tree; the verifier verifies all the BARG proofs and checks whether the hash of the top layer is the hash of rt. See Figure 1 for a detailed description. Clearly, the construction in Figure 1 is not sound since nothing is proved about the consistency between x and the hash values. More concretely, the problem is that the verifier does not check whether rt_ℓ is the hash value of x w.r.t. the hash function h_ℓ^{SEH}. Since we want the verifier to be holographic, we cannot let the verifier check $\mathsf{SEH.Hash}(h_\ell^{\mathsf{SEH}}, x) \stackrel{?}{=} \mathsf{rt}_\ell$ directly. We thus augment the candidate construction with a holographic SNARG that proves the consistency between rt_ℓ and x.

To see in more detail what type of SNARGs is needed to make the candidate construction secure, assume that a cheating prover P^* breaks the soundness of the candidate construction. That is, given (x, h), the cheating prover P^* adaptively chooses rt and makes V output 1 despite $\mathsf{TreeHash}_h(x) \neq \mathsf{rt}$, where $\mathsf{TreeHash}_h(x)$ denotes the tree-hash of x under the hash function h. (For simplicity, we assume that P^* succeeds with probability 1.)

Let $\{\mathsf{node}_{i,\sigma}\}_{i\in\{0,\ldots,\ell\},\sigma\in\{0,\ldots,2^i-1\}}$ denote the correct nodes of $\mathsf{TreeHash}_h(x)$. Then, we consider the following claim.

Claim 1 (Informal). *$\forall i \in \{0,\ldots,\ell\}$, $\exists\sigma \in \{0,\ldots,2^i-1\}$ s.t. if h_i^{SEH} is sampled in the CRS generation in a way that it is statistically binding in position σ, the hash value rt_i that P^* provides as a part of an accepting proof π satisfies $\mathsf{SEH.Extract}(\mathsf{td}_i^{\mathsf{SEH}},\mathsf{rt}_i) \neq \mathsf{node}_{i,\sigma}$,[5] where $\mathsf{td}_i^{\mathsf{SEH}}$ is the trapdoor corresponding to h_i^{SEH}.*

In words, this claim says that from each of the hash values $\{\mathsf{rt}_i\}_{i\in\{0,\ldots,\ell\}}$ that P^* provides as the layer-by-layer hashes of the nodes of $\mathsf{TreeHash}_h(x)$, we can extract a node that disagrees with the correct nodes $\{\mathsf{node}_{i,\sigma}\}_{i\in\{0,\ldots,\ell\},\sigma\in\{0,\ldots,2^i-1\}}$ in a certain position σ. We prove this claim by induction on i.

Base case. When $i = 0$ and $\sigma = 0$, the claim holds trivially because of our assumption that P^* breaks the soundness of the candidate construction.

Inductive step. Assume that the claim holds for $i-1$ and σ_{i-1}. The index hiding property of h_i^{SEH} guarantees that the claim remains to hold even when h_i^{SEH} is statistically binding in positions $2\sigma_{i-1}$ and $2\sigma_{i-1}+1$. Then, the semi-adaptive somewhere soundness of π_i guarantees that when we extract the statistically fixed values $\widetilde{\mathsf{node}}_{i-1,\sigma_{i-1}} := \mathsf{SEH.Extract}(\mathsf{td}_{i-1}^{\mathsf{SEH}},\mathsf{rt}_{i-1})$ and $(\widetilde{\mathsf{node}}_{i,2\sigma_{i-1}},\widetilde{\mathsf{node}}_{i,2\sigma_{i-1}+1}) := \mathsf{SEH.Extract}(\mathsf{td}_i^{\mathsf{SEH}},\mathsf{rt}_i)$, they satisfy $h(\widetilde{\mathsf{node}}_{i,2\sigma_{i-1}} \| \widetilde{\mathsf{node}}_{i,2\sigma_{i-1}+1}) = \widetilde{\mathsf{node}}_{i-1,\sigma_{i-1}}$. Since we assumed $\widetilde{\mathsf{node}}_{i-1,\sigma_{i-1}} \neq \mathsf{node}_{i-1,\sigma_{i-1}}$, we conclude that $\exists\sigma_i \in \{2\sigma_{i-1},2\sigma_{i-1}+1\} \subseteq \{0,\ldots,2^i-1\}$ s.t. $\widetilde{\mathsf{node}}_{i,\sigma_i} \neq \mathsf{node}_{i,\sigma_i}$ as desired.

For $i=\ell$, the above claim implies that a cheating prover can break the soundness of the candidate scheme only when $\exists\sigma\in\{0,\ldots,2^\ell-1\}$ s.t. the hash value rt_ℓ that the prover provides satisfies $\mathsf{Extract}(\mathsf{td}_\ell^{\mathsf{SEH}},\mathsf{rt}_\ell) \neq \mathsf{blk}_\sigma$, i.e., the extractor extracts a node that disagrees with the σ-th block of x. Thus, we need a holographic SNARG that prevents this type of inconsistency.

In conclusion, to make the candidate construction secure, we need a holographic SNARG that satisfies the following.

Holographic completeness. The prover can convince the verifier when given a statement (x,h,rt) s.t. $\mathsf{SEH.Hash}(h,x) = \mathsf{rt}$. The verifier is given (h,rt) as explicit inputs and is given oracle access to an encoding of x. The verifier runs in time sub-linear in $|x|$ (ideally, polylogarithmic in $|x|$).

Somewhere soundness. For any x and $\sigma\in\{0,\ldots,|x|-1\}$, and for an honest CRS and a random SE hash function key h that is statistically binding in position σ, no PPT prover can provide a hash value rt and an accepting proof satisfying $\mathsf{SEH.Extract}(\mathsf{td}^{\mathsf{SEH}},\mathsf{rt}) \neq x_\sigma$, where $\mathsf{td}^{\mathsf{SEH}}$ is the trapdoor corresponding to h.

Note that the somewhere soundness does not guarantee $\mathsf{SEH.Hash}(h,x)=\mathsf{rt}$. It only guarantees that for any σ, if we extract the σ-th position value from rt, the extracted value agrees with x. This guarantee is sufficient since the above claim guarantees that any successful cheating prover breaks this type of consistency in a randomly chosen σ with non-negligible probability. In the following, we explain how we obtain a holographic SNARG with the above properties under the LWE assumption.

Step 2. Somewhere sound holographic SNARG for SE-hashing from LWE. Similarly to recent SNARGs (e.g., [JKKZ21, CJJ22, KVZ21, HJKS22]), our construction is obtained by using a *correlation-intractable (CI) hash function* [CGH04]. Roughly speaking, a hash function family $\{\mathcal{H}_\lambda = \{h : X_\lambda \to Y_\lambda\}\}_{\lambda\in\mathbb{N}}$ is correlation intractable for a relation ensemble $\mathbf{R} = \{\mathbf{R}_\lambda \subseteq X_\lambda \times Y_\lambda\}_{\lambda\in\mathbb{N}}$ if no PPT adversary can find $x\in X_\lambda$ s.t. $(x,h(x))\in\mathbf{R}_\lambda$ for a random $h\in\mathcal{H}_\lambda$. It is known that a CI hash function family exists for *efficiently product verifiable relation ensembles* under the LWE assumption [HLR21b], where a relation $\mathbf{R}_\lambda \subseteq X_\lambda \times Y_\lambda^t$ is said to be efficiently product verifiable if we have $(x,(y_1,\ldots,y_t))\in\mathbf{R}_\lambda$ iff each y_i is included in a small efficiently verifiable set. More concretely, we use the following result [HLR21b].

[5] In this overview, we view rt_i as a hash of a vector that consists of 2^i blocks, where each block is a λ-bit string. Thus, $\mathsf{SEH.Extract}$ extracts a λ-bit string as the σ-th position of the pre-image.

Theorem (Informal. See Theorem 1 and Remark 6)**.** *Fix any (arbitrarily small) constant $\delta, \rho \in [0,1]$ and any function $t(\lambda) = \Omega(\lambda^\delta)$. Then, under the LWE assumption, there exists a hash function family that is correlation intractable for a relation ensemble $\boldsymbol{R} = \{\boldsymbol{R}_\lambda \subseteq X_\lambda \times Y_\lambda^{t(\lambda)}\}_{\lambda\in\mathbb{N}}$ if $\boldsymbol{R}$ satisfies the following: for every λ and $x \in X_\lambda$, (i) the set $\boldsymbol{R}_x := \{(y_1,\ldots,y_t) \mid (x,(y_1,\ldots,y_t)) \in \boldsymbol{R}_\lambda\} \subseteq Y_\lambda^{t(\lambda)}$ has a decomposition $\boldsymbol{R}_x = S_1 \times S_2 \times \cdots \times S_{t(\lambda)}$, (ii) each S_i satisfies $|S_i| \le \rho|Y_\lambda|$, and (iii) each S_i is efficiently verifiable, i.e., there exists a polynomial-size circuit C such that $C(x,y,i) = 1$ iff $y \in S_i$. Furthermore, each hash function in the family can be evaluated in time* $\mathsf{poly}(\log|X_\lambda|, |Y_\lambda|, t_\lambda, |C|)$.

In our SNARG construction, we use the following encoding scheme for holographic verification. For a finite field $\mathbb{F}$ and a subset $\mathbb{H} \subseteq \mathbb{F}$, the *low-degree extension (LDE)* $\widehat{x}$ of $x \in \{0,1\}^n$ is defined by first viewing x as a function $x : \mathbb{H}^m \to \{0,1\}$ for $m := \lceil \log_{|\mathbb{H}|} n \rceil$ and next defining $\widehat{x} : \mathbb{F}^m \to \mathbb{F}$ as the unique m-variate polynomial that satisfies $\widehat{x}|_{\mathbb{H}^m} \equiv x$ with individual degree at most $|\mathbb{H}| - 1$. (In this paper, $\mathbb{F}$ and $\mathbb{H}$ are chosen so that $|\mathbb{H}| = O(\log n)$ and $|\mathbb{F}| = \mathsf{poly}(|\mathbb{H}|)$.) What is crucial for us is that LDEs are *linear tensor codes*. In particular, the LDE $\widehat{x}$ of x satisfies the following for any $m' \in \{1,\ldots,m-1\}$.

Property. View $\widehat{x} : \mathbb{F}^m \to \mathbb{F}$ as a $|\mathbb{F}|^{m'} \times |\mathbb{F}|^{m-m'}$ matrix s.t. the i-th row is the truth table of $\widehat{x}(i,\cdot) : \mathbb{F}^{m-m'} \to \mathbb{F}$, where $i \in \{1,\ldots,|\mathbb{F}|^{m'}\}$ is mapped to an element of $\mathbb{F}^{m'}$ by a canonical bijection. Then, each row is a valid LDE of length $\mathbb{F}^{m-m'}$. In particular, the i-th row $\widehat{x}(i,\cdot)$ is the LDE of $\widehat{x}(i,\cdot)|_{\mathbb{H}^{m-m'}}$, where $\widehat{x}(i,\cdot)|_{\mathbb{H}^{m-m'}}$ is the restriction of $\widehat{x}(i,\cdot)$ to the domain $\mathbb{H}^{m-m'}$. Similarly, the j-th column $\widehat{x}(\cdot,j)$ is the LDE of $\widehat{x}(\cdot,j)|_{\mathbb{H}^{m'}}$ for $\forall j \in \{1,\ldots,|\mathbb{F}|^{m-m'}\}$.

Jumping ahead, we use the above property to obtain our SNARG in a recursive way. The prover creates a proof for x by recursively creating a proof for $\widehat{x}(i,\cdot)|_{\mathbb{H}^{m-m'}}$ for certain i. To verify the recursive proof, the verifier makes queries to $\widehat{x}(i,\cdot)$, which is indeed the LDE of $\widehat{x}(i,\cdot)|_{\mathbb{H}^{m-m'}}$ because of the above property. (The above property was used previously in a similar context by [RVW13].)

Now, let us describe our construction. Recall that in holographic SNARGs for SE-hash computations, a statement consists of an input x, an SE hash function key h, and a hash value rt, which are supposed to satisfy $\mathsf{SEH.Hash}(h,x) = \mathsf{rt}$. For simplicity, we view x as a function $x : \mathbb{H}^m \to \mathbb{F}$, where $\mathbb{F}$ and $\mathbb{H}$ are those to be used to compute the LDE for the verifier. Also, we assume for simplicity that the security parameter λ is a power of $|\mathbb{H}|$. Let $m_\lambda := \log_{|\mathbb{H}|} \lambda$, $\delta := 1/(\lfloor m/m_\lambda \rfloor + 1)$, and $\alpha := \lfloor \lambda^\delta \rfloor$. Then, our construction is obtained recursively based on the input length m. When $m < m_\lambda$, the CRS generation and the prover do nothing, and the verifier directly checks $\mathsf{SEH.Hash}(h,x) \stackrel{?}{=} \mathsf{rt}$, where the verifier obtains x by using its oracle access to the LDE of x. When $m \ge m_\lambda$, our construction proceeds as described in Figure 2. At a high level, the prover first views the LDE $\widehat{x}$ as a $|\mathbb{F}|^{m_\lambda} \times |\mathbb{F}|^{m-m_\lambda}$ matrix and obtains an SE hash value for each row (restricted to $\mathbb{H}^{m-m_\lambda}$ as suggested above). Then, the prover selects α rows based on the indices that are obtained by applying a CI hash function to the SE hash values, and recursively creates proofs about these rows. (The prover also creates batch proofs about the correctness of the SE hash values.)

Completeness can be verified by inspection. In particular, the verifier accepts the batch proof π^{idx} and the recursive proofs $\{\pi_{\mathbf{c}_i}\}_{\mathbf{c}_i \in S_{\mathbf{c}}}$ because of the above-described property of LDEs. Also, intuitively the efficiency requirement holds since the number of recursive executions is at most $\alpha^{\lfloor m/m_\lambda \rfloor + 1} \le \lambda$.

To see somewhere soundness, we start with a toy case. For a statement (x,h,rt), consider a prover that behaves honestly for an incorrect statement (x',h,rt') s.t. $x' \neq x$ and $\mathsf{SEH.Hash}(h,x') = \mathsf{rt}'$. Since $x' \neq x$, their LDEs $\widehat{x}, \widehat{x}'$ (viewed as matrices) disagree in a certain column, say, the $\mathbf{v}^*$-th one. Since the $\mathbf{v}^*$-th columns of $\widehat{x}, \widehat{x}'$ are valid LDEs of length $|\mathbb{F}|^{m_\lambda}$ (i.e., they are m_λ-variate polynomials of individual degree at most $|\mathbb{H}| - 1$), it follows that they agree in at most a $\rho := m_\lambda(|\mathbb{H}| - 1)/|\mathbb{F}|$ fraction of the positions. Thus, if we set $|\mathbb{F}|$ large enough, $\widehat{x}$ and $\widehat{x}'$ disagree in a constant fraction of the rows. By temporarily thinking as if the CI hash function h^{CIH} is a truly random function, we conclude that with high probability, one of the recursive proofs is created for a row in which $\widehat{x}$ and $\widehat{x}'$ disagree. Thus, after the recursions, the verifier rejects the proof in the base case.

Let us observe somewhere soundness in more detail. Naturally, we show the somewhere soundness by relying on the somewhere soundness of the recursive executions. Assume for contradiction that a PPT cheating prover P^* breaks the somewhere soundness, i.e., $\exists x : \mathbb{H}^m \to \mathbb{F}$ and $\sigma = (\mathbf{u}^*, \mathbf{v}^*) \in \mathbb{H}^{m_\lambda} \times \mathbb{H}^{m-m_\lambda} = \mathbb{H}^m$ s.t. for a random SE hash function key h that is statistically binding in position $(\mathbf{u}^*, \mathbf{v}^*)$, the cheating prover P^* produces an accepting proof π and a hash value rt s.t. $\mathsf{SEH.Extract}(\mathsf{td}, \mathsf{rt}) \neq x(\mathbf{u}^*, \mathbf{v}^*)$, where td is the

Building blocks: (i) $\mathsf{SEH.Hash}$—an SE hash function. (ii) $\mathsf{BARG}_{\mathrm{idx}}$—a BARG for the index language. (iii) $\mathcal{H} = \{\mathcal{H}_\lambda\}_{\lambda\in\mathbb{N}}$—a hash function family that is correlation intractable for efficiently product verifiable relation ensembles.

CRS generation: Sample an SE hash function key h', a CI hash function $h^{\mathrm{CIH}} \in \mathcal{H}_\lambda$, and a CRS of $\mathsf{BARG}_{\mathrm{idx}}$. Also, recursively sample a CRS of itself for input length $m - m_\lambda$.
Prover P: Given (x, h, rt) and the CRS, compute a proof π as follows.

1. Compute the LDE $\widehat{x} : \mathbb{F}^m \to \mathbb{F}$ of x and view $\widehat{x}$ as a $|\mathbb{F}|^{m_\lambda} \times |\mathbb{F}|^{m-m_\lambda}$ matrix. For each $\mathbf{u} \in \mathbb{F}^{m_\lambda}$, let $\widehat{x}_{(\mathbf{u},*)}$ denote the $\mathbf{u}$-th row of the matrix.
   (a) Compute $\mathsf{rt}_{\mathbf{u}} \coloneqq \mathsf{SEH.Hash}(h', x_{(\mathbf{u},*)})$ for $\forall \mathbf{u} \in \mathbb{F}^{m_\lambda}$, where $x_{(\mathbf{u},*)} \coloneqq \widehat{x}_{(\mathbf{u},*)}|_{\mathbb{H}^{m-m_\lambda}}$.
   (b) For $\forall \mathbf{u} \in \mathbb{H}^{m_\lambda}$, use $\mathsf{BARG}_{\mathrm{idx}}$ to compute a proof $\pi^{\mathrm{idx}}_{\mathbf{u}}$ for the consistency between rt and $\mathsf{rt}_{\mathbf{u}}$ (i.e., prove that $\forall \mathbf{v} \in \mathbb{H}^{m-m_\lambda} \exists x'_{\mathbf{u},\mathbf{v}} \in \mathbb{F}$ s.t. both rt and $\mathsf{rt}_{\mathbf{u}}$ have $x'_{\mathbf{u},\mathbf{v}}$ in the appropriate positions of their pre-images).
   (c) Use $\mathsf{BARG}_{\mathrm{idx}}$ to compute a proof π^{idx} proving that each column that is hashed to $\{\mathsf{rt}_{\mathbf{u}}\}_{\mathbf{u}\in\mathbb{F}^{m_\lambda}}$ is a valid LDE (i.e., prove that $\forall \mathbf{v} \in \mathbb{H}^{m-m_\lambda} \exists x'_{(*,\mathbf{v})} : \mathbb{H}^{m_\lambda} \to \mathbb{F}$ s.t. the values in the $\mathbf{v}$-th position of the pre-images of $\{\mathsf{rt}_{\mathbf{u}}\}_{\mathbf{u}\in\mathbb{F}^{m_\lambda}}$ constitute the LDE of $x'_{(*,\mathbf{v})}$).
2. Apply h^{CIH} to $\{\mathsf{rt}_{\mathbf{u}}\}_{\mathbf{u}\in\mathbb{H}^{m_\lambda}}$ to obtain $S_{\mathbf{c}} \coloneqq (\mathbf{c}_1, \ldots, \mathbf{c}_\alpha) \in \mathbb{F}^{m_\lambda} \times \cdots \times \mathbb{F}^{m_\lambda}$.
3. For $\forall \mathbf{c}_i \in S_{\mathbf{c}}$, recursively invoke the prover P for the statement $(x_{(\mathbf{c}_i,*)}, h', \mathsf{rt}_{\mathbf{c}_i})$ to obtain a proof $\pi_{\mathbf{c}_i}$.
4. Output $\pi \coloneqq (\{\mathsf{rt}_{\mathbf{u}}\}_{\mathbf{u}\in\mathbb{F}^{m_\lambda}}, \{\pi^{\mathrm{idx}}_{\mathbf{u}}\}_{\mathbf{u}\in\mathbb{H}^{m_\lambda}}, \pi^{\mathrm{idx}}, \{\pi_{\mathbf{c}_i}\}_{\mathbf{c}_i\in S_{\mathbf{c}}})$.

Verifier V: Given (h, rt, π), the CRS, and oracle access to the LDE $\widehat{x}$ do the following.

1. Parse π as $(\{\mathsf{rt}_{\mathbf{u}}\}_{\mathbf{u}\in\mathbb{F}^{m_\lambda}}, \{\pi^{\mathrm{idx}}_{\mathbf{u}}\}_{\mathbf{u}\in\mathbb{H}^{m_\lambda}}, \pi^{\mathrm{idx}}, \{\pi_{\mathbf{c}_i}\}_{\mathbf{c}_i\in S_{\mathbf{c}}})$.
2. Apply h^{CIH} to $\{\mathsf{rt}_{\mathbf{u}}\}_{\mathbf{u}\in\mathbb{H}^{m_\lambda}}$ to obtain $S_{\mathbf{c}} \coloneqq (\mathbf{c}_1, \ldots, \mathbf{c}_\alpha) \in \mathbb{F}^{m_\lambda} \times \cdots \times \mathbb{F}^{m_\lambda}$.
3. Output 1 iff all of the proofs $\{\pi^{\mathrm{idx}}_{\mathbf{u}}\}_{\mathbf{u}\in\mathbb{H}^{m_\lambda}}, \pi^{\mathrm{idx}}, \{\pi_{\mathbf{c}_i}\}_{\mathbf{c}_i\in S_{\mathbf{c}}}$ are accepting, where each recursive proof $\pi_{\mathbf{c}_i}$ is verified for the statement $(x_{(\mathbf{c}_i,*)}, h', \mathsf{rt}_{\mathbf{c}_i})$ by recursively invoking the verifier V with oracle $\widehat{x}(\mathbf{c}_i, \cdot)$.

Figure 2: Overview of our holographic SNARG for SE-hashing (when $m \geq m_\lambda$).

trapdoor corresponding to h. Note that, due to the index hiding property of the underlying SE hash function, P^* breaks the somewhere soundness even when the CRS generation algorithm samples the SE hash function key h' in a way that h' is statistically binding in position $\mathbf{v}^*$. Let $\{\widetilde{x}_{(\mathbf{u},\mathbf{v}^*)}\}_{\mathbf{u}\in\mathbb{F}^{m_\lambda}}$ denote the column that we can extract by using $\mathsf{SEH.Extract}$ for the hash values $\{\mathsf{rt}_{\mathbf{u}}\}_{\mathbf{u}\in\mathbb{F}^{m_\lambda}}$ that P^* provides as the hashes of the rows. To use P^* to break the soundness of a recursive proof, we show that $\exists i \in [\alpha]$ s.t. the i-th recursive statement $(x_{(\mathbf{c}_i,*)}, h', \mathsf{rt}_{\mathbf{c}_i})$ is false, i.e., the $\mathbf{c}_i$-th row of the extracted column $\{\widetilde{x}_{(\mathbf{u},\mathbf{v}^*)}\}_{\mathbf{u}\in\mathbb{F}^{m_\lambda}}$ disagrees with the correct value $\widehat{x}(\mathbf{c}_i, \mathbf{v}^*)$. Put differently, we show that the output $(\mathbf{c}_1, \ldots, \mathbf{c}_\alpha)$ of the CI hash function is not included in the set $S_1 \times \cdots \times S_\alpha$ s.t. $S_i \coloneqq \{\mathbf{c} \in \mathbb{F}^{m_\lambda} \mid \widetilde{x}_{(\mathbf{c},\mathbf{v}^*)} = \widehat{x}(\mathbf{c}, \mathbf{v}^*)\}$. Toward this goal, we can use the correlation intractability of h^{CIH} straightforwardly since we have $|S_i| \leq \rho|\mathbb{F}|^{m_\lambda}$ for a constant ρ as shown below.[6]

> Let $\widetilde{x}_{(*,\mathbf{v}^*)} : \mathbb{F}^{m_\lambda} \to \mathbb{F}$ be the function representing the extracted column, i.e., $\widetilde{x}_{(*,\mathbf{v}^*)} : \mathbf{u} \mapsto \widetilde{x}_{(\mathbf{u},\mathbf{v}^*)}$. The somewhere soundness of $\mathsf{BARG}_{\mathrm{idx}}$ guarantees that $\widetilde{x}_{(*,\mathbf{v}^*)}$ is a valid LDE s.t. $\widetilde{x}_{(*,\mathbf{v}^*)}(\mathbf{u}^*) = \mathsf{SEH.Extract}(\mathsf{td}, \mathsf{rt})$. Since we assumed $\mathsf{SEH.Extract}(\mathsf{td}, \mathsf{rt}) \neq x(\mathbf{u}^*, \mathbf{v}^*)$ for contradiction, it follows that $\widetilde{x}_{(*,\mathbf{v}^*)}$ and $\widehat{x}(\cdot, \mathbf{v}^*)$ are valid LDEs that disagree in position $\mathbf{u}^*$. Thus, by appropriately setting the parameter of the LDE, we can guarantee that $\widetilde{x}_{(*,\mathbf{v}^*)}$ and $\widehat{x}(\cdot, \mathbf{v}^*)$ disagree in a constant fraction of positions.

Thus, the correlation intractability of h^{CIH} guarantees that $\exists i \in [\alpha]$ s.t. $\widetilde{x}_{(\mathbf{c}_i,\mathbf{v}^*)} \neq \widehat{x}(\mathbf{c}_i, \mathbf{v}^*)$, i.e., the recursive statement $(x_{(\mathbf{c}_i,*)}, h', \mathsf{rt}_{\mathbf{c}_i})$ is false. Thus, we can break the soundness of a recursive execution and reach a contradiction.

[6] S_i is efficiently verifiable by a circuit that has $\{\mathsf{td}_{\mathbf{u}}\}_{\mathbf{u}\in\mathbb{F}^{m_\lambda}}$ and $\{\widehat{x}(\mathbf{u}, \mathbf{v}^*)\}_{\mathbf{u}\in\mathbb{F}^{m_\lambda}}$ as hardwired inputs.

2.2 Holographic SNARG for Batch-NP

Our starting point is the non-holographic SNARG for Batch-NP by Choudhuri, Jain, and Jin (CJJ) [CJJ22], which is non-holographic only in that the verifier computes an SE hash of the instances. Concretely, the CJJ SNARG works as follows. For k instances $x_1, \ldots, x_k$ of an NP language $\mathbf{L}$, the prover computes a hash rt of the instances using an SE hash function and then uses a BARG for the index language to prove the following for each $i \in [k]$: (i) the i-th position of the pre-image of the hash rt can be opened to an instance $\tilde{x}_i$ and (ii) $\tilde{x}_i \in \mathbf{L}$. The verifier computes the SE hash of the instances and verifies the BARG proof.

We make the CJJ SNARG holographic by letting the prover send the hash of the instances along with a holographic proof about the correctness of the hash. For the analysis of the CJJ SNARG to go through, the proof about the SE hash needs to have the following weak form of soundness: if the i-th instance is extracted from the hash, the extracted instance is equal to x_i. This form of soundness is precisely what our somewhere-sound holographic SNARG for SE-hashing guarantees. Thus, combining it with the CJJ SNARG suffices.

3 Preliminaries

3.1 Notations and Conventions

We use λ to denote the security parameter. For any binary strings $a, b \in \{0,1\}^*$, we use $a \,\|\, b$ to denote their concatenation. For any $n \in \mathbb{N}$, we use $[n]$ to denote the set $\{1, \ldots, n\}$. We use poly to denote an unspecified polynomial, negl to denote an unspecified negligible function, and PPT as an abbreviation of "probabilistic polynomial-time." For any NP language $\mathbf{L}$ and an instance $x \in \mathbf{L}$, we use $\mathbf{R}_{\mathbf{L}}$ to denote the witness relation (i.e., $\mathbf{R}_{\mathbf{L}}$ is the set of all instance-witness pairs of $\mathbf{L}$) and use $\mathbf{R}_{\mathbf{L}}(x)$ to denote the set of all witnesses of x. For a vector $\mathbf{v} = (v_1, \ldots, v_N)$ and a set $S \subseteq [N]$, let $\mathbf{v}|_S := \{v_i\}_{i \in S}$. Similarly, for a function $f : D \to R$ and a set $S \subseteq D$, let $f|_S$ be the function that is obtained by restricting the domain of f to S. We often view a function as a string that represents its truth table; e.g., we sometimes view $f|_S$ as $\{f(i)\}_{i \in S}$. For a set S, we denote by $s \leftarrow S$ the process of obtaining an element $s \in S$ by a uniform sampling from S. For any probabilistic algorithm Algo and an input x, we denote by $y \leftarrow \mathsf{Algo}(x)$ the process of obtaining an output y by running $\mathsf{Algo}(x)$ with uniform randomness. Unless otherwise stated, each algorithm runs in time polynomial in the length of its first input. (The length of the first input is usually equal to or polynomially bounded by the security parameter.)

3.2 Low-Degree Extensions

For any finite field $\mathbb{F}$, a subset $\mathbb{H} \subseteq \mathbb{F}$, and an integer $m \in \mathbb{N}$, the *low-degree extension (LDE)* of a function $f : \mathbb{H}^m \to \{0,1\}$ (or $f : \mathbb{H}^m \to \mathbb{F}$) is the unique function $\widehat{f} : \mathbb{F}^m \to \mathbb{F}$ such that (i) $\widehat{f}(\mathbf{z}) = f(\mathbf{z})$ for every $\mathbf{z} \in \mathbb{H}^m$ and (ii) $\widehat{f}$ is an m-variate polynomial of degree at most $|\mathbb{H}| - 1$ in each variable.[7] Concretely speaking, $\widehat{f}$ can be written as follows.

$$\widehat{f}(z_1, \ldots, z_m) = \sum_{(u_1, \ldots, u_m) \in \mathbb{H}^m} \prod_{i \in [m]} \prod_{v \in \mathbb{H} \setminus \{u_i\}} \frac{z_i - v}{u_i - v} \cdot f(u_1, \ldots, u_m) \ . \tag{1}$$

For any $n \in \mathbb{N}$, the LDE of a binary string $x = (x_1, \ldots, x_n) \in \{0,1\}^n$ (or a vector $x \in \mathbb{F}^n$) can be obtained by viewing x as a function as follows: choose $\mathbb{H}$ and m such that $n \leq |\mathbb{H}|^m$ in a canonical way; identify $\{1, \ldots, |\mathbb{H}|^m\}$ with $\mathbb{H}^m$ by the lexicographical order; and view x as a function $x : \mathbb{H}^m \to \{0,1\}$ such that $x(i) = x_i$ for $\forall i \in [n]$ and $x(i) = 0$ for $\forall i \in \{n+1, \ldots, |\mathbb{H}|^m\}$.

We use $\mathsf{LDE}_{\mathbb{F},\mathbb{H},m}(x)$ to denote the LDE of x w.r.t. $(\mathbb{F}, \mathbb{H}, m)$. For any appropriate $(\mathbb{F}, \mathbb{H}, m)$ and $\mathbf{z} \in \mathbb{F}^m$, the LDE of x can be evaluated on $\mathbf{z}$ in time $|\mathbb{H}|^m \cdot \mathsf{poly}(m, |\mathbb{H}|)$, where it is assumed that we have $|\mathbb{F}| = \mathsf{poly}(|\mathbb{H}|)$ and each field operation over $\mathbb{F}$ can be done in time $\mathsf{poly}(\log|\mathbb{F}|)$ (see, e.g., [GKR15, Claim 2.3]).

Below is a technical lemma that we use in Section 4.

Lemma 1. *For a finite field $\mathbb{F}$, a subset $\mathbb{H} \subseteq \mathbb{F}$, an integer $m \in \mathbb{N}$, and a function $f : \mathbb{H}^m \to \{0,1\}$, let $\widehat{f} : \mathbb{F}^m \to \mathbb{F}$ denote the LDE of f.*

[7] In this paper, the LDE of f is also considered for the case of $m = 0$. When $m = 0$, the function f is a single element b (which is viewed as a function with domain $\{0\}$), and the LDE of f is b itself.

1. *For any* $m' \in [m-1]$ *and* $\mathbf{c} = (c_1, \ldots, c_{m'}) \in \mathbb{F}^{m'}$*, let* $\widehat{f}_{(\mathbf{c},*)} : \mathbb{F}^{m-m'} \to \mathbb{F}$ *be defined as* $\widehat{f}_{(\mathbf{c},*)} : (z_{m'+1}, \ldots, z_m) \mapsto \widehat{f}(c_1, \ldots, c_{m'}, z_{m'+1}, \ldots, z_m)$*, and let* $f_{(\mathbf{c},*)} \equiv \widehat{f}_{(\mathbf{c},*)}|_{\mathbb{H}^{m-m'}}$*. Then,* $\widehat{f}_{(\mathbf{c},*)}$ *is the LDE of* $f_{(\mathbf{c},*)}$ *w.r.t.* $(\mathbb{F}, \mathbb{H}, m-m')$.

2. *For any* $m' \in [m-1]$ *and* $\mathbf{c} = (c_{m-m'+1}, \ldots, c_m) \in \mathbb{F}^{m'}$*, let* $\widehat{f}_{(*,\mathbf{c})} : \mathbb{F}^{m-m'} \to \mathbb{F}$ *be defined as* $\widehat{f}_{(*,\mathbf{c})} : (z_1, \ldots, z_{m-m'}) \mapsto \widehat{f}(z_1, \ldots, z_{m-m'}, c_{m-m'+1}, \ldots, c_m)$*, and let* $f_{(*,\mathbf{c})} \equiv \widehat{f}_{(*,\mathbf{c})}|_{\mathbb{H}^{m-m'}}$*. Then,* $\widehat{f}_{(*,\mathbf{c})}$ *is the LDE of* $f_{(*,\mathbf{c})}$ *w.r.t.* $(\mathbb{F}, \mathbb{H}, m-m')$.

Proof. This lemma follows from the uniqueness of LDEs since $\widehat{f}_{(\mathbf{c},*)}$ (resp., $\widehat{f}_{(*,\mathbf{c})}$) agrees with $f_{(\mathbf{c},*)}$ (resp., $f_{(*,\mathbf{c})}$) in $\mathbb{H}^{m-m'}$ and has individual degree at most $\mathbb{H}-1$. □

3.3 Hash Functions

Recall that a *hash function family* can be modeled by two algorithms $(\mathsf{Gen}, \mathsf{Hash})$ as follows for a domain-codomain ensemble $\{(X_\lambda, Y_\lambda)\}_{\lambda\in\mathbb{N}}$.

- $h \leftarrow \mathsf{Gen}(1^\lambda)$: Gen is a PPT algorithm that takes as input a security parameter λ (in unary), and it outputs a key h.
- $y := \mathsf{Hash}(h, x)$: Hash is a deterministic polynomial-time algorithm that takes as input a key $h \in \mathsf{Gen}(1^\lambda)$ and an input value $x \in X_\lambda$, and it outputs a hash value $y \in Y_\lambda$.

A hash function family is called *public-coin* [HR04] if $\mathsf{Gen}(1^\lambda)$ outputs a uniformly random string.

In this paper, we use the following simplified notations. For a hash function family $\mathcal{H} = (\mathsf{Gen}, \mathsf{Hash})$, we use $\mathcal{H}_\lambda$ to denote the range of $\mathsf{Gen}(1^\lambda)$, use $h \leftarrow \mathcal{H}_\lambda$ as a shorthand of $h \leftarrow \mathsf{Gen}(1^\lambda)$, and use $h(x)$ as a shorthand of $\mathsf{Hash}(h, x)$.

Tree hash. Recall that for any function $h : \{0,1\}^{2\lambda} \to \{0,1\}^\lambda$ and any $\ell \in \mathbb{N}$, the (binary) *tree-hash* of a string $x \in \{0,1\}^{2^\ell \lambda}$ is obtained as follows.

1. Partition x into 2^ℓ blocks $\mathsf{blk}_0, \ldots, \mathsf{blk}_{2^\ell-1}$ such that $|\mathsf{blk}_0| = \cdots = |\mathsf{blk}_{2^\ell-1}| = \lambda$.
2. Define $\mathsf{node}_{i,\sigma} \in \{0,1\}^\lambda$ for $\forall i \in \{0, \ldots, \ell\}, \sigma \in \{0, \ldots, 2^i - 1\}$ as follows.
   (a) $\mathsf{node}_{\ell,\sigma} := \mathsf{blk}_\sigma$ for $\forall \sigma \in \{0, \ldots, 2^\ell - 1\}$.
   (b) $\mathsf{node}_{i,\sigma} := h(\mathsf{node}_{i+1,2\sigma} \,\|\, \mathsf{node}_{i+1,2\sigma+1})$ for $\forall i \in \{\ell-1, \ldots, 0\}, \sigma \in \{0, \ldots, 2^i - 1\}$
3. Let the tree-hash of x be $\mathsf{node}_{0,0}$.

We use $\mathsf{TreeHash}_h(x)$ to denote the tree-hash of x.

3.4 Correlation-Intractable Hash Functions

We recall the definition of *correlation-intractable hash functions* [CGH04].

Definition 1 (Correlation intractability)**.** *Let* $\mathcal{H} = (\mathsf{Gen}, \mathsf{Hash})$ *be a hash function family and* $\{(X_\lambda, Y_\lambda)\}_{\lambda\in\mathbb{N}}$ *be its domain-codomain ensemble. Then,* $\mathcal{H}$ *is* ***correlation intractable*** *for a relation ensemble* $\boldsymbol{R} = \{\boldsymbol{R}_\lambda \subseteq X_\lambda \times Y_\lambda\}_{\lambda\in\mathbb{N}}$ *if for every* PPT *algorithm* $\mathcal{A}$*, there exists a negligible function* negl *such that for every* $\lambda \in \mathbb{N}$ *and* $z \in \{0,1\}^*$*,*

$$\Pr\left[(x,y) \in \boldsymbol{R}_\lambda \;\middle|\; \begin{array}{l} h \leftarrow \mathsf{Gen}(1^\lambda) \\ x \leftarrow \mathcal{A}(h, z) \\ y := \mathsf{Hash}(h, x) \end{array}\right] \leq \mathsf{negl}(\lambda) \ .$$

Correlation intractability for efficiently verifiable product relations. We recall a result by Holmgren, Lombardi, and Rothblum [HLR21b] about correlation intractability for *efficiently verifiable product relations*.

Definition 2 (Product relation [HLR21a, Definition 3.1])**.** *A relation $\boldsymbol{R} \subseteq X \times Y^t$ is said to be a **product relation** if for every $x \in X$, the set $\boldsymbol{R}_x := \{(y_1, \ldots, y_t) \mid (x, (y_1, \ldots, y_t)) \in \boldsymbol{R}\} \subseteq Y^t$ has a decomposition $\boldsymbol{R}_x = S_1 \times S_2 \times \cdots \times S_t$, where $S_1, \ldots, S_t \subseteq Y$ may depend on x.*

Definition 3 (Efficient product verifiability [HLR21a, Definition 3.3], slightly modified)**.** *For any $T \in \mathbb{N}$, a product relation $\boldsymbol{R} \subseteq X \times Y^t$ is called T-**time product verifiable** if there exists a size-T circuit C such that for every input $x \in X$ with the corresponding sets $S_1, \ldots, S_t$ as in Definition 2, it holds that $C(x, y, i) = 1$ iff $y \in S_i$.*[8]

Definition 4 (Product sparsity [HLR21a, Definition 3.4])**.** *For any $\rho \in [0, 1]$, a product relation $\boldsymbol{R} \subseteq X \times Y^t$ is said to have **product sparsity** ρ if for every input $x \in X$, the sets $S_1, \ldots, S_t$ as in Definition 2 have size at most $\rho|Y|$.*

Theorem 1 ([HLR21a, Theorem 5.1])**.** *Let $T : \mathbb{N} \to \mathbb{N}$ and $\rho : \mathbb{N} \to [0, 1]$ be functions and $\boldsymbol{R} = \{\boldsymbol{R}_\lambda \subseteq X_\lambda \times Y_\lambda^{t_\lambda}\}_{\lambda \in \mathbb{N}}$ be an ensemble of product relations such that (i) each $\boldsymbol{R}_\lambda$ is $T(\lambda)$-time product verifiable with product sparsity $\rho(\lambda)$, (ii) $|Y_\lambda|$, $\log|X_\lambda|$, $T(\lambda)$, and t_λ are all upper bounded by $\lambda^{O(1)}$, and (iii) $t_\lambda \geq \lambda / \log(1/\rho(\lambda))$. Let $X := \{X_\lambda\}_{\lambda \in \mathbb{N}}$, $Y := \{Y_\lambda\}_{\lambda \in \mathbb{N}}$ and $t := \{t_\lambda\}_{\lambda \in \mathbb{N}}$. Then, for the domain-codomain ensemble $\{(X_\lambda, Y_\lambda^{t_\lambda})\}_{\lambda \in \mathbb{N}}$, there exists a hash function family $\mathcal{H}$ that is correlation intractable for $\boldsymbol{R}$ under the LWE assumption. Moreover, $\mathcal{H}$ depends only on (X, Y, ρ, t, T) (and is otherwise independent of $\boldsymbol{R}$) and can be evaluated in time* $\mathsf{poly}(\log|X_\lambda|, |Y_\lambda|, t_\lambda, T(\lambda))$.

Several remarks about Theorem 1 are given below. First, the correlation-intractable hash function family $\mathcal{H}$ of Theorem 1 can be made public-coin. (This is because one of its main components, the LWE-based correlation-intractable hash family of Peikert and Shiehian [PS19], can be made public-coin.) Also, $\mathcal{H}$ can be efficiently determined given (X, Y, ρ, t, T), and its correlation intractability holds for a negligible function that depends only on (X, Y, ρ, t, T) (and is otherwise independent of $\mathbf{R}$). Additionally, as mentioned in [HLR21a, Section 5.1], the condition $t_\lambda \geq \lambda / \log(1/\rho(\lambda))$ in Theorem 1 can be weakened to $t_\lambda \geq \lambda^\delta / \log(1/\rho(\lambda))$ for an arbitrarily small constant $\delta > 0$. Similarly, if the LWE assumption holds against slightly super-polynomial-time adversaries, the condition can be weakened to $t_\lambda \geq \lambda^{1/\tau(\lambda)} / \log(1/\rho(\lambda))$ for a super-constant function $\tau(\lambda) = \omega(1)$. (See Section A.1 in the appendix for details.)

3.5 Somewhere Extractable Hash Functions

We recall the definition of *somewhere extractable hash functions* [HW15, CJJ21a, CJJ22]. The following definition is adapted from [CJJ21b, Section 3.5]; the differences from the original definition are summarized in Remark 1 below.

Definition 5 (Somewhere extractable hash with local opening)**.** *A **somewhere extractable hash function family** consists of a tuple of algorithms $(\mathsf{Gen}, \mathsf{TGen}, \mathsf{Hash}, \mathsf{Open}, \mathsf{Verify}, \mathsf{Extract})$ satisfying the following.*

- ***Syntax.*** Gen *and* TGen *are probabilistic and the others are deterministic.*
- ***Opening correctness.*** *For every $\lambda \in \mathbb{N}$, $N \in [2^\lambda]$, $M_I \in [N]$, $i \in [N]$, and $m = (m_1, \ldots, m_N) \in \{0, 1\}^N$,*

$$\Pr\left[\mathsf{Verify}(h, \mathsf{rt}, m_i, i, \mathsf{cert}_i) = 1 \;\middle|\; \begin{array}{l} h \leftarrow \mathsf{Gen}(1^\lambda, N, 1^{M_I}) \\ \mathsf{rt} := \mathsf{Hash}(h, m) \\ \mathsf{cert}_i := \mathsf{Open}(h, m, i) \end{array}\right] = 1 \ .$$

- ***Key indistinguishability.*** *For every* PPT *algorithms $(\mathcal{A}_1, \mathcal{A}_2)$ and polynomial poly_N, there exists a negligible function* negl *such that for every $\lambda \in \mathbb{N}$, $N \leq \mathsf{poly}_N(\lambda)$, and $z \in \{0, 1\}^*$,*

$$\left| \Pr\left[\mathcal{A}_2(\mathsf{st}, h) = 1 \;\middle|\; \begin{array}{l} (\mathsf{st}, I) \leftarrow \mathcal{A}_1(1^\lambda, N, z) \\ h \leftarrow \mathsf{Gen}(1^\lambda, N, 1^{|I|}) \end{array}\right] - \Pr\left[\mathcal{A}_2(\mathsf{st}, h) = 1 \;\middle|\; \begin{array}{l} (\mathsf{st}, I) \leftarrow \mathcal{A}_1(1^\lambda, N, z) \\ (h, \mathsf{td}) \leftarrow \mathsf{TGen}(1^\lambda, N, I) \end{array}\right] \right| \leq \mathsf{negl}(\lambda) \ .$$

[8] The definition is slightly modified in that the size of C is required to be bounded by T rather than an arbitrary polynomial.

- ***Somewhere (statistical) extractability.*** *For any polynomial* poly_I*, there exists a negligible function* negl *such that for every* $\lambda \in \mathbb{N}$*,* $N \in [2^\lambda]$*, and* $I \subseteq [N]$ *such that* $|I| \leq \mathsf{poly}_I(\lambda)$*,*

$$\Pr\left[\begin{array}{l|l} \exists\, \mathsf{rt}, m_{i^*}, i^*, \mathsf{cert}_{i^*} \text{ s.t.} & \\ i^* \in I & (h, \mathsf{td}) \leftarrow \mathsf{TGen}(1^\lambda, N, I) \\ \wedge\, \mathsf{Verify}(h, \mathsf{rt}, m_{i^*}, i^*, \mathsf{cert}_{i^*}) = 1 & \\ \wedge\, \widetilde{m}_{i^*} \neq m_{i^*}, \text{ where } \{\widetilde{m}_i\}_{i\in I} := \mathsf{Extract}(\mathsf{td}, \mathsf{rt}) & \end{array}\right] \leq \mathsf{negl}(\lambda) \ .$$

- ***Efficiency.*** *In the above opening correctness experiment, the following hold.*
  - *The key generation* Gen *runs in time* $\mathsf{poly}(\lambda, \log N, M_I)$.
  - *The hashing algorithm* Hash *runs in time* $\mathsf{poly}(\lambda, N)$ *and outputs a hash value* rt *of length* $\mathsf{poly}(\lambda, \log N, M_I)$.
  - *The opening algorithm* Open *runs in time* $\mathsf{poly}(\lambda, N)$ *and outputs a certificate* cert_i *of length* $\mathsf{poly}(\lambda, \log N, M_I)$.
  - *The verifier* Verify *runs in time* $\mathsf{poly}(\lambda, \log N, M_I)$.

  Also, in the above extractability experiment, the following hold.
  - *The trapdoor key generation* TGen *runs in time* $\mathsf{poly}(\lambda, \log N, M_I)$.
  - *The extractor* $\mathsf{Extract}$ *runs in time* $\mathsf{poly}(\lambda, \log N, M_I)$.

Remark 1 (Differences from the definition in [CJJ21b, Section 3.5].). First, following recent works (e.g., [JJ22]), we use the name "somewhere extractable hash functions" rather than "somewhere extractable commitments" to emphasize that no hiding property is guaranteed. Second, the syntax of Gen and TGen is slightly modified. In particular, the message length N is given in binary rather than in unary so that N can be super-polynomial in λ. (Recall that Gen runs in time polylogarithmic in N.) Third, the somewhere extractability is slightly weaker. In particular, we only require somewhere "statistical" extractability rather than somewhere "perfect" extractability. Finally, the efficiency condition for the running time of $\mathsf{Extract}$ is given explicitly. ♢

As observed in [CJJ22], the LWE-based (public-coin) hash functions family by Hubacek and Wichs [HW15] is somewhere extractable.

Theorem 2. *Under the LWE assumption, there exists a (public-coin) somewhere extractable hash function family.*

Somewhere extractable hash for non-binary alphabets. In this paper, somewhere extractable hash functions are also used for strings over a finite field $\mathbb{F}$, where Definition 5 is straightforwardly generalized for non-binary alphabets. Concretely, $\mathbf{v} = (v_1, \ldots, v_N) \in \mathbb{F}^N$ is hashed by hashing $(v_{1,j}, \ldots, v_{N,j})$ for every $j \in [\log|\mathbb{F}|]$, where $v_{i,j}$ is the j-th bit of v_i. The running time of Gen remains the same since the same key can be used for all the hashes, while the complexities related to the other algorithms increase by a multiplicative factor $\log|\mathbb{F}|$.

3.6 Keyless Multi-Collision Resistant Hash Functions

We recall the definition of multi-collision resistant hash functions from [BKP18], focusing on the keyless version.

Definition 6. *For any functions* $K : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ *and* $\gamma : \mathbb{N} \to \mathbb{N}$*, a keyless hash function* Hash *is called* ***weakly*** (K, γ)***-collision-resistant*** *if for every probabilistic* $\gamma^{O(1)}$*-time adversary* $\mathcal{A}$ *and every sequence of polynomial-size advice* $\{z_\lambda\}_{\lambda\in\mathbb{N}}$*, there exists a negligible function* negl *such that for every* $\lambda \in \mathbb{N}$*, the following holds for* $K = K(\lambda, |z_\lambda|)$*.*

$$\Pr\left[\begin{array}{l|l} y_1 = \cdots = y_K & (x_1, \ldots, x_K) \leftarrow \mathcal{A}(1^\lambda, z_\lambda) \\ \wedge\, \forall i \neq j : x_i \neq x_j & \forall i : y_i := \mathsf{Hash}(1^\lambda, x_i) \end{array}\right] \leq \mathsf{negl}(\gamma(\lambda)).$$

As in [BKP18, Kiy22a], we focus on the case that Hash is polynomially compressing. Concretely, we assume that $\mathsf{Hash}(1^\lambda, \cdot)$ takes a string of length λ^2 as input and outputs a string of length λ.

3.7 SNARGs for P (a.k.a. Non-Interactive Turing-Machine Delegations)

We recall the definition of SNARGs for P, a.k.a. *publicly verifiable non-interactive Turing-machine delegation schemes*. The following definitions are adapted from those given in [KPY19a, Section 3.1]; the differences from the original definitions are summarized in Remark 2 below.

Definition 7 (Language $\mathbf{L}_M$)**.** *For a (deterministic) Turing machine M, the language $\boldsymbol{L}_M$ is defined as follows.*

$$\boldsymbol{L}_M \coloneqq \{(\chi, T) \mid M \text{ accepts } \chi \text{ within } T \text{ steps}\} \ .$$

Definition 8 (Partially adaptive Turing-machine delegation)**.** *For a (deterministic) two-input Turing machine M and pair of functions $T_{\mathsf{Gen}} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, $L_\pi : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, a triple of algorithms $(\mathsf{Gen}, \mathsf{P}, \mathsf{V})$ is called a **partially adaptive publicly verifiable non-interactive delegation scheme for** M **with setup time** T_{Gen} **and proof length** L_π if it satisfies the following.*

- ***Syntax.*** Gen *is probabilistic and the others are deterministic.*
- ***Completeness.*** *For every $\lambda, T, n_1, n_2 \in \mathbb{N}$ and $\chi = (\chi_1, \chi_2) \in \{0,1\}^{n_1} \times \{0,1\}^{n_2}$ such that $n \coloneqq n_1 + n_2 \le T \le 2^\lambda$ and $(\chi, T) \in \boldsymbol{L}_M$,*

$$\Pr\left[\mathsf{V}(\mathsf{crs}, \chi, \pi) = 1 \;\middle|\; \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda, T, n_1, n_2) \\ \pi \coloneqq \mathsf{P}(\mathsf{crs}, \chi) \end{array}\right] = 1 \ .$$

- ***Partially adaptive soundness.*** *For every PPT algorithm P^* and triple of polynomials poly_T, poly_{n_1}, poly_{n_2}, there exists a negligible function negl such that for every $\lambda \in \mathbb{N}$, $T \le \mathsf{poly}_T(\lambda)$, $n_1 \le \mathsf{poly}_{n_1}(\lambda)$, $n_2 \le \mathsf{poly}_{n_2}(\lambda)$, $\chi_1 \in \{0,1\}^{n_1}$, and $z \in \{0,1\}^*$,*

$$\Pr\left[\mathsf{V}(\mathsf{crs}, \chi, \pi) = 1 \wedge (\chi, T) \notin \boldsymbol{L}_M \;\middle|\; \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda, T, n_1, n_2) \\ (\chi_2, \pi) \leftarrow \mathsf{P}^*(\mathsf{crs}, \chi_1, z) \\ \chi \coloneqq (\chi_1, \chi_2) \end{array}\right] \le \mathsf{negl}(\lambda) \ .$$

- ***Efficiency.*** *In the above completeness experiment, the following hold.*
  - *The setup algorithm Gen runs in time $T_{\mathsf{Gen}}(\lambda, T, n_1, n_2)$.*
  - *The prover P runs in time $\mathsf{poly}(\lambda, T)$ and outputs a proof π of length $L_\pi(\lambda, T, n_1, n_2)$.*
  - *The verifier V runs in time $O(L_\pi) + \mathsf{poly}(\lambda, n)$.*[9]

*A publicly verifiable non-interactive delegation scheme is called **public-coin** if the setup algorithm Gen is public-coin, i.e., it just outputs a string that is sampled uniformly randomly (possibly along with various parameters that are determined deterministically based on the input of Gen).*

Remark 2 (Differences from the original definition [KPY19b, Section 3.1].). Most importantly, we consider schemes for two-input Turing machines and define soundness in the *partially adaptive setting* [BR22], i.e., the setting where the first input is non-adaptively determined while the second one is adaptively chosen by the prover. Other differences are the following. First, since M is a two-input Turing machine, Gen takes the length of both inputs (n_1 and n_2) rather than the total input length n. Second, for editorial simplicity, the output of Gen is denoted as a single CRS and not parsed as a pair of public keys (a prover key pk and a verifier key vk). Third, to describe the setup time T_{Gen} and the proof length L_π of our schemes more precisely, we allow them to additionally depend on the input lengths n_1, n_2. Finally, we define the public-coin property naturally. $\lozenge$

Definition 9 (Super-polynomial security)**.** *For a super-polynomial function γ, a partially adaptive publicly verifiable non-interactive delegation scheme is called **γ-secure** if the partial adaptive soundness holds even when (i) the adversary P^* runs in time $\mathsf{poly}(\gamma(\lambda))$ and (ii) the polynomials poly_T, poly_{n_1}, poly_{n_2} are all replaced with γ.*

[9] For convenience, we use a slightly weaker bound than prior works [KPY19b], where the bound is $O(L_\pi) + n \cdot \mathsf{poly}(\lambda)$. As remarked in [KPY19a, Remark 3.3], any bound less than the running time of M is non-trivial.

3.8 SNARGs for Batch-NP (a.k.a. Non-Interactive BARGs)

We recall the definition of SNARGs for Batch-NP, a.k.a. *publicly verifiable non-interactive batch arguments (BARGs)*. The following definitions are adapted from those given in [CJJ21b, Section 4.1]; the differences from the original definitions are summarized in Remark 3.

Definition 10 (Circuit satisfiability)**.** *Let $\boldsymbol{R}_{\mathbf{CSAT}}$ be the following relation.*

$$\boldsymbol{R}_{\mathbf{CSAT}} \coloneqq \{((C,x),w) \mid C \text{ is a Boolean circuit s.t. } C(x,w)=1\} \ .$$

Then, $\mathbf{CSAT}$ *is the language defined as follows.*

$$\mathbf{CSAT} \coloneqq \{(C,x) \mid \exists w \in \{0,1\}^* \text{ s.t. } ((C,x),w) \in \boldsymbol{R}_{\mathbf{CSAT}}\} \ .$$

Definition 11 (Batch circuit satisfiability)**.** *For any $k \in \mathbb{N}$, let $\boldsymbol{R}^{\otimes k}_{\mathbf{CSAT}}$ be the following relation.*

$$\boldsymbol{R}^{\otimes k}_{\mathbf{CSAT}} \coloneqq \{((C,\mathbf{x}),\mathbf{w}) \mid \mathbf{x}=(x_1,\ldots,x_k) \text{ and } \mathbf{w}=(w_1,\ldots,w_k) \text{ satisfy } ((C,x_i),w_i) \in \boldsymbol{R}_{\mathbf{CSAT}} \text{ for } \forall i \in [k]\} \ .$$

Then, for any $k \in \mathbb{N}$, $\mathbf{CSAT}^{\otimes k}$ *is the language defined as follows.*

$$\mathbf{CSAT}^{\otimes k} \coloneqq \{(C,\mathbf{x}) \mid \exists \mathbf{w} \text{ s.t. } ((C,\mathbf{x}),\mathbf{w}) \in \boldsymbol{R}^{\otimes k}_{\mathbf{CSAT}}\} \ .$$

Definition 12 ((Non-adaptive) BARG for $\mathbf{CSAT}$)**.** *A triple of algorithms $(\mathsf{Gen},\mathsf{P},\mathsf{V})$ is called a* ***(non-adaptive) publicly verifiable non-interactive batch argument for*** $\mathbf{CSAT}$ *if it satisfies the following.*

- ***Syntax.*** Gen *is probabilistic and the others are deterministic.*
- ***Completeness.*** *For every $\lambda,k,n \in \mathbb{N}$ and $((C,\mathbf{x}),\mathbf{w}) \in \boldsymbol{R}^{\otimes k}_{\mathbf{CSAT}}$ such that $\mathbf{x} \in (\{0,1\}^n)^k$,*

$$\Pr\left[\mathsf{V}(\mathsf{crs},C,\mathbf{x},\pi)=1 \;\middle|\; \begin{array}{l}\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda,1^n,1^{|C|},k)\\ \pi \coloneqq \mathsf{P}(\mathsf{crs},C,\mathbf{x},\mathbf{w})\end{array}\right]=1 \ .$$

- ***(Non-adaptive) Soundness.*** *For every PPT algorithm P^* and pair of polynomials poly_k, poly_C, there exists a negligible function negl such that for every $\lambda \in \mathbb{N}$, $k \le \mathsf{poly}_k(\lambda)$, $M_C \le \mathsf{poly}_C(\lambda)$, $n \le M_C$, $(C,\mathbf{x}) \notin \mathbf{CSAT}^{\otimes k}$, and $z \in \{0,1\}^*$ such that $|C| \le M_C$ and $\mathbf{x} \in (\{0,1\}^n)^k$,*

$$\Pr\left[\mathsf{V}(\mathsf{crs},C,\mathbf{x},\pi)=1 \;\middle|\; \begin{array}{l}\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda,1^n,1^{M_C},k)\\ \pi \leftarrow \mathsf{P}^*(\mathsf{crs},C,\mathbf{x},z)\end{array}\right] \le \mathsf{negl}(\lambda) \ .$$

- ***Efficiency.*** *In the above completeness experiment, the following hold.*
  - *The setup algorithm Gen runs in time $\mathsf{poly}(\lambda,|C|,\log k)$.*
  - *The prover P runs in time $\mathsf{poly}(\lambda,|C|,k)$ and outputs a proof π of length $\mathsf{poly}(\lambda,|C|,\log k)$.*
  - *The verifier V runs in time $\mathsf{poly}(\lambda,n,k)+\mathsf{poly}(\lambda,M_C,\log k)$.*

A publicly verifiable non-interactive batch argument is called ***public-coin*** *if the setup algorithm Gen is public-coin, i.e., it just outputs a string that is sampled uniformly randomly (possibly along with various parameters that are determined deterministically based on the input of Gen).*

Definition 13 (Semi-adaptive somewhere soundness)**.** *A publicly verifiable non-interactive batch argument $(\mathsf{Gen},\mathsf{P},\mathsf{V})$ for $\mathbf{CSAT}$ is called* ***semi-adaptive somewhere sound*** *if there exists a PPT algorithm TGen that satisfies the following.*

- ***CRS indistinguishability.*** *For every PPT algorithms $(\mathcal{A}_1,\mathcal{A}_2)$ and pair of polynomials poly_k, poly_C, there exists a negligible function negl such that for every $\lambda \in \mathbb{N}$, $k \le \mathsf{poly}_k(\lambda)$, $M_C \le \mathsf{poly}_C(\lambda)$, $n \le M_C$, and $z \in \{0,1\}^*$,*

$$\left|\Pr\left[\mathcal{A}_2(\mathsf{st},\mathsf{crs})=1 \;\middle|\; \begin{array}{l}(\mathsf{st},i^*) \leftarrow \mathcal{A}_1(1^\lambda,1^n,1^{M_C},k,z)\\ \mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda,1^n,1^{M_C},k)\end{array}\right] - \Pr\left[\mathcal{A}_2(\mathsf{st},\mathsf{crs})=1 \;\middle|\; \begin{array}{l}(\mathsf{st},i^*) \leftarrow \mathcal{A}_1(1^\lambda,1^n,1^{M_C},k,z)\\ (\mathsf{crs},\mathsf{td}) \leftarrow \mathsf{TGen}(1^\lambda,1^n,1^{M_C},k,i^*)\end{array}\right]\right| \le \mathsf{negl}(\lambda) \ .$$

- ***Semi-adaptive somewhere soundness.*** *For every* PPT *algorithm* P^* *and pair of polynomials* poly_k, poly_C, *there exists a negligible function* negl *such that for every* $\lambda \in \mathbb{N}$, $k \leq \mathsf{poly}_k(\lambda)$, $M_C \leq \mathsf{poly}_C(\lambda)$, $n \leq M_C$, *and* $z \in \{0,1\}^*$,

$$\Pr\left[\begin{array}{l|l} \mathsf{V}(\mathsf{crs}, C, \mathbf{x}, \pi) = 1 & (\mathsf{st}, i^*) \leftarrow \mathsf{P}_1^*(1^\lambda, 1^n, 1^{M_C}, k, z) \\ \wedge\, i^* \in [k] & \mathsf{crs} \leftarrow \mathsf{TGen}(1^\lambda, 1^n, 1^{M_C}, k, i^*) \\ \wedge\, (C, x_{i^*}) \notin \mathbf{CSAT} & (C, \mathbf{x}, \pi) \leftarrow \mathsf{P}_2^*(\mathsf{st}, \mathsf{crs}), \\ & \quad \textit{where } \mathbf{x} = (x_1, \ldots, x_k) \in (\{0,1\}^n)^k \end{array}\right] \leq \mathsf{negl}(\lambda) \ .$$

Remark 3 (Differences from the original definitions [CJJ21b, Section 4.1]). First, by default the soundness is defined in the non-adaptive setting as in [CJJ21a], and the semi-adaptive soundness is given as an additional security notion. Second, the setup algorithm takes k in binary rather than in unary so that k can be super-polynomial in λ. (Recall that Gen runs in time polylogarithmic in k.) Third, Gen additionally takes n (the length of each instance) as input. (If the instance length is unknown at the setup time, the circuit size $|C|$ can be used as a loose upper bound. By adding dummy gates, any size-M_C circuit can be converted to a $2M_C$-size circuit that takes a length-M_C input.) Finally, we define the public-coin property naturally. $\lozenge$

Weakly semi-adaptive somewhere soundness. We define a new soundness notion that lies between non-adaptive soundness and semi-adaptive somewhere soundness. The difference from semi-adaptive somewhere soundness is that the instances $\mathbf{x}$ are fixed non-adaptively (the circuit C is still chosen adaptively).

Definition 14 (Weakly semi-adaptive somewhere soundness)**.** *A publicly verifiable non-interactive batch argument* $(\mathsf{Gen}, \mathsf{P}, \mathsf{V})$ *for* $\mathbf{CSAT}$ *is called* ***weakly semi-adaptive somewhere sound*** *if there exists a* PPT *algorithm* TGen *that satisfies the following.*

- ***CRS indistinguishability.*** *Identical with the one in Definition 13.*
- ***Weakly semi-adaptive somewhere soundness.*** *For every* PPT *algorithm* P^* *and pair of polynomials* poly_k, poly_C, *there exists a negligible function* negl *such that for every* $\lambda \in \mathbb{N}$, $k \leq \mathsf{poly}_k(\lambda)$, $M_C \leq \mathsf{poly}_C(\lambda)$, $n \leq M_C$, *and* $z \in \{0,1\}^*$,

$$\Pr\left[\begin{array}{l|l} \mathsf{V}(\mathsf{crs}, C, \mathbf{x}, \pi) = 1 & (\mathsf{st}, i^*, \mathbf{x}) \leftarrow \mathsf{P}_1^*(1^\lambda, 1^n, 1^{M_C}, k, z), \\ \wedge\, i^* \in [k] & \quad \textit{where } \mathbf{x} = (x_1, \ldots, x_k) \in (\{0,1\}^n)^k \\ \wedge\, (C, x_{i^*}) \notin \mathbf{CSAT} & \mathsf{crs} \leftarrow \mathsf{TGen}(1^\lambda, 1^n, 1^{M_C}, k, i^*) \\ & (C, \pi) \leftarrow \mathsf{P}_2^*(\mathsf{st}, \mathsf{crs}) \end{array}\right] \leq \mathsf{negl}(\lambda) \ .$$

Non-interactive BARGs for the index language. We recall the definition and a known result of *publicly verifiable non-interactive BARGs for the index language* [CJJ22].

Definition 15 (BARG for index language)**.** ***Publicly verifiable non-interactive batch arguments for the indexed language*** *are a special case of publicly verifiable non-interactive batch arguments for* $\mathbf{CSAT}$ *(Definition 12), with the following differences.*

- ***Syntax.*** *The instances* $\mathbf{x} = (x_1, \ldots, x_k)$ *are fixed to be the indices* $\mathbf{x} = (1, \ldots, k)$, *and they are not given to the prover* P *and the verifier* V. *Also, the instance length* n *is not given to the setup algorithm* Gen. *(It is assumed that* P *and* V *can learn* k *from the common reference string* crs.*)*
- ***Efficiency.*** *The requirement about the running time of the verifier* V *is strengthened to* $\mathsf{poly}(\lambda, M_C, \log k)$.

Theorem 3 ([CJJ22])**.** *Under the LWE assumption, there exists a public-coin non-interactive batch argument for the index language. Furthermore, it satisfies semi-adaptive somewhere soundness.*

3.9 Holographic SNARGs for P and Batch-NP

We define holographic SNARGs for P and Batch-NP by naturally combining the definitions of non-interactive Turing-machine delegations and BARGs with the definitions of holographic interactive proofs/arguments [GR17, BR22].

Definition 16 (Holographic Turing-machine delegation)**.** *Let M be a two-input Turing machine. A publicly verifiable non-interactive delegation scheme $(\mathsf{Gen}, \mathsf{P}, \mathsf{V})$ for M is called **holographic** if there exists a deterministic polynomial-time algorithm Encode such that the execution of the verifier on input $(\mathsf{crs}, (\chi_1, \chi_2), \pi)$ can be written as $\mathsf{V}^{\widehat{\chi}_1}(\mathsf{crs}, (|\chi_1|, \chi_2), \pi)$ for $\widehat{\chi}_1 := \mathsf{Encode}(\lambda, \chi_1)$, where the verifier works in two steps as follows.*

1. *Without making queries to $\widehat{\chi}_1$, the verifier either immediately outputs 0 (i.e., rejects the proof) or computes a set $I \subseteq [\widehat{n}]$ of queries along with a set $Z \subseteq \Sigma^{\widehat{n}}$ of expected responses. (Σ is the alphabet of $\widehat{\chi}_1$ and $\widehat{n}$ is the block length.) This step takes time at most $O(L_\pi) + \mathsf{poly}(\lambda, \log|\chi_1|, |\chi_2|)$, where L_π is the length of the proof π.*
2. *The verifier makes the queries to $\widehat{\chi}_1$, and it outputs 1 iff $\widehat{\chi}_1|_I = Z$.*

Definition 17 (Holographic batch argument)**.** *A publicly verifiable non-interactive batch argument $(\mathsf{Gen}, \mathsf{P}, \mathsf{V})$ is called **holographic** if there exists a deterministic polynomial-time algorithm Encode such that the execution of the verifier on input $(\mathsf{crs}, C, \mathbf{x}, \pi)$ can be written as $\mathsf{V}^{\widehat{\mathbf{x}}}(\mathsf{crs}, C, k, \pi)$ for $\widehat{\mathbf{x}} := \mathsf{Encode}(\lambda, \mathbf{x})$, where the verifier proceeds in two steps as follows.*

1. *Without making queries to $\widehat{\mathbf{x}}$, the verifier either immediately outputs 0 (i.e., rejects the proof) or computes a set $I \subseteq [\widehat{n}]$ of queries along with a set $Z \subseteq \Sigma^{\widehat{n}}$ of expected responses. (Σ is the alphabet of $\widehat{\mathbf{x}}$ and $\widehat{n}$ is the block length.) This step takes time at most $\mathsf{poly}(\lambda, |C|, \log k)$.*
2. *The verifier makes the queries to $\widehat{\mathbf{x}}$, and it outputs 1 iff $\widehat{\mathbf{x}}|_I = Z$.*

The following is a special case of the above definitions.

Definition 18 (LDE-holographic delegation)**.** *A publicly verifiable non-interactive delegation scheme (resp., a publicly verifiable non-interactive batch argument) is called **LDE-holographic** if it is holographic w.r.t. the encoding algorithm Encode that outputs the low-degree extension $\mathsf{LDE}_{\mathbb{F},\mathbb{H},m}(\chi)$ of the input χ for the parameter $(\mathbb{F}, \mathbb{H}, m)$ that is determined based on $(\lambda, |\chi|)$ (resp., the low-degree extension $\mathsf{LDE}_{\mathbb{F},\mathbb{H},m}(\mathbf{x})$ of the instances $\mathbf{x} = (x_1, \ldots, x_k)$ for the parameter $(\mathbb{F}, \mathbb{H}, m)$ that is determined based on $(\lambda, |x_1| + \cdots + |x_k|)$, where $\mathbf{x}$ is viewed as a binary string $x_1 \| \cdots \| x_k$).*

4 Somewhere-Sound Holographic SNARG for Somewhere-Extractable Hashing

In this section, we construct a specific type of holographic SNARGs that we use as a tool in the subsequent sections. The target of this section is defined as follows based on the definition of holographic publicly verifiable non-interactive Turing-machine delegations schemes (Definition 8, Definition 16).

Definition 19 (Partially adaptive somewhere-sound holographic delegation for SE hash)**.** *Let $\mathsf{SEH} = (\mathsf{SEH.Gen}, \mathsf{SEH.TGen}, \mathsf{SEH.Hash}, \mathsf{SEH.Open}, \mathsf{SEH.Verify}, \mathsf{SEH.Extract})$ be a somewhere extractable hash function family. A **partially adaptive somewhere-sound holographic non-interactive delegation scheme for** SEH consists of four algorithms $(\mathsf{Gen}, \mathsf{P}, \mathsf{V}, \mathsf{Encode})$ satisfying the following.*

- ***Holographic completeness.*** *For every $\lambda \in \mathbb{N}$, $N \in [2^\lambda]$, $x \in \{0,1\}^N$, and $M_I \in [N]$,*

$$\Pr\left[\mathsf{V}^{\widehat{x}}(\mathsf{crs}, (h, \mathsf{rt}), \pi) = 1 \;\middle|\; \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda, N, 1^{M_I}) \\ h \leftarrow \mathsf{SEH.Gen}(1^\lambda, N, 1^{M_I}) \\ \mathsf{rt} := \mathsf{SEH.Hash}(h, x) \\ \pi := \mathsf{P}(\mathsf{crs}, (x, (h, \mathsf{rt}))) \\ \widehat{x} := \mathsf{Encode}(\lambda, x) \end{array}\right] = 1 \ .$$

Furthermore, the execution of $\mathsf{V}^{\widehat{x}}(\mathsf{crs}, (h, \mathsf{rt}), \pi)$ proceeds in two steps as specified in the definition of holographic delegations (Definition 16).

- ***Partially adaptive somewhere soundness.*** *For every* PPT *algorithm* P^* *and pair of polynomials* poly_N, poly_I, *there exists a negligible function* negl *such that for every* $\lambda \in \mathbb{N}$, $N \in [2^\lambda]$, $x = (x_1, \ldots, x_N) \in \{0,1\}^N$, $I \subseteq [N]$, *and* $z \in \{0,1\}^*$ *such that* $N \le \mathsf{poly}_N(\lambda)$ *and* $|I| \le \mathsf{poly}_I(\lambda)$,

$$\Pr\left[\begin{array}{l|l} \mathsf{V}^{\widehat{x}}(\mathsf{crs}, (h, \mathsf{rt}), \pi) = 1 & \mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda, N, 1^{|I|}) \\ \wedge\, \exists i \in I \text{ s.t. } x_i \neq \widetilde{x}_i & (h, \mathsf{td}) \leftarrow \mathsf{SEH.TGen}(1^\lambda, N, I) \\ & (\mathsf{rt}, \pi) \leftarrow \mathsf{P}^*(\mathsf{crs}, (x, h), z) \\ & \widehat{x} := \mathsf{Encode}(\lambda, x) \\ & \{\widetilde{x}_i\}_{i \in I} := \mathsf{SEH.Extract}(\mathsf{td}, \mathsf{rt}) \end{array}\right] \le \mathsf{negl}(\lambda) \ .$$

- ***Efficiency.*** *In the above completeness experiment, the following hold.*
  - *The setup algorithm* Gen *runs in time* $\mathsf{poly}(\lambda, \log N, M_I)$.
  - *The prover* P *runs in time* $\mathsf{poly}(\lambda, N)$ *and outputs a proof* π *of length* $\mathsf{poly}(\lambda, \log N, M_I)$.
  - *The verifier* V *runs in time* $\mathsf{poly}(\lambda, \log N, M_I)$.

A holographic delegation scheme for SEH *is called* ***public-coin*** *if the setup algorithm* Gen *is public-coin, i.e., it just outputs a string that is sampled uniformly randomly (possibly along with various parameters that are determined deterministically based on the input of* Gen*). A holographic delegation scheme for* SEH *is called* ***LDE-holographic*** *if the encoding algorithm* Encode *outputs the low-degree extension* $\mathsf{LDE}_{\mathbb{F},\mathbb{H},m}(x)$ *of the input* x *for the parameter* $(\mathbb{F}, \mathbb{H}, m)$ *that is determined based on* $(\lambda, |x|)$.

The goal of this section is to prove the following lemma.

Lemma 2. *Under the LWE assumption, there exists a partially adaptive somewhere-sound holographic non-interactive delegation scheme for any somewhere extractable hash function family. The scheme is public-coin and LDE-holographic, where for the security parameter* λ *and an input* x *of length* N*, the encoding algorithm* Encode *outputs the LDE of* x *w.r.t. an arbitrary LDE parameter* $(\mathbb{F}, \mathbb{H}, m)$ *such that* $|\mathbb{F}| \le \mathsf{poly}(|\mathbb{H}|) \le \mathsf{poly}(\log N)$, $|\mathbb{H}|^m \le \mathsf{poly}(N)$, *and* $m|\mathbb{H}|/|\mathbb{F}| \le O(1)$.

Proof. Fix any somewhere extractable hash function family $\mathsf{SEH} = (\mathsf{SEH.Gen}, \mathsf{SEH.TGen}, \mathsf{SEH.Hash}, \mathsf{SEH.Open}, \mathsf{SEH.Verify}, \mathsf{SEH.Extract})$. Our goal is to give a partially adaptive somewhere-sound holographic non-interactive delegation scheme for SEH. For simplicity, we assume SEH is public-coin.[10]

First, we introduce notations. Let $\mathsf{param}_{\mathrm{LDE}}$ be any efficiently computable mapping that maps each $(\lambda, N) \in \mathbb{N} \times \mathbb{N}$ to an LDE parameter $(\mathbb{F}, \mathbb{H}, m)$ such that $|\mathbb{F}| \le \mathsf{poly}(|\mathbb{H}|) \le \mathsf{poly}(\log N)$, $|\mathbb{H}|^m \le \mathsf{poly}(N)$, and $m|\mathbb{H}|/|\mathbb{F}| \le O(1)$.[11] For a security parameter $\lambda \in \mathbb{N}$ and an LDE parameter $(\mathbb{F}, \mathbb{H}, m) := \mathsf{param}_{\mathrm{LDE}}(\lambda, N)$, define m_λ as $m_\lambda := \lceil \log_{|\mathbb{H}|} \lambda \rceil$ so that $\lambda \le |\mathbb{H}|^{m_\lambda} < \lambda|\mathbb{H}|$. (Since $|\mathbb{H}| \le \mathsf{poly}(\log N) \le \mathsf{poly}(\lambda)$, we have $|\mathbb{H}|^{m_\lambda} \le \mathsf{poly}(\lambda)$.) We identify $\mathbb{H}^m$ with $\{1, \ldots, |\mathbb{H}|^m\}$ by the lexicographical order. We often implicitly view each element of $\mathbb{H}^m$ as the corresponding element of $\{1, \ldots, |\mathbb{H}|^m\}$.

Next, we introduce the building blocks that we use in our scheme.

- Let $\mathsf{BARG}_{\mathrm{idx}} = (\mathsf{BARG.Gen}_{\mathrm{idx}}, \mathsf{BARG.P}_{\mathrm{idx}}, \mathsf{BARG.V}_{\mathrm{idx}})$ be a semi-adaptive somewhere-sound public-coin non-interactive BARG for the index language.
- For each (arbitrarily small) constant $\delta > 0$ and $\alpha := \lfloor \lambda^\delta \rfloor$, let CIH_α be a public-coin correlation-intractable hash function family that satisfies the following. For sufficiently large polynomials $\mathsf{poly}_X, \mathsf{poly}_Y, \mathsf{poly}_T$ and a constant $\rho \in [0,1]$,[12] (i) the domain-codomain ensemble of CIH_α is $\{(X_\lambda, Y_\lambda^\alpha)\}_{\lambda \in \mathbb{N}}$ for $X_\lambda := \{0,1\}^{\mathsf{poly}_X(\lambda)}$ and $Y_\lambda := \{1, \ldots, \mathsf{poly}_Y(\lambda)\}$, (ii) the correlation intractability of CIH_α holds for any product relation ensemble that is poly_T-time product verifiable with product sparsity ρ, and (iii) CIH_α can be evaluated in time $\mathsf{poly}(\log|X_\lambda|, |Y_\lambda|, \alpha, \mathsf{poly}_T(\lambda)) = \mathsf{poly}(\lambda, \alpha)$.

[10] If not, it suffices to additionally use any LWE-based somewhere extractable hash function family (cf. Theorem 2) as a building block in our scheme.

[11] E.g, $|\mathbb{H}| := \lceil \log N \rceil$, $|\mathbb{F}| := \mathsf{poly}(|\mathbb{H}|)$, and $m := \lceil \log_{|\mathbb{H}|} N \rceil$.

[12] The concrete requirements about poly_X, poly_Y, poly_T, and ρ are determined based on SEH and $\mathsf{param}_{\mathrm{LDE}}$ (cf. the proof of Claim 3).

$\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda, N, 1^{M_I})$:

1. Let $(\mathbb{F}, \mathbb{H}, m_N) := \mathsf{param}_{\mathrm{LDE}}(\lambda, N)$.
2. Sample $h' \leftarrow \mathsf{SEH.Gen}(1^\lambda, |\mathbb{H}|^{m_N}, 1^{M'_I})$, where $M'_I := 1$.
3. Sample $\mathsf{crs}^{\mathrm{idx}} \leftarrow \mathsf{BARG.Gen}_{\mathrm{idx}}(1^\lambda, 1^{M_C}, N)$, where $M_C = \mathsf{poly}(\lambda, \log N, M_I)$ is the size of the circuit C that is defined in the prover P below.
4. Sample $\mathsf{crs}' \leftarrow \mathsf{Gen}_{\mathrm{sub}}(1^\lambda, (\mathbb{F}, \mathbb{H}, m_N), \alpha)$, where $\alpha := \lfloor \lambda^{1/(\lfloor m_N/m_\lambda \rfloor + 1)} \rfloor$.
5. Output $\mathsf{crs} \leftarrow (1^\lambda, N, h', \mathsf{crs}^{\mathrm{idx}}, \mathsf{crs}')$.

$\pi := \mathsf{P}(\mathsf{crs}, (x, (h, \mathsf{rt})))$:

1. Parse crs as $(1^\lambda, N, h', \mathsf{crs}^{\mathrm{idx}}, \mathsf{crs}')$. Let $(\mathbb{F}, \mathbb{H}, m_N) := \mathsf{param}_{\mathrm{LDE}}(\lambda, N)$.
2. Let $x' : \mathbb{H}^{m_N} \to \mathbb{F}$ be the function that is obtained from $x = (x_1, \ldots, x_N) \in \{0,1\}^N$ by letting $x'(i) := x_i$ for $1 \le i \le N$ and $x'(i) := 0$ for $N < i \le |\mathbb{H}|^{m_N}$, where $\mathbb{H}^{m_N}$ is identified with $\{1, \ldots, |\mathbb{H}|^{m_N}\}$ by the lexicographical order and each $x_i \in \{0,1\}$ is viewed as an element of $\mathbb{F}$ in the natural way.
3. Compute $\mathsf{rt}' := \mathsf{SEH.Hash}(h', x')$. Also, compute $\mathsf{cert}_i := \mathsf{SEH.Open}(h, x, i)$ and $\mathsf{cert}'_i := \mathsf{SEH.Open}(h', x', i)$ for $\forall i \in [N]$.
4. Compute $\pi^{\mathrm{idx}} := \mathsf{BARG.P}_{\mathrm{idx}}(\mathsf{crs}^{\mathrm{idx}}, C, \mathbf{w})$, where $\mathbf{w} := \{w_i\}_{i \in [N]}$, $w_i := (x_i, \mathsf{cert}_i, \mathsf{cert}'_i)$, and C is the following circuit.
   - C has $(h, \mathsf{rt}, h', \mathsf{rt}')$ as hardwired inputs, takes an index $i \in [N]$ and a witness $w_i = (x_i, \mathsf{cert}_i, \mathsf{cert}'_i)$ as inputs, and outputs 1 iff $\mathsf{SEH.Verify}(h, \mathsf{rt}, x_i, i, \mathsf{cert}_i) = 1$ and $\mathsf{SEH.Verify}(h', \mathsf{rt}', x_i, i, \mathsf{cert}'_i) = 1$.
5. Compute $\pi' \leftarrow \mathsf{P}_{\mathrm{sub}}(\mathsf{crs}', (x', (h', \mathsf{rt}')))$.
6. Output $\pi := (\mathsf{rt}', \pi^{\mathrm{idx}}, \pi')$.

$b := \mathsf{V}^{\widehat{x}}(\mathsf{crs}, (h, \mathsf{rt}), \pi)$:

1. Parse crs as $(1^\lambda, N, h', \mathsf{crs}^{\mathrm{idx}}, \mathsf{crs}')$ and π as $(\mathsf{rt}', \pi^{\mathrm{idx}}, \pi')$.
2. Output 1 iff $\mathsf{BARG.V}_{\mathrm{idx}}(\mathsf{crs}^{\mathrm{idx}}, C, \pi^{\mathrm{idx}}) = 1$ and $\mathsf{V}^{\widehat{x}}_{\mathrm{sub}}(\mathsf{crs}', (h', \mathsf{rt}'), \pi') = 1$.

$\widehat{x} := \mathsf{Encode}(\lambda, x)$:

1. Output $\widehat{x} := \mathsf{LDE}_{\mathbb{F}, \mathbb{H}, m_N}(x)$, where $(\mathbb{F}, \mathbb{H}, m_N) := \mathsf{param}_{\mathrm{LDE}}(\lambda, N)$ for $N := |x|$.

Figure 3: $\mathsf{SEH\text{-}Del} = (\mathsf{Gen}, \mathsf{P}, \mathsf{V}, \mathsf{Encode})$.

Both of the above exist under the LWE assumption (cf. Theorem 3, Theorem 1, Remark 6).

Now, we describe our scheme $\mathsf{SEH\text{-}Del} = (\mathsf{Gen}, \mathsf{P}, \mathsf{V}, \mathsf{Encode})$ using a subroutine scheme $\mathsf{SEH\text{-}Del}_{\mathrm{sub}} = (\mathsf{Gen}_{\mathrm{sub}}, \mathsf{P}_{\mathrm{sub}}, \mathsf{V}_{\mathrm{sub}})$. Intuitively, $\mathsf{SEH\text{-}Del}_{\mathrm{sub}}$ is a holographic SNARG for SEH w.r.t. strings over finite fields (rather than binary strings), and SEH-Del is a wrapper scheme that enables us to use $\mathsf{SEH\text{-}Del}_{\mathrm{sub}}$ w.r.t. strings over binary strings. That is, given a binary string x and its SE hash value rt, SEH-Del converts x into an equivalent string x' over a finite field, computes the SE hash value rt' of x', and invokes $\mathsf{SEH\text{-}Del}_{\mathrm{sub}}$ for x' and rt' while using $\mathsf{BARG}_{\mathrm{idx}}$ to prove the consistency between rt and rt'. The formal description of SEH-Del is given in Figure 3. The subroutine scheme $\mathsf{SEH\text{-}Del}_{\mathrm{sub}}$ is defined recursively in Figure 4 (a high-level idea is explained in Section 2.1).[13] [14]

In the following, we prove that SEH-Del satisfies the completeness, efficiency, and soundness requirements as described in Definition 19.

4.1 Completeness

To prove the holographic completeness of SEH-Del, we prove the following claim about the completeness of $\mathsf{SEH\text{-}Del}_{\mathrm{sub}}$.

[13] In SEH-Del and $\mathsf{SEH\text{-}Del}_{\mathrm{sub}}$, the somewhere extractable hash function SEH is used for strings over a finite field (cf. Section 3.5).

[14] The recursive structure of $\mathsf{SEH\text{-}Del}_{\mathrm{sub}}$ is the reason why we define it w.r.t. strings over finite fields.

$\underline{\mathsf{crs} \leftarrow \mathsf{Gen}_{\mathrm{sub}}(1^\lambda, (\mathbb{F}, \mathbb{H}, m), \alpha):}$

1. If $m < m_\lambda$, output $\mathsf{crs} \coloneqq (1^\lambda, (\mathbb{F}, \mathbb{H}, m), \bot, \bot, \bot, \bot)$ and terminate. If $m \geq m_\lambda$, continue to the next step.
2. Sample $h^+ \leftarrow \mathsf{SEH.Gen}(1^\lambda, |\mathbb{H}|^{m-m_\lambda}, 1^{M_I})$, where $M_I \coloneqq 1$.
3. Sample $\mathsf{crs}^{\mathrm{idx}} \leftarrow \mathsf{BARG.Gen}_{\mathrm{idx}}(1^\lambda, 1^{M_C}, |\mathbb{H}|^{m-m_\lambda})$, where $M_C \coloneqq \max(|C_{\mathbf{u}}|, |C|)$ for the circuits $C_{\mathbf{u}}, C$ that are defined in the prover $\mathsf{P}_{\mathrm{sub}}$ below.
4. Sample $h^{\mathrm{CIH}} \leftarrow \mathsf{CIH.Gen}_\alpha(1^\lambda)$.
5. Sample $\mathsf{crs}^+ \leftarrow \mathsf{Gen}_{\mathrm{sub}}(1^\lambda, (\mathbb{F}, \mathbb{H}, m - m_\lambda), \alpha)$.
6. Output $\mathsf{crs} \coloneqq (1^\lambda, (\mathbb{F}, \mathbb{H}, m), \alpha, h^+, \mathsf{crs}^{\mathrm{idx}}, h^{\mathrm{CIH}}, \mathsf{crs}^+)$.

$\underline{\pi \leftarrow \mathsf{P}_{\mathrm{sub}}(\mathsf{crs}, (x, (h, \mathsf{rt}))):}$

1. Parse crs as $(1^\lambda, (\mathbb{F}, \mathbb{H}, m), \alpha, h^+, \mathsf{crs}^{\mathrm{idx}}, h^{\mathrm{CIH}}, \mathsf{crs}^+)$, where $(\mathbb{F}, \mathbb{H}, m)$ is expected to be an LDE parameter such that $x : \mathbb{H}^m \to \mathbb{F}$. If $m < m_\lambda$, output $\pi \coloneqq x$ and terminate. If $m \geq m_\lambda$, compute $\widehat{x} \coloneqq \mathsf{LDE}_{\mathbb{F},\mathbb{H},m}(x)$ and continue to the next step.
2. Compute $\mathsf{rt}^+_{\mathbf{u}} \coloneqq \mathsf{SEH.Hash}(h^+, x_{(\mathbf{u},*)})$ for $\forall \mathbf{u} \in \mathbb{F}^{m_\lambda}$, where $x_{(\mathbf{u},*)} : \mathbb{H}^{m-m_\lambda} \to \mathbb{F}$ is defined as $x_{(\mathbf{u},*)} : \mathbf{v} \mapsto \widehat{x}(\mathbf{u}, \mathbf{v})$. Also, compute $\mathsf{cert}_{(\mathbf{u},\mathbf{v})} \coloneqq \mathsf{SEH.Open}(h, x, (\mathbf{u}, \mathbf{v}))$ for $\forall (\mathbf{u}, \mathbf{v}) \in \mathbb{H}^{m_\lambda} \times \mathbb{H}^{m-m_\lambda}$ and $\mathsf{cert}^+_{(\mathbf{u},\mathbf{v})} \coloneqq \mathsf{SEH.Open}(h^+, x_{(\mathbf{u},*)}, \mathbf{v})$ for $\forall (\mathbf{u}, \mathbf{v}) \in \mathbb{F}^{m_\lambda} \times \mathbb{H}^{m-m_\lambda}$.
3. Compute $\pi^{\mathrm{idx}}_{\mathbf{u}} \coloneqq \mathsf{BARG.P}_{\mathrm{idx}}(\mathsf{crs}^{\mathrm{idx}}, C_{\mathbf{u}}, \mathbf{w})$ for $\forall \mathbf{u} \in \mathbb{H}^{m_\lambda}$, where $\mathbf{w} \coloneqq \{w_{\mathbf{v}}\}_{\mathbf{v} \in \mathbb{H}^{m-m_\lambda}}$, $w_{\mathbf{v}} \coloneqq (x(\mathbf{u}, \mathbf{v}), \mathsf{cert}_{(\mathbf{u},\mathbf{v})}, \mathsf{cert}^+_{(\mathbf{u},\mathbf{v})})$, and $C_{\mathbf{u}}$ is the following circuit.
   - $C_{\mathbf{u}}$ has $(\mathbf{u}, h, \mathsf{rt}, h^+, \mathsf{rt}^+_{\mathbf{u}})$ as hardwired inputs, takes an index $\mathbf{v}$ and a witness $w = (y, \mathsf{cert}, \mathsf{cert}^+)$ as inputs, and outputs 1 iff $\mathsf{SEH.Verify}(h, \mathsf{rt}, y, (\mathbf{u}, \mathbf{v}), \mathsf{cert}) = 1$ and $\mathsf{SEH.Verify}(h^+, \mathsf{rt}^+_{\mathbf{u}}, y, \mathbf{v}, \mathsf{cert}') = 1$.
4. Compute $\pi^{\mathrm{idx}} \coloneqq \mathsf{BARG.P}_{\mathrm{idx}}(\mathsf{crs}^{\mathrm{idx}}, C, \mathbf{w})$, where $\mathbf{w} \coloneqq \{w_{\mathbf{v}}\}_{\mathbf{v} \in \mathbb{H}^{m-m_\lambda}}$, $w_{\mathbf{v}} = \{\widehat{x}(\mathbf{u}, \mathbf{v}), \mathsf{cert}^+_{(\mathbf{u},\mathbf{v})}\}_{\mathbf{u} \in \mathbb{F}^{m_\lambda}}$, and C is the following circuit.
   - C has $(h^+, \{\mathsf{rt}^+_{\mathbf{u}}\}_{\mathbf{u} \in \mathbb{F}^{m_\lambda}})$ as hardwired inputs, takes an index $\mathbf{v} \in \mathbb{H}^{m-m_\lambda}$ and a witness $w = \{y_{(\mathbf{u},\mathbf{v})}, \mathsf{cert}_{(\mathbf{u},\mathbf{v})}\}_{\mathbf{u} \in \mathbb{F}^{m_\lambda}}$ as inputs, and outputs 1 iff (i) $\mathsf{SEH.Verify}(h^+, \mathsf{rt}^+_{\mathbf{u}}, y_{(\mathbf{u},\mathbf{v})}, \mathbf{v}, \mathsf{cert}_{(\mathbf{u},\mathbf{v})}) = 1$ for $\forall \mathbf{u} \in \mathbb{F}^{m_\lambda}$ and (ii) the function $\widehat{y}_{(*,\mathbf{v})} : \mathbb{F}^{m_\lambda} \to \mathbb{F}$ defined as $\widehat{y}_{(*,\mathbf{v})} : \mathbf{u} \mapsto y_{(\mathbf{u},\mathbf{v})}$ is the LDE of $y_{(*,\mathbf{v})} \coloneqq \widehat{y}_{(*,\mathbf{v})}|_{\mathbb{H}^{m_\lambda}}$ w.r.t. $(\mathbb{F}, \mathbb{H}, m_\lambda)$.
5. Compute $(\mathbf{c}_1, \ldots, \mathbf{c}_\alpha) \coloneqq \mathsf{CIH.Hash}_\alpha(h^{\mathrm{CIH}}, \{\mathsf{rt}^+_{\mathbf{u}}\}_{\mathbf{u} \in \mathbb{H}^{m_\lambda}})$, where each $\mathbf{c}_i$ is viewed as an element of $\mathbb{F}^{m_\lambda}$. Let $S_{\mathbf{c}} \coloneqq \{\mathbf{c}_1, \ldots, \mathbf{c}_\alpha\}$. (It is assumed that the domain $X_\lambda = \{0,1\}^{\mathsf{poly}_X(\lambda)}$ of CIH is large enough so that $\{\mathsf{rt}^+_{\mathbf{u}}\}_{\mathbf{u} \in \mathbb{H}^{m_\lambda}}$ can be encoded as an element of X_λ in a canonical way.)
6. Compute $\pi^+_{\mathbf{c}_i} \leftarrow \mathsf{P}_{\mathrm{sub}}(\mathsf{crs}^+, (x_{(\mathbf{c}_i,*)}, (h^+, \mathsf{rt}^+_{\mathbf{c}_i})))$ for $\forall \mathbf{c}_i \in S_{\mathbf{c}}$.
7. Output $\pi \coloneqq (\{\mathsf{rt}^+_{\mathbf{u}}\}_{\mathbf{u} \in \mathbb{F}^{m_\lambda}}, \{\pi^{\mathrm{idx}}_{\mathbf{u}}\}_{\mathbf{u} \in \mathbb{H}^{m_\lambda}}, \pi^{\mathrm{idx}}, \{\pi^+_{\mathbf{c}_i}\}_{\mathbf{c}_i \in S_{\mathbf{c}}})$.

$\underline{b \coloneqq \mathsf{V}^{\widehat{x}}_{\mathrm{sub}}(\mathsf{crs}, (h, \mathsf{rt}), \pi):}$

1. Parse crs as $(1^\lambda, (\mathbb{F}, \mathbb{H}, m), \alpha, h^+, \mathsf{crs}^{\mathrm{idx}}, h^{\mathrm{CIH}}, \mathsf{crs}^+)$. If $m < m_\lambda$, output 1 iff $\mathsf{SEH.Hash}(h, \pi) = \mathsf{rt}$ and $\pi = \widehat{x}|_{\mathbb{H}^m}$. If $m \geq m_\lambda$, continue to the next step.
2. Parse π as $(\{\mathsf{rt}^+_{\mathbf{u}}\}_{\mathbf{u} \in \mathbb{F}^{m_\lambda}}, \{\pi^{\mathrm{idx}}_{\mathbf{u}}\}_{\mathbf{u} \in \mathbb{H}^{m_\lambda}}, \pi^{\mathrm{idx}}, \{\pi^+_{\mathbf{c}_i}\}_{\mathbf{c}_i \in S_{\mathbf{c}}})$.
3. Compute $(\mathbf{c}_1, \ldots, \mathbf{c}_\alpha) \coloneqq \mathsf{CIH.Hash}_\alpha(h^{\mathrm{CIH}}, \{\mathsf{rt}^+_{\mathbf{u}}\}_{\mathbf{u} \in \mathbb{H}^{m_\lambda}})$, where each $\mathbf{c}_i$ is viewed as an element of $\mathbb{F}^{m_\lambda}$. Let $S_{\mathbf{c}} \coloneqq \{\mathbf{c}_1, \ldots, \mathbf{c}_\alpha\}$.
4. Output 1 iff all of the following hold.
   - $\mathsf{BARG.V}_{\mathrm{idx}}(\mathsf{crs}^{\mathrm{idx}}, C_{\mathbf{u}}, \pi^{\mathrm{idx}}_{\mathbf{u}}) = 1$ for $\forall \mathbf{u} \in \mathbb{H}^{m_\lambda}$, where $C_{\mathbf{u}}$ is defined as in $\mathsf{P}_{\mathrm{sub}}$.
   - $\mathsf{BARG.V}_{\mathrm{idx}}(\mathsf{crs}^{\mathrm{idx}}, C, \pi^{\mathrm{idx}}) = 1$, where C is defined as in $\mathsf{P}_{\mathrm{sub}}$.
   - $\mathsf{V}^{\widehat{x}(\mathbf{c}_i,\cdot)}_{\mathrm{sub}}(\mathsf{crs}^+, (h^+, \mathsf{rt}^+_{\mathbf{c}_i}), \pi^+_{\mathbf{c}_i}) = 1$ for $\forall \mathbf{c}_i \in S_{\mathbf{c}}$.

Figure 4: $\mathsf{SEH\text{-}Del}_{\mathrm{sub}} = (\mathsf{Gen}_{\mathrm{sub}}, \mathsf{P}_{\mathrm{sub}}, \mathsf{V}_{\mathrm{sub}})$.

Claim 1 (LDE-holographic completeness of $\mathsf{SEH\text{-}Del}_{\mathrm{sub}}$). *For every* $\lambda \in \mathbb{N}$, $N \in [2^\lambda]$, $(\mathbb{F}, \mathbb{H}, m_N) := \mathsf{param}_{\mathrm{LDE}}(\lambda, N)$, $m \in \{m_N, m_N - m_\lambda, \ldots, m_N - \lfloor m_N/m_\lambda \rfloor \cdot m_\lambda\}$, $x : \mathbb{H}^m \to \mathbb{F}$, *and* $\alpha := \lfloor \lambda^{1/(\lfloor m_N/m_\lambda \rfloor + 1)} \rfloor$,

$$\Pr\left[\mathsf{V}^{\widehat{x}}_{\mathrm{sub}}(\mathsf{crs}, (h, \mathsf{rt}), \pi) = 1 \;\middle|\; \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{Gen}_{\mathrm{sub}}(1^\lambda, (\mathbb{F}, \mathbb{H}, m), \alpha) \\ h \leftarrow \mathsf{SEH.Gen}(1^\lambda, |\mathbb{H}|^m, 1) \\ \mathsf{rt} := \mathsf{SEH.Hash}(h, x) \\ \pi \leftarrow \mathsf{P}_{\mathrm{sub}}(\mathsf{crs}, (x, (h, \mathsf{rt}))) \\ \widehat{x} := \mathsf{LDE}_{\mathbb{F},\mathbb{H},m}(x) \end{array}\right] = 1 \ .$$

Clearly, the completeness of SEH-Del (Definition 19) follows from Claim 1 by setting $m := m_N$.[15] (The furthermore part of the completeness condition, i.e., that V proceeds in two steps where it makes non-adaptive queries to $\widehat{x}$ at the last moment, can be verified by inspection by unfolding the recursive executions of $\mathsf{V}_{\mathrm{sub}}$.) Thus, it remains to prove Claim 1.

Proof of Claim 1. We prove the claim by induction on m, where the base case is $m = m_N - \lfloor m_N/m_\lambda \rfloor \cdot m_\lambda$. For the base case, the claim trivially holds since we have $m < m_\lambda$ in this case. For the inductive step, assume that the claim holds for $m = m' - m_\lambda$ for some $m' \in \{m_N, m_N - m_\lambda, \ldots, m_N - (\lfloor m_N/m_\lambda \rfloor - 1) \cdot m_\lambda\}$. Then, the claim for the case of $m = m'$ can be verified by inspection by relying on Lemma 1. Specifically, the verifier $\mathsf{V}^{\widehat{x}}_{\mathrm{sub}}$ accepts the proof π since

1. the BARG verifier $\mathsf{BARG.V}_{\mathrm{idx}}$ accepts the proof π^{idx} since for every $\mathbf{v} \in \mathbb{H}^{m-m_\lambda}$, the function $\widehat{x}_{(*,\mathbf{v})} : \mathbb{F}^{m_\lambda} \to \mathbb{F}$ defined as $\widehat{x}_{(*,\mathbf{v})} : \mathbf{u} \mapsto \widehat{x}(\mathbf{u}, \mathbf{v})$ is the LDE of $x_{(*,\mathbf{v})} := \widehat{x}_{(*,\mathbf{v})}|_{\mathbb{H}^{m_\lambda}}$ w.r.t. $(\mathbb{F}, \mathbb{H}, m_\lambda)$, and
2. for each $\mathbf{c}_i \in S_{\mathbf{c}}$, the recursive execution $\mathsf{V}^{\widehat{x}(\mathbf{c}_i,\cdot)}_{\mathrm{sub}}$ accepts the proof $\pi^+_{\mathbf{c}_i}$ since the function $\widehat{x}(\mathbf{c}_i, \cdot)$ is the LDE of $x_{(\mathbf{c}_i,*)}$ w.r.t. $(\mathbb{F}, \mathbb{H}, m - m_\lambda)$.

□

4.2 Efficiency

Note that Gen, P, and V recursively execute $\mathsf{Gen}_{\mathrm{sub}}$, $\mathsf{P}_{\mathrm{sub}}$, and $\mathsf{V}_{\mathrm{sub}}$ at most $\alpha^{\lfloor m_N/m_\lambda \rfloor + 1}$ times in total, and we have $\alpha^{\lfloor m_N/m_\lambda \rfloor + 1} = (\lfloor \lambda^{1/(\lfloor m_N/m_\lambda \rfloor + 1)} \rfloor)^{\lfloor m_N/m_\lambda \rfloor + 1} \leq \lambda$. Thus, to prove the efficiency of SEH-Del, it suffices to observe that (i) $\mathsf{Gen}_{\mathrm{sub}}$, $\mathsf{P}_{\mathrm{sub}}$, and $\mathsf{V}_{\mathrm{sub}}$ run in time at most $\mathsf{poly}(\lambda, \log|\mathbb{H}|^{m_N})$, $\mathsf{poly}(\lambda, |\mathbb{H}|^{m_N})$, and $\mathsf{poly}(\lambda, \log|\mathbb{H}|^{m_N})$ respectively other than the recursive executions, and (ii) $\mathsf{P}_{\mathrm{sub}}$ outputs a proof of length at most $\mathsf{poly}(\lambda, \log|\mathbb{H}|^{m_N})$ other than the recursive proofs (each poly is independent of m). These two can be verified by inspection by observing that the circuits $C_{\mathbf{u}}$ and C have size at most $\mathsf{poly}(\lambda, \log|\mathbb{H}|^{m_N})$.

4.3 Partially Adaptive Somewhere Soundness

We prove the partially adaptive somewhere soundness of SEH-Del by proving a related soundness notion for $\mathsf{SEH\text{-}Del}_{\mathrm{sub}}$. Concretely, we consider the following two claims about $\mathsf{SEH\text{-}Del}_{\mathrm{sub}}$.

Claim 2 (Base case). *For every* PPT *algorithm* P^* *and polynomial* poly_N, *there exists a negligible function* negl *such that for every* $\lambda \in \mathbb{N}$, $N \leq \mathsf{poly}_N(\lambda)$, $(\mathbb{F}, \mathbb{H}, m_N) := \mathsf{param}_{\mathrm{LDE}}(\lambda, N)$, $m := m_N - \lfloor m_N/m_\lambda \rfloor \cdot m_\lambda$, $x : \mathbb{H}^m \to \mathbb{F}$, $i^* \in [|\mathbb{H}|^m]$, $\alpha := \lfloor \lambda^{1/(\lfloor m_N/m_\lambda \rfloor + 1)} \rfloor$, *and* $z \in \{0,1\}^*$,

$$\Pr\left[\begin{array}{l} \mathsf{V}^{\widehat{x}}_{\mathrm{sub}}(\mathsf{crs}, (h, \mathsf{rt}), \pi) = 1 \\ \wedge\, x(i^*) \neq \widetilde{x}_{i^*} \end{array} \;\middle|\; \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{Gen}_{\mathrm{sub}}(1^\lambda, (\mathbb{F}, \mathbb{H}, m), \alpha) \\ (h, \mathsf{td}) \leftarrow \mathsf{SEH.TGen}(1^\lambda, |\mathbb{H}|^m, \{i^*\}) \\ (\mathsf{rt}, \pi) \leftarrow \mathsf{P}^*(\mathsf{crs}, (x, h), z) \\ \widehat{x} := \mathsf{LDE}_{\mathbb{F},\mathbb{H},m}(x) \\ \widetilde{x}_{i^*} := \mathsf{SEH.Extract}(\mathsf{td}, \mathsf{rt}) \end{array}\right] \leq \mathsf{negl}(\lambda) \ . \tag{2}$$

[15] In the verifier V, Claim 1 guarantees that the subroutine verifier $\mathsf{V}_{\mathrm{sub}}$ outputs 1 since $\widehat{x} = \mathsf{LDE}_{\mathbb{F},\mathbb{H},m}(x)$ is equal to $\mathsf{LDE}_{\mathbb{F},\mathbb{H},m}(x')$.

Claim 3 (Inductive step)**.** *For every* PPT *algorithm* P^* *and polynomial* poly_N*, there exists a* PPT *algorithm* P^+ *and a negligible function* negl *such that the following holds. Assume there exists a polynomial* $\mathsf{poly}_{\mathsf{P}^*}$ *such that for infinitely many* $\lambda \in \mathbb{N}$*, there exist* $N \le \mathsf{poly}_N(\lambda)$*,* $(\mathbb{F}, \mathbb{H}, m_N) := \mathsf{param}_{\mathrm{LDE}}(\lambda, N)$*,* $m \in \{m_N, m_N - m_\lambda, \ldots, m_N - (\lfloor m_N/m_\lambda \rfloor - 1) \cdot m_\lambda\}$*,* $x : \mathbb{H}^m \to \mathbb{F}$*,* $i^* \in [|\mathbb{H}|^m]$*,* $\alpha := \lfloor \lambda^{1/(\lfloor m_N/m_\lambda \rfloor + 1)} \rfloor$*, and* $z \in \{0,1\}^*$ *such that*

$$\Pr\left[\begin{array}{l} \mathsf{V}^{\widehat{x}}_{\mathrm{sub}}(\mathsf{crs}, (h, \mathsf{rt}), \pi) = 1 \\ \wedge\, x(i^*) \neq \widetilde{x}_{i^*} \end{array} \middle| \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{Gen}_{\mathrm{sub}}(1^\lambda, (\mathbb{F}, \mathbb{H}, m), \alpha) \\ (h, \mathsf{td}) \leftarrow \mathsf{SEH.TGen}(1^\lambda, |\mathbb{H}|^m, \{i^*\}) \\ (\mathsf{rt}, \pi) \leftarrow \mathsf{P}^*(\mathsf{crs}, (x, h), z) \\ \widehat{x} := \mathsf{LDE}_{\mathbb{F},\mathbb{H},m}(x) \\ \widetilde{x}_{i^*} := \mathsf{SEH.Extract}(\mathsf{td}, \mathsf{rt}) \end{array}\right] \ge \frac{1}{\mathsf{poly}_{\mathsf{P}^*}(\lambda)} \ . \tag{3}$$

Then, for such a polynomial $\mathsf{poly}_{\mathsf{P}^*}$ *and for infinitely many such* λ*,* N*,* $(\mathbb{F}, \mathbb{H}, m_N)$*,* m*, and* α*, there exist* $x^+ : \mathbb{H}^{m-m_\lambda} \to \mathbb{F}$*,* $i^+ \in [|\mathbb{H}|^{m-m_\lambda}]$*, and* $z^+ \in \{0,1\}^*$ *such that*

$$\Pr\left[\begin{array}{l} \mathsf{V}^{\widehat{x}^+}_{\mathrm{sub}}(\mathsf{crs}, (h, \mathsf{rt}), \pi) = 1 \\ \wedge\, x^+(i^*) \neq \widetilde{x}_{i^+} \end{array} \middle| \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{Gen}_{\mathrm{sub}}(1^\lambda, (\mathbb{F}, \mathbb{H}, m - m_\lambda), \alpha) \\ (h, \mathsf{td}) \leftarrow \mathsf{SEH.TGen}(1^\lambda, |\mathbb{H}|^{m-m_\lambda}, \{i^+\}) \\ (\mathsf{rt}, \pi) \leftarrow \mathsf{P}^+(\mathsf{crs}, (x^+, h), z^+) \\ \widehat{x}^+ := \mathsf{LDE}_{\mathbb{F},\mathbb{H},m}(x^+) \\ \widetilde{x}_{i^+} := \mathsf{SEH.Extract}(\mathsf{td}, \mathsf{rt}) \end{array}\right] \ge \frac{1}{|\mathbb{F}|^{m_\lambda}} \left(\frac{1}{\mathsf{poly}_{\mathsf{P}^*}(\lambda)} - \mathsf{negl}(\lambda)\right) \ . \tag{4}$$

Furthermore, the running time of P^+ *is upper bounded by* $T_{\mathsf{P}^*}(\lambda) + \mathsf{poly}(\lambda)$*, where* T_{P^*} *is the running time of* P^* *and* poly *is a polynomial that is independent of* T_{P^*}*.*

Before proving Claim 2 and Claim 3, we use them to complete the proof of the partially adaptive somewhere soundness of SEH-Del.

Assume for contradiction that SEH-Del is not partially adaptive somewhere sound, i.e., there exist a PPT algorithm P^*, pair of polynomials poly_N, poly_I, and polynomial $\mathsf{poly}_{\mathsf{P}^*}$ such that the following holds: for infinitely many $\lambda \in \mathbb{N}$, there exist $N \in [2^\lambda]$, $x \in \{0,1\}^N$, $I \subseteq [N]$, and $z \in \{0,1\}^*$ such that $N \le \mathsf{poly}_N(\lambda)$, $|I| \le \mathsf{poly}_I(\lambda)$, and

$$\Pr\left[\begin{array}{l} \mathsf{V}^{\widehat{x}}(\mathsf{crs}, (h, \mathsf{rt}), \pi) = 1 \\ \wedge\, \exists i \in I \text{ s.t. } x(i) \neq \widetilde{x}_i \end{array} \middle| \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda, N, 1^{|I|}) \\ (h, \mathsf{td}) \leftarrow \mathsf{SEH.TGen}(1^\lambda, N, I) \\ (\mathsf{rt}, \pi) \leftarrow \mathsf{P}^*(\mathsf{crs}, (x, h), z) \\ \widehat{x} := \mathsf{Encode}(\lambda, x) \\ \{\widetilde{x}_i\}_{i \in I} := \mathsf{SEH.Extract}(\mathsf{td}, \mathsf{rt}) \end{array}\right] \ge \frac{1}{\mathsf{poly}_{\mathsf{P}^*}(\lambda)} \ . \tag{5}$$

Fix any such P^*, poly_N, poly_I, and $\mathsf{poly}_{\mathsf{P}^*}$. It follows from the average argument that for infinitely many λ, N, x, I, and z as above, there exists $i^* \in I$ such that

$$\Pr\left[\begin{array}{l} \mathsf{V}^{\widehat{x}}(\mathsf{crs}, (h, \mathsf{rt}), \pi) = 1 \\ \wedge\, x(i^*) \neq \widetilde{x}_{i^*} \end{array} \middle| \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda, N, 1^{|I|}) \\ (h, \mathsf{td}) \leftarrow \mathsf{SEH.TGen}(1^\lambda, N, I) \\ (\mathsf{rt}, \pi) \leftarrow \mathsf{P}^*(\mathsf{crs}, (x, h), z) \\ \widehat{x} := \mathsf{Encode}(\lambda, x) \\ \{\widetilde{x}_i\}_{i \in I} := \mathsf{SEH.Extract}(\mathsf{td}, \mathsf{rt}) \end{array}\right] \ge \frac{1}{|I| \cdot \mathsf{poly}_{\mathsf{P}^*}(\lambda)} \ . \tag{6}$$

Let $\mathsf{Gen}_{\mathrm{hyb}}$ and $\mathsf{V}_{\mathrm{hyb}}$ be the hybrid setup and verifier that are defined in Figure 5. (The differences from Gen and V are highlighted by colored backgrounds.)

First, we replace Gen with $\mathsf{Gen}_{\mathrm{hyb}}$ in the probability experiment of (6). Note that $\mathsf{Gen}_{\mathrm{hyb}}$ differs from Gen in that it uses the trapdoor setup algorithms of SEH and $\mathsf{BARG}_{\mathrm{idx}}$. Also, note that the experiment of (6) (including the decision of whether the event $\mathsf{V}^{\widehat{x}}(\mathsf{crs}, (h, \mathsf{rt}), \pi) = 1 \wedge x(i^*) \neq \widetilde{x}_{i^*}$ occurs or not) can be efficiently emulated using arbitrary h' and $\mathsf{crs}^{\mathrm{idx}}$ in Gen. Thus, from a standard hybrid argument, there exists a negligible function

negl such that for infinitely many λ, N, x, I, z, and i^* as above,

$$\Pr\left[\begin{array}{l|l} \mathsf{V}^{\widehat{x}}(\mathsf{crs},(h,\mathsf{rt}),\pi)=1 & (\mathsf{crs},\mathsf{td}')\leftarrow\mathsf{Gen}_{\mathrm{hyb}}(1^\lambda,N,1^{|I|}) \\ \wedge\, x(i^*)\neq\widetilde{x}_{i^*} & (h,\mathsf{td})\leftarrow\mathsf{SEH.TGen}(1^\lambda,N,I) \\ & (\mathsf{rt},\pi)\leftarrow\mathsf{P}^*(\mathsf{crs},(x,h),z) \\ & \widehat{x}:=\mathsf{Encode}(\lambda,x) \\ & \{\widetilde{x}_i\}_{i\in I}:=\mathsf{SEH.Extract}(\mathsf{td},\mathsf{rt}) \end{array}\right] \geq \frac{1}{|I|\cdot\mathsf{poly}_{\mathsf{P}^*}(\lambda)}-\mathsf{negl}(\lambda)\ . \tag{7}$$

Next, let us replace V with $\mathsf{V}_{\mathrm{hyb}}$ in (7). To show that the probability in (7) only decreases by a negligible amount, it suffices to show that the event $\mathsf{BARG.V}_{\mathrm{idx}}(\mathsf{crs}^{\mathrm{idx}},C,\pi^{\mathrm{idx}})=1\wedge\widetilde{x}_{i^*}\neq\widetilde{x}'_{i^*}$ occurs in the last step of $\mathsf{V}_{\mathrm{hyb}}$ with negligible probability. (Indeed, the outputs of V and $\mathsf{V}_{\mathrm{hyb}}$ are identical unless this event occurs.) First, the semi-adaptive somewhere soundness of $\mathsf{BARG}_{\mathrm{idx}}$ guarantees that when $\mathsf{BARG.V}_{\mathrm{idx}}(\mathsf{crs}^{\mathrm{idx}},C,\pi^{\mathrm{idx}})=1$, there exists $w_{i^*}=(x_{i^*},\mathsf{cert}_{i^*},\mathsf{cert}'_{i^*})$ such that $\mathsf{SEH.Verify}(h,\mathsf{rt},x_{i^*},i^*,\mathsf{cert}_{i^*})=1$ and $\mathsf{SEH.Verify}(h',\mathsf{rt}',x_{i^*},i^*,\mathsf{cert}'_{i^*})=1$ except with negligible probability. Next, the somewhere extractability of SEH guarantees that such $w_{i^*}=(x_{i^*},\mathsf{cert}_{i^*},\mathsf{cert}'_{i^*})$ satisfies $\widetilde{x}_{i^*}=x_{i^*}\wedge\widetilde{x}'_{i^*}=x_{i^*}$ except with negligible probability. From these two, it follows that when $\mathsf{BARG.V}_{\mathrm{idx}}(\mathsf{crs}^{\mathrm{idx}},C,\pi^{\mathrm{idx}})=1$, we have $\widetilde{x}_{i^*}=\widetilde{x}'_{i^*}$ except with negligible probability. Thus, there exists a negligible function negl' such that for infinitely many λ, N, x, I, z, and i^* as above,

$$\Pr\left[\begin{array}{l|l} \mathsf{V}^{\widehat{x}}_{\mathrm{hyb}}(\mathsf{crs},(h,\mathsf{rt}),\pi,\mathsf{td},\mathsf{td}')=1 & (\mathsf{crs},\mathsf{td}')\leftarrow\mathsf{Gen}_{\mathrm{hyb}}(1^\lambda,N,1^{|I|}) \\ \wedge\, x(i^*)\neq\widetilde{x}_{i^*} & (h,\mathsf{td})\leftarrow\mathsf{SEH.TGen}(1^\lambda,N,I) \\ & (\mathsf{rt},\pi)\leftarrow\mathsf{P}^*(\mathsf{crs},(x,h),z) \\ & \widehat{x}:=\mathsf{Encode}(\lambda,x) \\ & \{\widetilde{x}_i\}_{i\in I}:=\mathsf{SEH.Extract}(\mathsf{td},\mathsf{rt}) \end{array}\right] \geq \frac{1}{|I|\cdot\mathsf{poly}_{\mathsf{P}^*}(\lambda)}-\mathsf{negl}'(\lambda)\ . \tag{8}$$

Now, let us derive a contradiction using Claim 2 and Claim 3. Since $\mathsf{V}_{\mathrm{hyb}}$ outputs 0 unless $\widetilde{x}_{i^*}=\widetilde{x}'_{i^*}$, when we have $\mathsf{V}^{\widehat{x}}_{\mathrm{hyb}}(\mathsf{crs},(h,\mathsf{rt}),\pi)=1\wedge x(i^*)\neq\widetilde{x}_{i^*}$ as in (8), we have $x(i^*)\neq\widetilde{x}'_{i^*}$, i.e., the statement for the subroutine scheme is false. Then, since $\mathsf{V}_{\mathrm{hyb}}$ outputs 1 only when the cheating prover P^* provides an accepting proof of the subroutine scheme as a part of its proof, we can naturally use P^* to obtain a successful cheating prover against the subroutine scheme. In particular, it follows from (8) that there exists a PPT algorithm $\mathsf{P}^*_{\mathrm{sub}}$ and a polynomial $\mathsf{poly}'_{\mathsf{P}^*}$ such that for infinitely many $\lambda\in\mathbb{N}$, there exist $N\leq\mathsf{poly}_N(\lambda)$, $(\mathbb{F},\mathbb{H},m_N):=$

$\underline{\mathsf{crs}\leftarrow\mathsf{Gen}_{\mathrm{hyb}}(1^\lambda,N,1^{M_I})}$:

1. Let $(\mathbb{F},\mathbb{H},m_N):=\mathsf{param}_{\mathrm{LDE}}(\lambda,N)$.
2. Sample $(h',\mathsf{td}')\leftarrow\mathsf{SEH.TGen}(1^\lambda,|\mathbb{H}|^{m_N},\{i^*\})$.
3. Sample $\mathsf{crs}^{\mathrm{idx}}\leftarrow\mathsf{BARG.TGen}_{\mathrm{idx}}(1^\lambda,1^{M_C},N,i^*)$.
4. Sample $\mathsf{crs}'\leftarrow\mathsf{Gen}_{\mathrm{sub}}(1^\lambda,(\mathbb{F},\mathbb{H},m_N),\alpha)$, where $\alpha:=\lfloor\lambda^{1/(\lfloor m_N/m_\lambda\rfloor+1)}\rfloor$.
5. Output $\mathsf{crs}\leftarrow(1^\lambda,N,h',\mathsf{crs}^{\mathrm{idx}},\mathsf{crs}')$ and td'.

$\underline{b:=\mathsf{V}^{\widehat{x}}_{\mathrm{hyb}}(\mathsf{crs},(h,\mathsf{rt}),\pi,\mathsf{td},\mathsf{td}')}$:

1. Parse crs as $(1^\lambda,N,h',\mathsf{crs}^{\mathrm{idx}},\mathsf{crs}')$ and π as $(\mathsf{rt}',\pi^{\mathrm{idx}},\pi')$.
2. Output 1 iff $\mathsf{BARG.V}_{\mathrm{idx}}(\mathsf{crs}^{\mathrm{idx}},C,\pi^{\mathrm{idx}})=1$, $\mathsf{V}^{\widehat{x}}_{\mathrm{sub}}(\mathsf{crs}',(h',\mathsf{rt}'),\pi')=1$, and $\widetilde{x}_{i^*}=\widetilde{x}'_{i^*}$, where $\{\widetilde{x}_i\}_{i\in I}:=\mathsf{SEH.Extract}(\mathsf{td},\mathsf{rt})$ and $\widetilde{x}'_{i^*}:=\mathsf{SEH.Extract}(\mathsf{td}',\mathsf{rt}')$.

Figure 5: Hybrid setup algorithm $\mathsf{Gen}_{\mathrm{hyb}}$ and verifier $\mathsf{V}_{\mathrm{hyb}}$.

$\mathsf{param}_{\mathrm{LDE}}(\lambda, N)$, $x : \mathbb{H}^m \to \mathbb{F}$, $i^* \in [|\mathbb{H}|^m]$, $\alpha := \lfloor \lambda^{1/(\lfloor m_N/m_\lambda \rfloor + 1)} \rfloor$, and $z \in \{0,1\}^*$,

$$\Pr\left[\begin{array}{l|l} \mathsf{V}^{\widehat{x}}_{\mathrm{sub}}(\mathsf{crs}, (h, \mathsf{rt}), \pi) = 1 \\ \wedge\, x(i^*) \neq \widetilde{x}_{i^*} \end{array} \middle| \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{Gen}_{\mathrm{sub}}(1^\lambda, (\mathbb{F}, \mathbb{H}, m_N), \alpha) \\ (h, \mathsf{td}) \leftarrow \mathsf{SEH.TGen}(1^\lambda, |\mathbb{H}|^{m_N}, \{i^*\}) \\ (\mathsf{rt}, \pi) \leftarrow \mathsf{P}^*_{\mathrm{sub}}(\mathsf{crs}, (x, h), z) \\ \widehat{x} := \mathsf{LDE}_{\mathbb{F},\mathbb{H},m_N}(x) \\ \widetilde{x}_{i^*} := \mathsf{SEH.Extract}(\mathsf{td}, \mathsf{rt}) \end{array}\right] \geq \frac{1}{\mathsf{poly}'_{\mathsf{P}^*}(\lambda)} \ .$$

By repeated applications of Claim 3, there exists a PPT algorithm P^+ such that for infinitely many such λ, N, $(\mathbb{F}, \mathbb{H}, m_N)$, α and for $m := m_N - \lfloor m_N/m_\lambda \rfloor \cdot m_\lambda$, there exist $x^+ : \mathbb{H}^m \to \mathbb{F}$, $i^+ \in [|\mathbb{H}|^m]$, and $z^+ \in \{0,1\}^*$ such that

$$\Pr\left[\begin{array}{l|l} \mathsf{V}^{\widehat{x}^+}_{\mathrm{sub}}(\mathsf{crs}, (h, \mathsf{rt}), \pi) = 1 \\ \wedge\, x^+(i^*) \neq \widetilde{x}_{i^+} \end{array} \middle| \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{Gen}_{\mathrm{sub}}(1^\lambda, (\mathbb{F}, \mathbb{H}, m), \alpha) \\ (h, \mathsf{td}) \leftarrow \mathsf{SEH.TGen}(1^\lambda, |\mathbb{H}|^m, \{i^+\}) \\ (\mathsf{rt}, \pi) \leftarrow \mathsf{P}^+(\mathsf{crs}, (x^+, h), z^+) \\ \widehat{x}^+ := \mathsf{LDE}_{\mathbb{F},\mathbb{H},m}(x^+) \\ \widetilde{x}_{i^+} := \mathsf{SEH.Extract}(\mathsf{td}, \mathsf{rt}) \end{array}\right] \geq \left(\frac{1}{2|\mathbb{F}|^{m_\lambda}}\right)^{\lfloor m_N/m_\lambda \rfloor} \cdot \frac{1}{\mathsf{poly}'_{\mathsf{P}^*}(\lambda)} \ . \tag{9}$$

Note that when $N = \mathsf{poly}(\lambda)$, we have $\lfloor m_N/m_\lambda \rfloor = O(1)$ since we have $m_N \leq O(\log_{|\mathbb{H}|} N)$ and $m_\lambda = \lceil \log_{|\mathbb{H}|} \lambda \rceil$. Thus $(2|\mathbb{F}|^{m_\lambda})^k = \mathsf{poly}(\lambda)$ for every $k \in \{1, \ldots, \lfloor m_N/m_\lambda \rfloor\}$. Then, since $(2|\mathbb{F}|^{m_\lambda})^{\lfloor m_N/m_\lambda \rfloor} = \mathsf{poly}(\lambda)$, Eq. (9) contradicts to Claim 2.

Proofs of Claim 2 and Claim 3. It remains to prove Claim 2 and Claim 3. At a high level, we proceeds as follows. Claim 2 holds trivially since $m = m_N - \lfloor m_N/m_\lambda \rfloor \cdot m_\lambda < m_\lambda$. Regarding Claim 3, we prove it following the idea given in the technical overview (Section 2.1). That is, using the security of SEH and $\mathsf{BARG}_{\mathrm{idx}}$ as above and also relying on the correlation intractability of CIH_α, we show that any successful cheating prover P^* against $\mathsf{SEH\text{-}Del}_{\mathrm{sub}}$ can be used to obtain a successful cheating prover P^+ against one of the recursive proofs. A subtlety is that the correlation intractability of CIH_α is non-adaptive in the sense that the relation ensemble in the security experiment needs to be fixed in advance (cf. Definition 1). Since P^* chooses the indices $\{\mathbf{c}_i\}_{i \in [\alpha]} \in \mathbb{F}^{m_\lambda}$ of the recursive proofs adaptively, we can rely on the correlation intractability of CIH_α only when we correctly guess the index on which the prover cheats. Consequently, we can only show that P^+ succeeds with probability that decreases by a multiplicative factor $1/|\mathbb{F}|^{m_\lambda}$ as shown in (4). The formal proofs are given below.

Proof of Claim 2. Since $m = m_N - \lfloor m_N/m_\lambda \rfloor \cdot m_\lambda < m_\lambda$, the condition $\mathsf{V}^{\widehat{x}}_{\mathrm{sub}}(\mathsf{crs}, (h, \mathsf{rt}), \pi) = 1$ implies $\mathsf{SEH.Hash}(h, x) = \mathsf{rt}$, which in turn implies $\exists \mathsf{cert}_{i^*}$ s.t. $\mathsf{SEH.Verify}(h, \mathsf{rt}, x(i^*), i^*, \mathsf{cert}_{i^*}) = 1$. Thus, the somewhere extractability of SEH implies (2). □

Proof of Claim 3. Fix any PPT algorithm P^* and a polynomial poly_N. For any λ, N, $(\mathbb{F}, \mathbb{H}, m_N)$, m, x, i^*, α, and z as in the statement of the claim, let $\delta(\lambda, N, (\mathbb{F}, \mathbb{H}, m_N), m, x, i^*, \alpha, z)$ be the LHS of (3), i.e.,

$$\delta := \Pr\left[\begin{array}{l|l} \mathsf{V}^{\widehat{x}}_{\mathrm{sub}}(\mathsf{crs}, (h, \mathsf{rt}), \pi) = 1 \\ \wedge\, x(i^*) \neq \widetilde{x}_{i^*} \end{array} \middle| \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{Gen}_{\mathrm{sub}}(1^\lambda, (\mathbb{F}, \mathbb{H}, m), \alpha) \\ (h, \mathsf{td}) \leftarrow \mathsf{SEH.TGen}(1^\lambda, |\mathbb{H}|^m, \{i^*\}) \\ (\mathsf{rt}, \pi) \leftarrow \mathsf{P}^*(\mathsf{crs}, (x, h), z) \\ \widehat{x} := \mathsf{LDE}_{\mathbb{F},\mathbb{H},m}(x) \\ \widetilde{x}_{i^*} := \mathsf{SEH.Extract}(\mathsf{td}, \mathsf{rt}) \end{array}\right] ,$$

where $\delta(\lambda, N, (\mathbb{F}, \mathbb{H}, m_N), m, x, i^*, \alpha, z)$ is written as δ for editorial simplicity.

First, for any λ, N, $(\mathbb{F}, \mathbb{H}, m_N)$, m, x, i^*, α, and z as in the statement of the claim, let $\mathsf{Gen}_{\mathrm{sub},1}$ and $\mathsf{V}_{\mathrm{sub},1}$ be the hybrid setup and verifier that are defined in Figure 5. (The differences from $\mathsf{Gen}_{\mathrm{sub}}, \mathsf{V}_{\mathrm{sub}}$ are highlighted by colored backgrounds.) Let δ_1 be defined similarly to δ based on $(\mathsf{Gen}_{\mathrm{sub},1}, \mathsf{V}_{\mathrm{sub},1})$, i.e.,

$(\mathsf{crs}, \mathsf{td}_{\mathrm{sub}}) \leftarrow \mathsf{Gen}_{\mathrm{sub},1}(1^\lambda, (\mathbb{F}, \mathbb{H}, m), 1^{M_I}, \alpha)$:

1. If $m < m_\lambda$, output $\mathsf{crs} := (1^\lambda, (\mathbb{F}, \mathbb{H}, m), \bot, \bot, \bot, \bot)$ and $\mathsf{td}_{\mathrm{sub}} := \bot$ and terminate. If $m \geq m_\lambda$, continue to the next step.
2. Sample $(h^+, \mathsf{td}^+) \leftarrow \mathsf{SEH.TGen}(1^\lambda, |\mathbb{H}|^{m-m_\lambda}, \{i^+\})$, where $i^+ \in [|\mathbb{H}|^{m-m_\lambda}]$ is defined by first viewing $i^* \in [|\mathbb{H}|^m]$ as $(\mathbf{u}^+, \mathbf{v}^+) \in \mathbb{H}^{m_\lambda} \times \mathbb{H}^{m-m_\lambda}$ by the lexicographical order and then setting $i^+ := \mathbf{v}^+$.
3. Sample $(\mathsf{crs}^{\mathrm{idx}}, \mathsf{td}^{\mathrm{idx}}) \leftarrow \mathsf{BARG.TGen}_{\mathrm{idx}}(1^\lambda, 1^{M_C}, |\mathbb{H}|^{m-m_\lambda}, i^+)$, where $M_C := \max(|C_{\mathbf{u}}|, |C_{\mathbf{c}_i}|)$.
4. Sample $h^{\mathrm{CIH}} \leftarrow \mathsf{CIH.Gen}_\alpha(1^\lambda)$.
5. Sample $\mathsf{crs}^+ \leftarrow \mathsf{Gen}_{\mathrm{sub}}(1^\lambda, (\mathbb{F}, \mathbb{H}, m - m_\lambda), \alpha)$.
6. Output $\mathsf{crs} := (1^\lambda, (\mathbb{F}, \mathbb{H}, m), \alpha, h^+, \mathsf{crs}^{\mathrm{idx}}, h^{\mathrm{CIH}}, \mathsf{crs}^+)$ and $\mathsf{td}_{\mathrm{sub}} := (\mathsf{td}^+, \mathsf{td}^{\mathrm{idx}})$.

$b := \mathsf{V}^{\widehat{x}}_{\mathrm{sub},1}(\mathsf{crs}, (h, \mathsf{rt}), \pi, \mathsf{td}_{\mathrm{sub}}, \mathsf{td})$:

1. Parse crs as $(1^\lambda, (\mathbb{F}, \mathbb{H}, m), \alpha, h^+, \mathsf{crs}^{\mathrm{idx}}, h^{\mathrm{CIH}}, \mathsf{crs}^+)$. If $m < m_\lambda$, output 1 iff $\mathsf{SEH.Hash}(h, \pi) = \mathsf{rt}$ and $\pi = \widehat{x}|_{\mathbb{H}^m}$. If $m \geq m_\lambda$, continue to the next step.
2. Parse π as $(\{\mathsf{rt}^+_{\mathbf{u}}\}_{\mathbf{u} \in \mathbb{F}^{m_\lambda}}, \{\pi^{\mathrm{idx}}_{\mathbf{u}}\}_{\mathbf{u} \in \mathbb{H}^{m_\lambda}}, \pi^{\mathrm{idx}}, \{\pi^+_{\mathbf{c}_i}\}_{\mathbf{c}_i \in S_{\mathbf{c}}})$ and parse $\mathsf{td}_{\mathrm{sub}}$ as $(\mathsf{td}^+, \mathsf{td}^{\mathrm{idx}})$.
3. Compute $(\mathbf{c}_1, \ldots, \mathbf{c}_\alpha) := \mathsf{CIH.Hash}_\alpha(h^{\mathrm{CIH}}, \{\mathsf{rt}^+_{\mathbf{u}}\}_{\mathbf{u} \in \mathbb{H}^{m_\lambda}})$, where each $\mathbf{c}_i$ is viewed as an element of $\mathbb{F}^{m_\lambda}$. Let $S_{\mathbf{c}} := \{\mathbf{c}_1, \ldots, \mathbf{c}_\alpha\}$.
4. Output 1 iff all of the following hold.
   - $\mathsf{BARG.V}_{\mathrm{idx}}(\mathsf{crs}^{\mathrm{idx}}, C_{\mathbf{u}}, \pi^{\mathrm{idx}}_{\mathbf{u}}) = 1$ for $\forall \mathbf{u} \in \mathbb{H}^{m_\lambda}$, where $C_{\mathbf{u}}$ is defined as in $\mathsf{P}_{\mathrm{sub}}$.
   - $\mathsf{BARG.V}_{\mathrm{idx}}(\mathsf{crs}^{\mathrm{idx}}, C, \pi^{\mathrm{idx}}) = 1$, where C is defined as in $\mathsf{P}_{\mathrm{sub}}$.
   - $\mathsf{V}^{\widehat{x}(\mathbf{c}_i, \cdot)}_{\mathrm{sub}}(\mathsf{crs}^+, (h^+, \mathsf{rt}^+_{\mathbf{c}_i}), \pi^+_{\mathbf{c}_i}) = 1$ for $\forall \mathbf{c}_i \in S_{\mathbf{c}}$.
   - $\widetilde{x}_{i^*} = \tilde{x}^+_{(\mathbf{u}^+, \mathbf{v}^+)}$, where $\widetilde{x}_{i^*} := \mathsf{SEH.Extract}(\mathsf{td}, \mathsf{rt})$, and $\tilde{x}^+_{(\mathbf{u}^+, \mathbf{v}^+)} := \mathsf{SEH.Extract}(\mathsf{td}^+, \mathsf{rt}^+_{\mathbf{u}^+})$.
   - $\{\tilde{x}^+_{(\mathbf{u}, \mathbf{v}^+)}\}_{\mathbf{u} \in \mathbb{F}^{m_\lambda}}$ is the LDE of $\{\tilde{x}^+_{(\mathbf{u}, \mathbf{v}^+)}\}_{\mathbf{u} \in \mathbb{H}^{m_\lambda}}$ w.r.t. $(\mathbb{F}, \mathbb{H}, m_\lambda)$, where $\tilde{x}^+_{(\mathbf{u}, \mathbf{v}^+)} := \mathsf{SEH.Extract}(\mathsf{td}^+, \mathsf{rt}^+_{\mathbf{u}})$.

Figure 6: A hybrid setup algorithm $\mathsf{Gen}_{\mathrm{sub},1}$ and verifier $\mathsf{V}_{\mathrm{sub},1}$.

$$\delta_1 := \Pr\left[\begin{array}{l} \mathsf{V}^{\widehat{x}}_{\mathrm{sub},1}(\mathsf{crs}, (h, \mathsf{rt}), \pi) = 1 \\ \wedge\, x(i^*) \neq \widetilde{x}_{i^*} \end{array} \middle| \begin{array}{l} (\mathsf{crs}, \mathsf{td}_{\mathrm{sub}}) \leftarrow \mathsf{Gen}_{\mathrm{sub},1}(1^\lambda, (\mathbb{F}, \mathbb{H}, m), \alpha) \\ (h, \mathsf{td}) \leftarrow \mathsf{SEH.TGen}(1^\lambda, |\mathbb{H}|^m, \{i^*\}) \\ (\mathsf{rt}, \pi) \leftarrow \mathsf{P}^*(\mathsf{crs}, (x, h), z) \\ \widehat{x} := \mathsf{LDE}_{\mathbb{F},\mathbb{H},m}(x) \\ \widetilde{x}_{i^*} := \mathsf{SEH.Extract}(\mathsf{td}, \mathsf{rt}) \end{array}\right].$$

At a high level, $(\mathsf{Gen}_{\mathrm{sub},1}, \mathsf{V}_{\mathrm{sub},1})$ differs from $(\mathsf{Gen}_{\mathrm{sub}}, \mathsf{V}_{\mathrm{sub}})$ in that (i) $\mathsf{Gen}_{\mathrm{sub},1}$ uses the trapdoor setup algorithms of SEH and $\mathsf{BARG}_{\mathrm{idx}}$, and (ii) $\mathsf{V}_{\mathrm{sub},1}$ outputs 0 if the $\mathsf{BARG}_{\mathrm{idx}}$ proofs that P^* provides are accepting but the values that are extracted P^* do not satisfy the statements of the $\mathsf{BARG}_{\mathrm{idx}}$ proofs. Thus, by using the security of SEH and $\mathsf{BARG}_{\mathrm{idx}}$ as in the proof of the soundness of SEH-Del above, we can show that there exists a negligible function negl_1 such that for every λ, N, $(\mathbb{F}, \mathbb{H}, m_N)$, m, x, i^*, α, and z as in the statement of the claim, it holds $|\delta - \delta_1| \leq \mathsf{negl}_1(\lambda)$.

Next, for any λ, N, $(\mathbb{F}, \mathbb{H}, m_N)$, m, x, i^*, α, and z as in the statement of the claim, let $\mathsf{Gen}_{\mathrm{sub},2}$ be identical with $\mathsf{Gen}_{\mathrm{sub},1}$ and $\mathsf{V}_{\mathrm{sub},2}$ be the hybrid verifier that is defined in Figure 7. Let δ_2 be defined similarly to δ based on $(\mathsf{Gen}_{\mathrm{sub},2}, \mathsf{V}_{\mathrm{sub},2})$, i.e.,

$$\delta_2 := \Pr\left[\begin{array}{l} \mathsf{V}^{\widehat{x}}_{\mathrm{sub},2}(\mathsf{crs}, (h, \mathsf{rt}), \pi, \mathsf{td}_{\mathrm{sub}}, \mathsf{td}) = 1 \\ \wedge\, x(i^*) \neq \widetilde{x}_{i^*} \end{array} \middle| \begin{array}{l} (\mathsf{crs}, \mathsf{td}_{\mathrm{sub}}) \leftarrow \mathsf{Gen}_{\mathrm{sub},2}(1^\lambda, (\mathbb{F}, \mathbb{H}, m), \alpha) \\ (h, \mathsf{td}) \leftarrow \mathsf{SEH.TGen}(1^\lambda, |\mathbb{H}|^m, \{i^*\}) \\ (\mathsf{rt}, \pi) \leftarrow \mathsf{P}^*(\mathsf{crs}, (x, h), z) \\ \widehat{x} := \mathsf{LDE}_{\mathbb{F},\mathbb{H},m}(x) \\ \widetilde{x}_{i^*} := \mathsf{SEH.Extract}(\mathsf{td}, \mathsf{rt}) \end{array}\right]. \quad (10)$$

We show that there exists a negligible function negl_2 such that for every λ, N, $(\mathbb{F}, \mathbb{H}, m_N)$, m, x, i^*, α, and z as in the statement of the claim, it holds $|\delta_1 - \delta_2| \leq \mathsf{negl}_2(\lambda)$. Toward this end, it suffices to show that when

$b \coloneqq \mathsf{V}^{\widehat{x}}_{\mathrm{sub},2}(\mathsf{crs}, (h, \mathsf{rt}), \pi, \mathsf{td}_{\mathrm{sub}}, \mathsf{td})$:

1. Parse crs as $(1^\lambda, (\mathbb{F}, \mathbb{H}, m), \alpha, h^+, \mathsf{crs}^{\mathrm{idx}}, h^{\mathsf{CIH}}, \mathsf{crs}^+)$. If $m < m_\lambda$, output 1 iff $\mathsf{SEH.Hash}(h, \pi) = \mathsf{rt}$ and $\pi = \widehat{x}|_{\mathbb{H}^m}$. If $m \geq m_\lambda$, continue to the next step.
2. Parse π as $(\{\mathsf{rt}^+_{\mathbf{u}}\}_{\mathbf{u}\in\mathbb{F}^{m_\lambda}}, \{\pi^{\mathrm{idx}}_{\mathbf{u}}\}_{\mathbf{u}\in\mathbb{H}^{m_\lambda}}, \pi^{\mathrm{idx}}, \{\pi^+_{\mathbf{c}_i}\}_{\mathbf{c}_i\in S_{\mathbf{c}}})$ and parse $\mathsf{td}_{\mathrm{sub}}$ as $(\mathsf{td}^+, \mathsf{td}^{\mathrm{idx}})$.
3. Compute $(\mathbf{c}_1, \ldots, \mathbf{c}_\alpha) \coloneqq \mathsf{CIH.Hash}_\alpha(h^{\mathsf{CIH}}, \{\mathsf{rt}^+_{\mathbf{u}}\}_{\mathbf{u}\in\mathbb{H}^{m_\lambda}})$, where each $\mathbf{c}_i$ is viewed as an element of $\mathbb{F}^{m_\lambda}$. Let $S_{\mathbf{c}} \coloneqq \{\mathbf{c}_1, \ldots, \mathbf{c}_\alpha\}$.
4. Output 1 iff all of the following hold.
   - $\mathsf{BARG.V}_{\mathrm{idx}}(\mathsf{crs}^{\mathrm{idx}}, C_{\mathbf{u}}, \pi^{\mathrm{idx}}_{\mathbf{u}}) = 1$ for $\forall \mathbf{u} \in \mathbb{H}^{m_\lambda}$, where $C_{\mathbf{u}}$ is defined as in $\mathsf{P}_{\mathrm{sub}}$.
   - $\mathsf{BARG.V}_{\mathrm{idx}}(\mathsf{crs}^{\mathrm{idx}}, C, \pi^{\mathrm{idx}}) = 1$, where C is defined as in $\mathsf{P}_{\mathrm{sub}}$.
   - $\mathsf{V}^{\widehat{x}(\mathbf{c}_i,\cdot)}_{\mathrm{sub}}(\mathsf{crs}^+, (h^+, \mathsf{rt}^+_{\mathbf{c}_i}), \pi^+_{\mathbf{c}_i}) = 1$ for $\forall \mathbf{c}_i \in S_{\mathbf{c}}$.
   - $\widetilde{x}_{i^*} = \tilde{x}^+_{(\mathbf{u}^+,\mathbf{v}^+)}$, where $\widetilde{x}_{i^*} \coloneqq \mathsf{SEH.Extract}(\mathsf{td}, \mathsf{rt})$ and $\tilde{x}^+_{(\mathbf{u}^+,\mathbf{v}^+)} \coloneqq \mathsf{SEH.Extract}(\mathsf{td}^+, \mathsf{rt}^+_{\mathbf{u}^+})$.
   - $\{\tilde{x}^+_{(\mathbf{u},\mathbf{v}^+)}\}_{\mathbf{u}\in\mathbb{F}^{m_\lambda}}$ is the LDE of $\{\tilde{x}^+_{(\mathbf{u},\mathbf{v}^+)}\}_{\mathbf{u}\in\mathbb{H}^{m_\lambda}}$ w.r.t. $(\mathbb{F}, \mathbb{H}, m_\lambda)$, where $\tilde{x}^+_{(\mathbf{u},\mathbf{v}^+)} \coloneqq \mathsf{SEH.Extract}(\mathsf{td}^+, \mathsf{rt}^+_{\mathbf{u}})$.
   - $\exists \mathbf{c}_i \in S_{\mathbf{c}}$ s.t. $\tilde{x}^+_{(\mathbf{c}_i,\mathbf{v}^+)} \neq \widehat{x}(\mathbf{c}_i, \mathbf{v}^+)$.

Figure 7: A hybrid verifier $\mathsf{V}_{\mathrm{sub},2}$.

$x(i^*) \neq \widetilde{x}_{i^*}$, the probability that $\mathsf{V}_{\mathrm{sub},1}$ outputs 1 while $\mathsf{V}_{\mathrm{sub},2}$ outputs 0 is negligible. Thus, it suffices to show that the probability that all of the following occur simultaneously in Step 4 of $\mathsf{V}_{\mathrm{sub},2}$ is negligible.

- $x(i^*) \neq \widetilde{x}_{i^*}$.
- $\widetilde{x}_{i^*} = \tilde{x}^+_{(\mathbf{u}^+,\mathbf{v}^+)}$.
- $\{\tilde{x}^+_{(\mathbf{u},\mathbf{v}^+)}\}_{\mathbf{u}\in\mathbb{F}^{m_\lambda}}$ is the LDE of $\{\tilde{x}^+_{(\mathbf{u},\mathbf{v}^+)}\}_{\mathbf{u}\in\mathbb{H}^{m_\lambda}}$ w.r.t. $(\mathbb{F}, \mathbb{H}, m_\lambda)$,
- $\tilde{x}^+_{(\mathbf{c}_i,\mathbf{v}^+)} = \widehat{x}(\mathbf{c}_i, \mathbf{v}^+)$ for $\forall \mathbf{c}_i \in S_{\mathbf{c}}$.

(Indeed, unless the last three conditions hold, the outputs of $\mathsf{V}_{\mathrm{sub},1}$ and $\mathsf{V}_{\mathrm{sub},2}$ are identical.) Observe that when all of the above conditions hold, $\{x(\mathbf{u}, \mathbf{v}^+)\}_{\mathbf{u}\in\mathbb{H}^{m_\lambda}}$ and $\{\tilde{x}^+_{(\mathbf{u},\mathbf{v}^+)}\}_{\mathbf{u}\in\mathbb{H}^{m_\lambda}}$ disagree on $\mathbf{u}^+$ but their LDEs (w.r.t. $(\mathbb{F}, \mathbb{H}, m_\lambda)$) agree on every $\mathbf{c}_i \in S_{\mathbf{c}}$.[16] Based on this observation, we use the correlation intractability of CIH to evaluate the probability that the above conditions hold. Let us first define a product-relation $\mathbf{R}_\lambda \subseteq X_\lambda \times Y^\alpha_\lambda$.

- Let C_{CIH} be the following circuit. It has $(\mathbb{F}, \mathbb{H}, m)$, td^+, $\mathbf{u}^+$, and $\{x(\mathbf{u}, \mathbf{v}^+)\}_{\mathbf{u}\in\mathbb{H}^{m_\lambda}}$ as hardwired inputs, and takes $\{\mathsf{rt}^+_{\mathbf{u}}\}_{\mathbf{u}\in\mathbb{H}^{m_\lambda}}$ and $\mathbf{c}_i \in \mathbb{F}^{m_\lambda}$ as inputs. Then, it outputs 1 iff $\{x(\mathbf{u}, \mathbf{v}^+)\}_{\mathbf{u}\in\mathbb{H}^{m_\lambda}}$ and $\{\tilde{x}^+_{(\mathbf{u},\mathbf{v}^+)}\}_{\mathbf{u}\in\mathbb{H}^{m_\lambda}}$ disagree on $\mathbf{u}^+$ but their LDEs (w.r.t. $(\mathbb{F}, \mathbb{H}, m_\lambda)$) agree on $\mathbf{c}_i$, where $\tilde{x}^+_{(\mathbf{u},\mathbf{v}^+)} \coloneqq \mathsf{SEH.Extract}(\mathsf{td}^+, \mathsf{rt}^+_{\mathbf{u}})$ as before.
- Now, $\mathbf{R}_\lambda$ is defined as follows. For each $x \in X_\lambda$, we have $(x, (y_1, \ldots, y_\alpha)) \in \mathbf{R}_\lambda$ iff $C_{\mathsf{CIH}}(x, y_i) = 1$ for $\forall i \in [\alpha]$.

Note that $\mathbf{R}_\lambda$ has product sparsity at most $m_\lambda|\mathbb{H}|/|\mathbb{F}| \leq O(1)$, and it is T_{CIH}-time product verifiable for $T_{\mathsf{CIH}}(\lambda) = \mathsf{poly}(\lambda, m_\lambda, |\mathbb{H}|^{m_\lambda}, \log|\mathbb{H}|^{m-m_\lambda}) \leq \mathsf{poly}(\lambda)$. Thus, when the requirements about CIH_α are appropriately specified (i.e., the polynomials $\mathsf{poly}_X, \mathsf{poly}_Y, \mathsf{poly}_T$ and the constant ρ in the definition of CIH_α are sufficiently large), CIH_α is correlation intractable for the relation ensemble $\{\mathbf{R}_\lambda\}_{\lambda\in\mathbb{N}}$. Thus, it follows from the correlation intractability of CIH_α that when $\{x(\mathbf{u}, \mathbf{v}^+)\}_{\mathbf{u}\in\mathbb{H}^{m_\lambda}}$ and $\{\tilde{x}^+_{(\mathbf{u},\mathbf{v}^+)}\}_{\mathbf{u}\in\mathbb{H}^{m_\lambda}}$ disagree on $\mathbf{u}^+$, their LDEs disagree on some $\mathbf{c}_i \in S_{\mathbf{c}}$ except with negligible probability. Thus, we obtain $|\delta_1 - \delta_2| \leq \mathsf{negl}_2(\lambda)$ as desired.

[16] The disagreement follows from the first two conditions since we have $x(i^*) = x(\mathbf{u}^+, \mathbf{v}^+)$ by the definition of $(\mathbf{u}^+, \mathbf{v}^+)$. The agreement follows from the last two conditions.

$(\mathsf{rt}^+, \pi^+) \leftarrow \mathsf{P}^+(\mathsf{crs}^+, (h^+, x^+), z^+)$:

1. Parse z^+ as $(\lambda, N, (\mathbb{F}, \mathbb{H}, m_N), m, x, i^*, \alpha, z, \mathbf{c}^*)$.
2. Define crs by emulating $\mathsf{Gen}_{\mathrm{sub},2}$ as follows using crs^+ and h^+.
   (a) Sample $(\mathsf{crs}^{\mathrm{idx}}, \mathsf{td}^{\mathrm{idx}}) \leftarrow \mathsf{BARG.TGen}_{\mathrm{idx}}(1^\lambda, 1^{M_C}, |\mathbb{H}|^{m-m_\lambda}, \{i^+\})$, where $M_C := \max(|C_{\mathbf{u}}|, |C|)$ and i^+ are defined as in $\mathsf{Gen}_{\mathrm{sub},2}$.
   (b) Sample $h^{\mathrm{CIH}} \leftarrow \mathsf{CIH.Gen}_\alpha(1^\lambda)$.
   (c) Let $\mathsf{crs} := (1^\lambda, (\mathbb{F}, \mathbb{H}, m), \alpha, h^+, \mathsf{crs}^{\mathrm{idx}}, h^{\mathrm{CIH}}, \mathsf{crs}^+)$.
3. Sample $(h, \mathsf{td}) \leftarrow \mathsf{SEH.TGen}(1^\lambda, |\mathbb{H}|^m, \{i^*\})$.
4. Run $(\mathsf{rt}, \pi) \leftarrow \mathsf{P}^*(\mathsf{crs}, (x, h), z)$.
5. Parse π as $(\{\mathsf{rt}^+_{\mathbf{u}}\}_{\mathbf{u}\in\mathbb{F}^{m_\lambda}}, \{\pi^{\mathrm{idx}}_{\mathbf{u}}\}_{\mathbf{u}\in\mathbb{H}^{m_\lambda}}, \pi^{\mathrm{idx}}, \{\pi^+_{\mathbf{c}_i}\}_{\mathbf{c}_i\in S_{\mathbf{c}}})$.
6. Compute $(\mathbf{c}_1, \ldots, \mathbf{c}_\alpha) := \mathsf{CIH.Hash}_\alpha(h^{\mathrm{CIH}}, \{\mathsf{rt}^+_{\mathbf{u}}\}_{\mathbf{u}\in\mathbb{H}^{m_\lambda}})$, where each $\mathbf{c}_i$ is viewed as an element of $\mathbb{F}^{m_\lambda}$. Let $S_{\mathbf{c}} := \{\mathbf{c}_1, \ldots, \mathbf{c}_\alpha\}$.
7. If $\exists \mathbf{c}_i \in S_{\mathbf{c}}$ s.t. $\mathbf{c}_i = \mathbf{c}^*$, output $\mathsf{rt}^+ := \mathsf{rt}^+_{\mathbf{c}_i}$ and $\pi^+ := \pi^+_{\mathbf{c}_i}$. Otherwise, abort.

Figure 8: Cheating prover P^+.

Let us combine the analyses about $(\mathsf{Gen}_{\mathrm{sub},i}, \mathsf{V}_{\mathrm{sub},i})$ for $i \in [2]$. Since we have $\delta_2 \geq \delta - \sum_{1\leq i\leq 2} \mathsf{negl}_i(\lambda)$, it follows that in the probabilistic experiment of (10), we have the following in Step 4 of $\mathsf{V}_{\mathrm{sub},2}$.

$$\Pr\left[\begin{array}{l}\exists \mathbf{c}_i \in S_{\mathbf{c}} \text{ s.t.} \\ \mathsf{V}_{\mathrm{sub}}^{\widehat{x}(\mathbf{c}_i,\cdot)}(\mathsf{crs}^+, (h^+, \mathsf{rt}^+_{\mathbf{c}_i}), \pi^+_{\mathbf{c}_i}) = 1 \wedge \tilde{x}^+_{(\mathbf{c}_i,\mathbf{v}^+)} \neq \widehat{x}(\mathbf{c}_i, \mathbf{v}^+)\end{array}\right] \geq \delta - \sum_{1\leq i\leq 2} \mathsf{negl}_i(\lambda) \ . \tag{11}$$

Eq. (11) suggests that the cheating prover P^* can be used to break the soundness of one of the recursive proofs, and we indeed do so below.

We are ready to present the cheating prover P^+ for (4). For any λ, N, $(\mathbb{F}, \mathbb{H}, m_N)$, m, x, i^*, α, and z as in the statement of the claim and any $\mathbf{c}^* \in \mathbb{F}^{m_\lambda}$, let $z^+ := (\lambda, N, (\mathbb{F}, \mathbb{H}, m_N), m, x, i^*, \alpha, z, \mathbf{c}^*)$. Then, P^+ is defined in Figure 8 using P^* as a subroutine. (Essentially, P^+ emulates the experiment of (11) while hoping that the event considered in (11) occurs for $\mathbf{c}_i = \mathbf{c}^*$.) Let us analyze the success probability of P^+. First, we consider running P^+ for random $\mathbf{c}^*$ in the experiment of (4), and for any λ, N, $(\mathbb{F}, \mathbb{H}, m_N)$, m, x, i^*, α, and z such that we have (3), we obtain

$$\Pr\left[\begin{array}{l}\mathsf{V}_{\mathrm{sub}}^{\widehat{x}^+}(\mathsf{crs}^+, (h^+, \mathsf{rt}^+), \pi^+) = 1 \\ \wedge\ \tilde{x}^+_{i^+} \neq x^+(i^+)\end{array}\ \middle|\ \begin{array}{l}\mathbf{c}^* \leftarrow \mathbb{F}^{m_\lambda} \\ \text{Let } x^+ : \mathbb{H}^{m-m_\lambda} \to \mathbb{F} \text{ be defined as } x^+ : \mathbf{v} \mapsto \widehat{x}(\mathbf{c}^*, \mathbf{v}) \\ \text{Let } i^+ \text{ be defined based on } i^* \text{ as in } \mathsf{Gen}_{\mathrm{sub},2} \\ z^+ := (\lambda, N, (\mathbb{F}, \mathbb{H}, m_N), m, x, i^*, \alpha, z, \mathbf{c}^*) \\ \mathsf{crs}^+ \leftarrow \mathsf{Gen}_{\mathrm{sub}}(1^\lambda, (\mathbb{F}, \mathbb{H}, m - m_\lambda), \alpha) \\ (h^+, \mathsf{td}^+) \leftarrow \mathsf{SEH.TGen}(1^\lambda, |\mathbb{H}|^{m-m_\lambda}, \{i^+\}) \\ (\mathsf{rt}^+, \pi^+) \leftarrow \mathsf{P}^+(\mathsf{crs}^+, (h^+, x^+), z^+) \\ \widehat{x}^+ := \mathsf{LDE}_{\mathbb{F},\mathbb{H},m}(x^+) \\ \tilde{x}^+_{i^+} := \mathsf{SEH.Extract}(\mathsf{td}^+, \mathsf{rt}^+)\end{array}\right] \geq \frac{1}{|\mathbb{F}|^{m_\lambda}} \left(\frac{1}{\mathsf{poly}_{\mathsf{P}^*}(\lambda)} - \sum_{1\leq i\leq 2} \mathsf{negl}_i(\lambda)\right) \ . \tag{12}$$

(Eq. (12) follows from (3) and (11) since in the experiment of (12), P^+ perfectly emulates the experiment of (11) for P^*.[17]) Thus, from the average argument, there exists $\mathbf{c}^* \in \mathbb{F}^{m_\lambda}$ such that by setting x^+, i^+, and z^+ as

[17] Concretely, suppose that when P^+ internally invokes P^* in the experiment of (12), the output of P^* satisfies $\exists \mathbf{c}_i \in S_{\mathbf{c}}$ s.t. $\mathsf{V}_{\mathrm{sub}}^{\widehat{x}(\mathbf{c}_i,\cdot)}(\mathsf{crs}^+, (h^+, \mathsf{rt}^+_{\mathbf{c}_i}), \pi^+_{\mathbf{c}_i}) = 1 \wedge \tilde{x}^+_{(\mathbf{c}_i,\mathbf{v}^+)} \neq \widehat{x}(\mathbf{c}_i, \mathbf{v}^+)$. (Eq. (11) guarantees that this event occurs with probability at least

in (12), we have

$$\Pr\left[\begin{array}{l|l} \mathsf{V}_{\mathrm{sub}}^{\widehat{x}^+}(\mathsf{crs}^+,(h^+,\mathsf{rt}^+),\pi^+)=1 \\ \wedge\ \tilde{x}^+_{i^+}\neq x^+(i^+) \end{array}\ \middle|\ \begin{array}{l} \mathsf{crs}^+\leftarrow\mathsf{Gen}_{\mathrm{sub}}(1^\lambda,(\mathbb{F},\mathbb{H},m-m_\lambda),\alpha) \\ (h^+,\mathsf{td}^+)\leftarrow\mathsf{SEH.TGen}(1^\lambda,|\mathbb{H}|^{m-m_\lambda},\{i^+\}) \\ (\mathsf{rt}^+,\pi^+)\leftarrow\mathsf{P}^+(\mathsf{crs}^+,(h^+,x^+),z^+) \\ \widehat{x}^+:=\mathsf{LDE}_{\mathbb{F},\mathbb{H},m}(x^+) \\ \tilde{x}^+_{i^+}:=\mathsf{SEH.Extract}(\mathsf{td}^+,\mathsf{rt}^+) \end{array}\right]$$
$$\geq\frac{1}{|\mathbb{F}|^{m_\lambda}}\left(\frac{1}{\mathsf{poly}_{\mathsf{P}^*}(\lambda)}-\sum_{1\leq i\leq 2}\mathsf{negl}_i(\lambda)\right).$$

Thus, we have (4) as desired. This completes the proof of Claim 3. □

This completes the proof of the soundness of SEH-Del. Therefore, this completes the proof of Lemma 2. □

5 Holographic SNARG for Tree-Hash

In this section, we construct a holographic SNARG for the correctness of Merkle tree-hash computations. The target of this section is defined as follows.

Definition 20 (Tree-hash delegation)**.** ***Publicly verifiable non-interactive delegation schemes for tree-hash computations*** *(or publicly verifiable non-interactive tree-hash delegation schemes in short) are a special case of publicly verifiable non-interactive delegation schemes for Turing machines (Definition 8), where the following restrictions are imposed.*

- *The instance χ is restricted to the form $\chi=(x,(h,\mathsf{rt}))$, where $x\in\{0,1\}^{2^\ell\lambda}$ ($\ell\in\mathbb{N}$) is a binary string, $h:\{0,1\}^{2\lambda}\to\{0,1\}^\lambda$ is a hash function (represented as a circuit), and $\mathsf{rt}\in\{0,1\}^\lambda$ is a binary string.*
- *The Turing machine M is fixed to be the two-input Turing machine $M_{\text{tree-hash}}$ that takes an input of the form $\chi=(x,(h,\mathsf{rt}))$ and outputs 1 iff $\mathsf{TreeHash}_h(x)=\mathsf{rt}$. Also, the time bound T (given to Gen along with the input length bounds n_1,n_2) is fixed to be $T_{\text{tree-hash}}(n_1,n_2)$, where $T_{\text{tree-hash}}$ is a polynomial upper bound of the running time of $M_{\text{tree-hash}}$.*

The goal of this section is to prove the following lemma.

Lemma 3. *Assume the existence of the following primitives.*

- *A somewhere extractable hash function family* SEH.
- *A semi-adaptive somewhere-sound publicly verifiable non-interactive BARG for the index language.*
- *A partially adaptive somewhere-sound holographic delegation scheme* SEH-Del *for* SEH.

Then, there exists a partially adaptive publicly verifiable non-interactive tree-hash delegation scheme that satisfies the following properties.

- *The scheme is holographic w.r.t. the same encoding algorithm as* SEH-Del.
- *The setup time $T_{\mathsf{Gen}}(\lambda,T,n_1,n_2)$ and the proof length $L_\pi(\lambda,T,n_1,n_2)$ are both at most $\mathsf{poly}(\lambda,\log n_1,n_2)$.*
- *The scheme is public-coin if the above-listed primitives are public-coin.*

$1/\mathsf{poly}_{\mathsf{P}^*}(\lambda)-\sum_{1\leq i\leq 2}\mathsf{negl}_i(\lambda)$.) Then, with probability $1/|\mathbb{F}|^{m_\lambda}$, we have $\mathbf{c}_i=\mathbf{c}^*$, and thus, P^+ outputs $\mathsf{rt}^+:=\mathsf{rt}^+_{\mathbf{c}_i}$ and $\pi^+:=\pi^+_{\mathbf{c}_i}$. When P^+ outputs such rt^+ and π^+, we have $\mathsf{V}_{\mathrm{sub}}^{\widehat{x}^+}(\mathsf{crs}^+,(h^+,\mathsf{rt}^+),\pi^+)=1$ and $\tilde{x}^+_{i^+}\neq x^+(i^+)$. Indeed, the former follows from $\mathsf{V}_{\mathrm{sub}}^{\widehat{x}(\mathbf{c}_i,\cdot)}(\mathsf{crs}^+,(h^+,\mathsf{rt}^+_{\mathbf{c}_i}),\pi^+_{\mathbf{c}_i})=1$ since $\widehat{x}(\mathbf{c}_i,\cdot)$ is the LDE of $x^+=\widehat{x}(\mathbf{c}_i,\cdot)|_{\mathbb{H}^{m-m_\lambda}}$. The latter follows from $\tilde{x}^+_{(\mathbf{c}_i,\mathbf{v}^+)}\neq\widehat{x}(\mathbf{c}_i,\mathbf{v}^+)$ since (i) $\tilde{x}^+_{i^+}=\mathsf{SEH.Extract}(\mathsf{td}^+,\mathsf{rt}^+)=\mathsf{SEH.Extract}(\mathsf{td}^+,\mathsf{rt}^+_{\mathbf{c}_i})=\tilde{x}^+_{(\mathbf{c}_i,\mathbf{v}^+)}$ and (ii) $x^+(i^+)=x^+(\mathbf{v}^+)=\widehat{x}(\mathbf{c}_i,\mathbf{v}^+)$ because of the definition of i^+.

Since the primitives listed in Lemma 3 exist under the LWE assumption (Theorem 2, Theorem 3, Lemma 2), Lemma 3 implies the following corollary.

Corollary 1. *Under the LWE assumption, there exists a partially adaptive public-coin non-interactive tree-hash delegation scheme that satisfies the following properties.*

- *The scheme is LDE-holographic, where for the security parameter λ and an input x of length N, the encoding algorithm* Encode *outputs the LDE of x w.r.t. an arbitrary LDE parameter $(\mathbb{F}, \mathbb{H}, m)$ such that $|\mathbb{F}| \leq \mathsf{poly}(|\mathbb{H}|) \leq \mathsf{poly}(\log N)$, $|\mathbb{H}|^m \leq \mathsf{poly}(N)$, and $m|\mathbb{H}|/|\mathbb{F}| \leq O(1)$.*
- *The setup time $T_{\mathsf{Gen}}(\lambda, T, n_1, n_2)$ and the proof length $L_\pi(\lambda, T, n_1, n_2)$ are both at most $\mathsf{poly}(\lambda, \log n_1, n_2)$.*

In the rest of this section, we prove Lemma 3.

Proof of Lemma 3. Let $\mathsf{SEH} = (\mathsf{SEH.Gen}, \mathsf{SEH.TGen}, \mathsf{SEH.Hash}, \mathsf{SEH.Open}, \mathsf{SEH.Verify}, \mathsf{SEH.Extract})$ be a somewhere extractable hash function family, $\mathsf{BARG}_{\mathrm{idx}} = (\mathsf{BARG.Gen}_{\mathrm{idx}}, \mathsf{BARG.P}_{\mathrm{idx}}, \mathsf{BARG.V}_{\mathrm{idx}})$ be a semi-adaptive somewhere-sound publicly verifiable non-interactive BARG for the index language, and $\mathsf{SEH\text{-}Del} = (\mathsf{SEH\text{-}Del.Gen}, \mathsf{SEH\text{-}Del.P}, \mathsf{SEH\text{-}Del.V}, \mathsf{SEH\text{-}Del.Encode})$ be a partially adaptive somewhere-sound holographic delegation scheme for SEH. For simplicity, we adjust the definition of somewhere extractable hash functions (Definition 5) so that the set of the indices is $\{0, \ldots, N-1\}$ rather than $\{1, \ldots, N\}$. The same adjustment is also made to the definition of BARGs (Definition 12).

Our scheme $\mathsf{TH\text{-}Del} = (\mathsf{Gen}, \mathsf{P}, \mathsf{V})$ is given in Figure 9. (We remind that, as stated in Definition 20, the length of the input x is fixed to be $2^\ell\lambda$ for $\ell \in \mathbb{N}$.) We prove the security of $\mathsf{TH\text{-}Del}$ following the idea given in the technical overview (Section 2.1). That is, we prove the soundness by considering the tree nodes that we can extract from the prover. The formal proofs are given below.

5.1 Completeness

The completeness can be verified by inspection. Also, it is easy to see that $(\mathsf{Gen}, \mathsf{P}, \mathsf{V})$ is holographic w.r.t. the same encoding algorithm as $\mathsf{SEH\text{-}Del}$. (In particular, the verifier V only uses x to execute $\mathsf{SEH\text{-}Del.V}^{\widehat{x}}$ for $\widehat{x} \coloneqq \mathsf{SEH\text{-}Del.Encode}(\lambda, x)$ in the last step.)

5.2 Efficiency

The efficiency condition can be verified by inspection as follows.

- The setup time T_{Gen} is at most $\mathsf{poly}(\lambda, \log n_1, n_2)$ since (i) each $\mathsf{SEH.Gen}$ runs in time $\mathsf{poly}(\lambda, \log n_1)$, (ii) each $\mathsf{BARG.Gen}_{\mathrm{idx}}$ runs in time $\mathsf{poly}(\lambda, \log n_1, n_2)$ since we have $|C_i| \leq \mathsf{poly}(\lambda, \log n_1, n_2)$ given that the evaluation time of h, represented as a circuit, can be bounded by $\mathsf{poly}(n_2)$, and (iii) $\mathsf{SEH\text{-}Del.Gen}$ runs in time $\mathsf{poly}(\lambda, \log n_1)$.
- The prover running time is at most $\mathsf{poly}(\lambda, n_1, n_2) \leq \mathsf{poly}(\lambda, T)$ since each $\mathsf{SEH.Hash}$ and $\mathsf{SEH\text{-}Del.P}$ run in time $\mathsf{poly}(\lambda, n_1)$ and each $\mathsf{BARG.P}_{\mathrm{idx}}$ runs in time $\mathsf{poly}(\lambda, n_1, n_2)$. Also, the proof length L_π is at most $\mathsf{poly}(\lambda, \log n_1, n_2)$ since $|\mathsf{rt}_i| \leq \mathsf{poly}(\lambda, \log n_1)$ and $|\pi_i| \leq \mathsf{poly}(\lambda, \log n_1, n_2)$.
- The verifier running time (excluding the computation of $\widehat{x} \coloneqq \mathsf{SEH\text{-}Del.Encode}(\lambda, x)$) is at most $O(L_\pi) + \mathsf{poly}(\lambda, \log n_1, n_2)$ since $\mathsf{SEH.Hash}$ runs in time $\mathsf{poly}(\lambda)$, each $\mathsf{BARG.V}_{\mathrm{idx}}$ runs in time $\mathsf{poly}(\lambda, \log n_1, n_2)$, and $\mathsf{SEH\text{-}Del.V}$ runs in time $\mathsf{poly}(\lambda, \log n_1)$.

5.3 Partial Adaptive Soundness

Fix any PPT algorithm P^* and a pair of polynomials $\mathsf{poly}_{n_1}, \mathsf{poly}_{n_2}$. Our goal is to prove the following claim.

Claim 4 (Partial adaptive soundness of $\mathsf{TH\text{-}Del}$)**.** *There exists a negligible function* negl *such that for every $\lambda \in \mathbb{N}$, $n_1 \leq \mathsf{poly}_{n_1}(\lambda)$, $n_2 \leq \mathsf{poly}_{n_2}(\lambda)$, $T \coloneqq T_{\text{tree-hash}}(n_1, n_2)$, $x \in \{0,1\}^{n_1}$, and $z \in \{0,1\}^*$,*

$$\Pr\left[\begin{array}{l|l} \mathsf{V}(\mathsf{crs}, (x, (h, \mathsf{rt})), \pi) = 1 & \mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda, T, n_1, n_2) \\ \wedge\ \mathsf{TreeHash}_h(x) \neq \mathsf{rt} & ((h, \mathsf{rt}), \pi) \leftarrow \mathsf{P}^*(\mathsf{crs}, x, z) \end{array}\right] \leq \mathsf{negl}(\lambda) \ . \tag{13}$$

$\underline{\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda, T, n_1, n_2)}$:

1. Let $\ell := \log(n_1/\lambda)$.
2. Sample $h_0^{\mathsf{SEH}} \leftarrow \mathsf{SEH.Gen}(1^\lambda, \lambda, 1^\lambda)$ and $h_i^{\mathsf{SEH}} \leftarrow \mathsf{SEH.Gen}(1^\lambda, 2^i\lambda, 1^{2\lambda})$ for $\forall i \in \{1, \ldots, \ell\}$.
3. Sample $\mathsf{crs}_i^{\mathrm{idx}} \leftarrow \mathsf{BARG.Gen}_{\mathrm{idx}}(1^\lambda, 1^{|C_i|}, 2^i)$ for $\forall i \in \{0, \ldots, \ell-1\}$, where the circuit C_i is defined in the prover P below.
4. Sample $\mathsf{crs}^{\mathsf{SEH\text{-}Del}} \leftarrow \mathsf{SEH\text{-}Del.Gen}(1^\lambda, 2^\ell\lambda, 1^{2\lambda})$.
5. Output $\mathsf{crs} := (1^\lambda, \{h_i^{\mathsf{SEH}}\}_{i\in\{0,\ldots,\ell\}}, \{\mathsf{crs}_i^{\mathrm{idx}}\}_{i\in\{0,\ldots,\ell-1\}}, \mathsf{crs}^{\mathsf{SEH\text{-}Del}})$.

$\underline{\pi := \mathsf{P}(\mathsf{crs}, (x, (h, \mathsf{rt})))}$:

1. Parse crs as $(1^\lambda, \{h_i^{\mathsf{SEH}}\}_{i\in\{0,\ldots,\ell\}}, \{\mathsf{crs}_i^{\mathrm{idx}}\}_{i\in\{0,\ldots,\ell-1\}}, \mathsf{crs}^{\mathsf{SEH\text{-}Del}})$.
2. Compute $\mathsf{TreeHash}_h(x)$ along with its nodes $\{\mathsf{node}_{i,\sigma}\}_{i\in\{0,\ldots,\ell\},\sigma\in\{0,\ldots,2^i-1\}}$. That is, do the following.
   (a) Partition x into 2^ℓ blocks $\mathsf{blk}_0, \ldots, \mathsf{blk}_{2^\ell-1}$ such that $|\mathsf{blk}_0| = \cdots = |\mathsf{blk}_{2^\ell-1}| = \lambda$.
   (b) For each $i \in \{0, \ldots, \ell\}$, define $\mathsf{node}_{i,\sigma} \in \{0,1\}^\lambda$ for $\forall\sigma \in \{0, \ldots, 2^i-1\}$ as follows. First, let $\mathsf{node}_{\ell,\sigma} := \mathsf{blk}_\sigma$ for $\forall\sigma \in \{0, \ldots, 2^\ell - 1\}$. Then, for each $i \in \{\ell-1, \ldots, 0\}$, let $\mathsf{node}_{i,\sigma} := h(\mathsf{node}_{i+1,2\sigma} \parallel \mathsf{node}_{i+1,2\sigma+1})$ for $\forall\sigma \in \{0, \ldots, 2^i - 1\}$.

   (The hash value rt that P takes as input is supposed to satisfy $\mathsf{rt} = \mathsf{node}_{0,0}$.)
3. Compute $\mathsf{rt}_i := \mathsf{SEH.Hash}(h_i^{\mathsf{SEH}}, x_i)$ for each $i \in \{0, \ldots, \ell\}$, where $x_i := \mathsf{node}_{i,0} \parallel \cdots \parallel \mathsf{node}_{i,2^i-1} \in \{0,1\}^{2^i\lambda}$ is the concatenation of the tree nodes at depth i. For each $i \in \{0, \ldots, \ell-1\}$ and $\sigma \in \{0, \ldots, 2^i-1\}$, let $\mathsf{cert}_{i,\sigma}$ be the certificates that open the appropriate positions of the pre-image of rt_i to $\mathsf{node}_{i,\sigma}$; that is, $\mathsf{cert}_{i,\sigma} := (\mathsf{cert}_{i,\sigma,0}, \ldots, \mathsf{cert}_{i,\sigma,\lambda-1})$ and $\mathsf{cert}_{i,\sigma,j} := \mathsf{SEH.Open}(h_i^{\mathsf{SEH}}, b_{i,\sigma,j}, \lambda\sigma+j)$ for $\forall j \in \{0, \ldots, \lambda-1\}$, where $b_{i,\sigma,j}$ is the j-th bit of $\mathsf{node}_{i,\sigma} \in \{0,1\}^\lambda$.
4. For each $i \in \{0, \ldots, \ell-1\}$, compute $\pi_i := \mathsf{BARG.P}_{\mathrm{idx}}(\mathsf{crs}^{\mathrm{idx}}, C_i, \mathbf{w}_i)$, where $\mathbf{w}_i := \{w_{i,\sigma}\}_{\sigma\in\{0,\ldots,2^i-1\}}$, $w_{i,\sigma} := (\mathsf{node}_{i,\sigma}, \mathsf{cert}_{i,\sigma}, \mathsf{node}_{i+1,2\sigma}, \mathsf{cert}_{i+1,2\sigma}, \mathsf{node}_{i+1,2\sigma+1}, \mathsf{cert}_{i+1,2\sigma+1})$, and C_i is the following circuit.
   - C_i has $(h, h_i^{\mathsf{SEH}}, \mathsf{rt}_i, h_{i+1}^{\mathsf{SEH}}, \mathsf{rt}_{i+1})$ as hardwired inputs, and takes an index $\sigma \in \{0, \ldots, 2^i-1\}$ and a witness $w = (\mathsf{node}^{(0,0)}, \mathsf{cert}^{(0,0)}, \mathsf{node}^{(1,0)}, \mathsf{cert}^{(1,0)}, \mathsf{node}^{(1,1)}, \mathsf{cert}^{(1,1)})$ as inputs. First, C_i parses $\mathsf{node}^{(u,v)}$ as $(b_0^{(u,v)}, \ldots, b_{\lambda-1}^{(u,v)})$ and $\mathsf{cert}^{(u,v)}$ as $(\mathsf{cert}_0^{(u,v)}, \ldots, \mathsf{cert}_{\lambda-1}^{(u,v)})$ for each $(u,v) \in \{(0,0),(1,0),(1,1)\}$. Then, C_i outputs 1 iff (i) $\mathsf{SEH.Verify}(h_{i+u}^{\mathsf{SEH}}, \mathsf{rt}_{i+u}, b_j^{(u,v)}, \lambda(2^u\sigma + v) + j, \mathsf{cert}_j^{(u,v)}) = 1$ for $\forall(u,v) \in \{(0,0),(1,0),(1,1)\}$, $j \in \{0, \ldots \lambda-1\}$ and (ii) $h(\mathsf{node}^{(1,0)} \parallel \mathsf{node}^{(1,1)}) = \mathsf{node}^{(0,0)}$.
5. Compute $\pi_\ell := \mathsf{SEH\text{-}Del.P}(\mathsf{crs}^{\mathsf{SEH\text{-}Del}}, (x_\ell, (h_\ell^{\mathsf{SEH}}, \mathsf{rt}_\ell)))$.
6. Output $\pi := \{\mathsf{rt}_i, \pi_i\}_{i\in\{0,\ldots,\ell\}}$.

$\underline{b := \mathsf{V}(\mathsf{crs}, (x, (h, \mathsf{rt})), \pi)}$:

1. Parse crs as $(1^\lambda, \{h_i^{\mathsf{SEH}}\}_{i\in\{0,\ldots,\ell\}}, \{\mathsf{crs}_i^{\mathrm{idx}}\}_{i\in\{0,\ldots,\ell-1\}}, \mathsf{crs}^{\mathsf{SEH\text{-}Del}})$ and π as $\{\mathsf{rt}_i, \pi_i\}_{i\in\{0,\ldots,\ell\}}$. Abort unless $|\mathsf{rt}| = \lambda$ and $h : \{0,1\}^{2\lambda} \to \{0,1\}^\lambda$.
2. Output 1 iff all of the following hold.
   - $\mathsf{SEH.Hash}(h_0^{\mathsf{SEH}}, \mathsf{rt}) = \mathsf{rt}_0$.
   - $\mathsf{BARG.V}_{\mathrm{idx}}(\mathsf{crs}_i^{\mathrm{idx}}, C_i, \pi_i) = 1$ for $\forall i \in \{0, \ldots, \ell-1\}$, where C_i is the circuit defined in the prover P.
   - $\mathsf{SEH\text{-}Del.V}^{\widehat{x}}(\mathsf{crs}^{\mathsf{SEH\text{-}Del}}, (h_\ell^{\mathsf{SEH}}, \mathsf{rt}_\ell), \pi_\ell) = 1$, where $\widehat{x} := \mathsf{SEH\text{-}Del.Encode}(\lambda, x)$.

Figure 9: A publicly verifiable non-interactive tree-hash delegation scheme $\mathsf{TH\text{-}Del} = (\mathsf{Gen}, \mathsf{P}, \mathsf{V})$.

Extractor $E(\mathsf{params}, i^*, \sigma^*)$, where $\mathsf{params} := (1^\lambda, n_1, n_2, x, z)$:

1. Compute crs by emulating Gen with several modifications as follows.
   (a) Let $\ell := \log(n_1/\lambda)$, $\sigma^{(\mathrm{P})} := \sigma^*$, $\sigma^{(\mathrm{L})} := 2\sigma^*$, and $\sigma^{(\mathrm{R})} := 2\sigma^* + 1$.
   (b) Sample $(h_{i^*}^{\mathsf{SEH}}, \mathsf{td}_{i^*}^{\mathsf{SEH}}) \leftarrow \mathsf{SEH.TGen}(1^\lambda, 2^{i^*}\lambda, \{\lambda\sigma^{(\mathrm{P})} + j\}_{j\in\{0,\dots,\lambda-1\}})$ and $h_{i^*+1}^{\mathsf{SEH}} \leftarrow \mathsf{SEH.TGen}(1^\lambda, 2^{i^*+1}\lambda, \{\lambda\sigma^{(\mathrm{L})} + j, \lambda\sigma^{(\mathrm{R})} + j\}_{j\in\{0,\dots,\lambda-1\}})$. Also, sample h_i^{SEH} for $\forall i \in \{0,\dots,\ell\} \setminus \{i^*, i^*+1\}$ as in Gen, i.e., $h_i^{\mathsf{SEH}} \leftarrow \mathsf{SEH.Gen}(1^\lambda, \lambda, 1^\lambda)$ when $i = 0$ and $h_i^{\mathsf{SEH}} \leftarrow \mathsf{SEH.Gen}(1^\lambda, 2^i\lambda, 1^{2\lambda})$ when $i \neq 0$.
   (c) Sample $\mathsf{crs}_{i^*}^{\mathrm{idx}} \leftarrow \mathsf{BARG.TGen}_{\mathrm{idx}}(1^\lambda, 1^{|C_i|}, 2^{i^*}, \sigma^{(\mathrm{P})})$. Also, sample $\mathsf{crs}_i^{\mathrm{idx}}$ for $\forall i \in \{0,\dots,\ell-1\} \setminus \{i^*\}$ as in Gen, i.e., $\mathsf{crs}_i^{\mathrm{idx}} \leftarrow \mathsf{BARG.Gen}_{\mathrm{idx}}(1^\lambda, 1^{|C_i|}, 2^i)$.
   (d) Sample $\mathsf{crs}^{\mathsf{SEH\text{-}Del}} \leftarrow \mathsf{SEH\text{-}Del.Gen}(1^\lambda, 2^\ell\lambda, 1^{2\lambda})$ as in Gen.
   (e) Let $\mathsf{crs} := (1^\lambda, \{h_i^{\mathsf{SEH}}\}_{i\in\{0,\dots,\ell\}}, \{\mathsf{crs}_i^{\mathrm{idx}}\}_{i\in\{0,\dots,\ell-1\}}, \mathsf{crs}^{\mathsf{SEH\text{-}Del}})$.
2. Run $((h, \mathsf{rt}), \pi) \leftarrow \mathsf{P}^*(\mathsf{crs}, x, z)$ and parse π as $\{\mathsf{rt}_i, \pi_i\}_{i\in\{0,\dots,\ell\}}$.
3. Run $b_{\mathsf{V}} := \mathsf{V}(\mathsf{crs}, (x, (h, \mathsf{rt})), \pi)$.
4. Define $\widetilde{\mathsf{node}}^{(\mathrm{P})}$, $\widetilde{\mathsf{node}}^{(\mathrm{L})}$, and $\widetilde{\mathsf{node}}^{(\mathrm{R})}$ as follows.
   (a) Compute $\{\widetilde{b}_j^{(\mathrm{P})}\}_{j\in\{0,\dots,\lambda-1\}} := \mathsf{SEH.Extract}(\mathsf{td}_{i^*}^{\mathsf{SEH}}, \mathsf{rt}_{i^*})$ and $\{\widetilde{b}_j^{(\mathrm{L})}, \widetilde{b}_j^{(\mathrm{R})}\}_{j\in\{0,\dots,\lambda-1\}} := \mathsf{SEH.Extract}(\mathsf{td}_{i^*+1}^{\mathsf{SEH}}, \mathsf{rt}_{i^*+1})$.
   (b) Let $\widetilde{\mathsf{node}}^{(\mathrm{P})} := (\widetilde{b}_0^{(\mathrm{P})}, \dots, \widetilde{b}_{\lambda-1}^{(\mathrm{P})})$, $\widetilde{\mathsf{node}}^{(\mathrm{L})} := (\widetilde{b}_0^{(\mathrm{L})}, \dots, \widetilde{b}_{\lambda-1}^{(\mathrm{L})})$, and $\widetilde{\mathsf{node}}^{(\mathrm{R})} := (\widetilde{b}_0^{(\mathrm{R})}, \dots, \widetilde{b}_{\lambda-1}^{(\mathrm{R})})$.
5. Output $(h, \mathsf{rt}, b_{\mathsf{V}}, \widetilde{\mathsf{node}}^{(\mathrm{P})}, \widetilde{\mathsf{node}}^{(\mathrm{L})}, \widetilde{\mathsf{node}}^{(\mathrm{R})})$.

Figure 10: The extractor algorithm E.

At a high level, we prove Claim 4 in three steps. First, we give an extractor algorithm that takes any $(i, \sigma) \in \{0,\dots,\ell-1\} \times \{0,\dots,2^{i^*}-1\}$ as input and extracts three tree nodes from P^*. (One is extracted as the σ-th node at depth i of $\mathsf{TreeHash}_h(x)$. The other two are extracted as the left and right children.) Next, we prove a sequence of claims about consistency between the extracted tree nodes and the correct tree nodes of $\mathsf{TreeHash}_h(x)$. Finally, we prove (13) using the extractor based on the idea explained in the technical overview (Section 2.1).

First, we give an extractor algorithm. Let $\mathsf{BARG.TGen}_{\mathrm{idx}}$ be the trapdoor setup algorithm of $\mathsf{BARG}_{\mathrm{idx}}$. Then, the extractor is given in Figure 10. We note that the extractor is given $\mathsf{params} := (1^\lambda, n_1, n_2, x, z)$ and (i^*, σ^*) as inputs, where λ, n_1, n_2, x, and z are as in Claim 4 while i^* and σ^* are such that $i^* \in \{0,\dots,\ell-1\}$ and $\sigma^* \in \{0,\dots,2^{i^*}-1\}$ for $\ell := \log(n_1/\lambda)$.

Remark 4. Step 1b of the extractor is a little over-simplified. In particular, when running $(h_{i^*}^{\mathsf{SEH}}, \mathsf{td}_{i^*}^{\mathsf{SEH}}) \leftarrow \mathsf{SEH.TGen}(1^\lambda, 2^{i^*}\lambda, \{\lambda\sigma^{(\mathrm{P})} + j\}_{j\in\{0,\dots,\lambda-1\}})$ for $i^* \neq 0$, the extractor should add λ arbitrary indices to the binding index set $\{\lambda\sigma^{(\mathrm{P})} + j\}_{j\in\{0,\dots,\lambda-1\}}$ so that this execution of $\mathsf{SEH.TGen}$ is indistinguishable from $\mathsf{SEH.Gen}(1^\lambda, 2^{i^*}\lambda, 1^{2\lambda})$. For concreteness, we assume that the extractor adds the λ indices that correspond to the sibling of the tree node $(i^*, \sigma^{(\mathrm{P})})$, i.e., the extractor uses the set $\{\lambda(2\lfloor\sigma^{(\mathrm{P})}/2\rfloor) + j, \lambda(2\lfloor\sigma^{(\mathrm{P})}/2\rfloor + 1) + j\}_{j\in\{0,\dots,\lambda-1\}}$ instead of the set $\{\lambda\sigma^{(\mathrm{P})} + j\}_{j\in\{0,\dots,\lambda-1\}}$. ◊

Next, we prove a sequence of claims about the extractor. The first claim is that the values $(h, \mathsf{rt}, b_{\mathsf{V}})$ that the extractor outputs are indistinguishable from those that we obtain from P^* in (13).

Claim 5 (Indistinguishability from P^*)**.** *For any polynomial T_D, there exists a negligible function* negl *such that for any probabilistic T_D-time algorithm D and every $\lambda \in \mathbb{N}$, $n_1 \leq \mathsf{poly}_{n_1}(\lambda)$, $n_2 \leq \mathsf{poly}_{n_2}(\lambda)$, $T := T_{\text{tree-hash}}(n_1, n_2)$, $x \in \{0,1\}^{n_1}$, $z \in \{0,1\}^*$, $i^* \in \{0,\dots,\ell-1\}$, and $\sigma^* \in \{0,\dots,2^{i^*}-1\}$, it holds $|\Pr[D(\mathsf{out}_1) = 1] - \Pr[D(\mathsf{out}_2) = 1]| \leq \mathsf{negl}(\lambda)$, where out_1 and out_2 are sampled as follows.*

- ***Sampling of*** out_1***:***
  1. *Run* $\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda, T, n_1, n_2)$.

2. *Run* $((h, \mathsf{rt}), \pi) \leftarrow \mathsf{P}^*(\mathsf{crs}, x, z)$.
3. *Run* $b_{\mathsf{V}} \coloneqq \mathsf{V}(\mathsf{crs}, (x, (h, \mathsf{rt})), \pi)$.
4. *Let* $\mathsf{out}_1 \coloneqq (\mathsf{params}, h, \mathsf{rt}, b_{\mathsf{V}})$, *where* $\mathsf{params} \coloneqq (1^\lambda, n_1, n_2, x, z)$.

- ***Sampling of*** out_2***:***

1. *Run* $(h, \mathsf{rt}, b_{\mathsf{V}}, \widetilde{\mathsf{node}}^{(\mathrm{P})}, \widetilde{\mathsf{node}}^{(\mathrm{L})}, \widetilde{\mathsf{node}}^{(\mathrm{R})}) \leftarrow E(\mathsf{params}, i^*, \sigma^*)$, *where* params *is as above.*
2. *Let* $\mathsf{out}_2 \coloneqq (\mathsf{params}, h, \mathsf{rt}, b_{\mathsf{V}})$.

Proof. The sampling of out_2 differs from that of out_1 in that when E emulates Gen, some executions of $\mathsf{SEH.Gen}$ and $\mathsf{BARG.Gen}_{\mathrm{idx}}$ are replaced with $\mathsf{SEH.TGen}$ and $\mathsf{BARG.TGen}_{\mathrm{idx}}$, respectively. Thus, the indistinguishability follows from the key indistinguishability of SEH and the CRS indistinguishability of $\mathsf{BARG}_{\mathrm{idx}}$. □

The second claim is that when the extractor is used with two different inputs to extract the same tree node (e.g., the extractor is used with (i^*, σ^*) and $(i^* + 1, 2\sigma^*)$ to extract the tree node $(i^* + 1, 2\sigma^*)$), the extracted tree nodes are indistinguishable.

Claim 6 (No-signaling)**.** *For any polynomial* T_D*, there exists a negligible function* negl *such that for any probabilistic* T_D*-time algorithm* D *and every* $\lambda \in \mathbb{N}$*,* $n_1 \leq \mathsf{poly}_{n_1}(\lambda)$*,* $n_2 \leq \mathsf{poly}_{n_2}(\lambda)$*,* $T \coloneqq T_{\text{tree-hash}}(n_1, n_2)$*,* $x \in \{0,1\}^{n_1}$*,* $z \in \{0,1\}^*$*,* $i^* \in \{0, \ldots, \ell - 1\}$*,* $\sigma^* \in \{0, \ldots, 2^{i^*} - 1\}$*, and* $(\mathrm{X}, \sigma^{(\mathrm{X})}) \in \{(\mathrm{L}, 2\sigma^*), (\mathrm{R}, 2\sigma^* + 1)\}$*, it holds* $|\Pr[D(\mathsf{out}_1) = 1] - \Pr[D(\mathsf{out}_2) = 1]| \leq \mathsf{negl}(\lambda)$*, where* out_1 *and* out_2 *are sampled as follows.*

- ***Sampling of*** out_1***:***

1. $(h, \mathsf{rt}, b_{\mathsf{V}}, \widetilde{\mathsf{node}}^{(\mathrm{P})}, \widetilde{\mathsf{node}}^{(\mathrm{L})}, \widetilde{\mathsf{node}}^{(\mathrm{R})}) \leftarrow E(\mathsf{params}, i^*, \sigma^*)$, *where* $\mathsf{params} \coloneqq (1^\lambda, n_1, n_2, x, z)$.
2. *Let* $\mathsf{out}_1 \coloneqq (\mathsf{params}, i^*, \sigma^*, h, \mathsf{rt}, b_{\mathsf{V}}, \widetilde{\mathsf{node}}^{(\mathrm{X})})$

- ***Sampling of*** out_2***:***

1. $(h, \mathsf{rt}, b_{\mathsf{V}}, \widetilde{\mathsf{node}}^{(\mathrm{P})}, \widetilde{\mathsf{node}}^{(\mathrm{L})}, \widetilde{\mathsf{node}}^{(\mathrm{R})}) \leftarrow E(\mathsf{params}, i^* + 1, \sigma^{(\mathrm{X})})$. *where* params *is as above.*
2. *Let* $\mathsf{out}_2 \coloneqq (\mathsf{params}, i^*, \sigma^*, h, \mathsf{rt}, b_{\mathsf{V}}, \widetilde{\mathsf{node}}^{(\mathrm{P})})$

Proof. Note that the value $\widetilde{\mathsf{node}}^{(\mathrm{X})}$ in $E(\mathsf{params}, i^*, \sigma^*)$ and the value $\widetilde{\mathsf{node}}^{(\mathrm{P})}$ in $E(\mathsf{params}, i^* + 1, \sigma^{(\mathrm{X})})$ are both extracted from rt_{i^*+1} as the substring of the pre-image in position $(\lambda\sigma^{(\mathrm{X})}, \ldots \lambda\sigma^{(\mathrm{X})} + \lambda - 1)$. Thus, the indistinguishability follows from the key indistinguishability of SEH and the CRS indistinguishability of $\mathsf{BARG}_{\mathrm{idx}}$. □

The third claim is that when the extractor is used to extract leaf nodes, the extracted nodes are consistent with the input x.

Claim 7 (Consistency with the input)**.** *There exists a negligible function* negl *such that for every* $\lambda \in \mathbb{N}$*,* $n_1 \leq \mathsf{poly}_{n_1}(\lambda)$*,* $n_2 \leq \mathsf{poly}_{n_2}(\lambda)$*,* $T \coloneqq T_{\text{tree-hash}}(n_1, n_2)$*,* $x \in \{0,1\}^{n_1}$*,* $z \in \{0,1\}^*$*,* $\sigma^* \in \{0, \ldots, 2^{\ell-1} - 1\}$*, and* $(\mathrm{X}, \sigma^{(\mathrm{X})}) \in \{(\mathrm{L}, 2\sigma^*), (\mathrm{R}, 2\sigma^* + 1)\}$*,*

$$\Pr\left[\begin{array}{l|l} b_{\mathsf{V}} = 1 & \\ \wedge\, \widetilde{\mathsf{node}}^{(\mathrm{X})} \neq \mathsf{blk}_{\sigma^{(\mathrm{X})}} & (h, \mathsf{rt}, b_{\mathsf{V}}, \widetilde{\mathsf{node}}^{(\mathrm{P})}, \widetilde{\mathsf{node}}^{(\mathrm{L})}, \widetilde{\mathsf{node}}^{(\mathrm{R})}) \leftarrow E(\mathsf{params}, \ell - 1, \sigma^*) \end{array}\right] \leq \mathsf{negl}(\lambda) \ ,$$

where (i) $\mathsf{params} \coloneqq (1^\lambda, n_1, n_2, x, z)$ *and (ii)* $\{\mathsf{blk}_\sigma\}_{\sigma \in \{0, \ldots, 2^\ell - 1\}}$ *are the partition of* x *as in the computation of* $\mathsf{TreeHash}_h(x)$.

Proof. Note that $b_{\mathsf{V}} = 1$ implies that E obtains π such that $\mathsf{V}(\mathsf{crs}, (x, (h, \mathsf{rt})), \pi) = 1$, which in turn implies that π contains π_ℓ such that $\mathsf{SEH\text{-}Del.V}^{\widehat{x}}(\mathsf{crs}^{\mathsf{SEH\text{-}Del}}, (h_\ell^{\mathsf{SEH}}, \mathsf{rt}_\ell), \pi_\ell) = 1$, where $\widehat{x} \coloneqq \mathsf{SEH\text{-}Del.Encode}(\lambda, x)$. Also, note that $\widetilde{\mathsf{node}}^{(\mathrm{X})}$ is extracted from rt_ℓ as the substring of x in position $(\lambda\sigma^{(\mathrm{X})}, \ldots \lambda\sigma^{(\mathrm{X})} + \lambda - 1)$, and $\mathsf{blk}_{\sigma^{(\mathrm{X})}}$ is the actual substring in that position. Thus, this claim follows from the partially adaptive somewhere soundness of $\mathsf{SEH\text{-}Del}$. □

The fourth claim is that when the extractor is used to extract the root node, the extracted node is equal to the root rt that the extractor outputs.

Claim 8 (Consistency with the output)**.** *There exists a negligible function* negl *such that for every* $\lambda \in \mathbb{N}$, $n_1 \leq \mathsf{poly}_{n_1}(\lambda)$, $n_2 \leq \mathsf{poly}_{n_2}(\lambda)$, $T := T_{\text{tree-hash}}(n_1, n_2)$, $x \in \{0,1\}^{n_1}$, *and* $z \in \{0,1\}^*$,

$$\Pr\left[\begin{array}{l} b_{\mathsf{V}} = 1 \\ \wedge\ \widetilde{\mathsf{node}}^{(\mathrm{P})} \neq \mathsf{rt} \end{array} \middle| (h, \mathsf{rt}, b_{\mathsf{V}}, \widetilde{\mathsf{node}}^{(\mathrm{P})}, \widetilde{\mathsf{node}}^{(\mathrm{L})}, \widetilde{\mathsf{node}}^{(\mathrm{R})}) \leftarrow E(\mathsf{params}, 0, 0)\right] \leq \mathsf{negl}(\lambda) \ ,$$

where $\mathsf{params} := (1^\lambda, n_1, n_2, x, z)$.

Proof. Note that $b_{\mathsf{V}} = 1$ implies that E obtains π such that $\mathsf{V}(\mathsf{crs}, (x, (h, \mathsf{rt})), \pi) = 1$, which in turn implies that π contains rt_0 such that $\mathsf{SEH.Hash}(h_0^{\mathsf{SEH}}, \mathsf{rt}) = \mathsf{rt}_0$. Also, note that $\widetilde{\mathsf{node}}^{(\mathrm{P})}$ is extracted from rt_0 as the pre-image of rt_0. Thus, this claim follows from the somewhere extractability of SEH. □

The final claim is that the tree nodes that the extractor extracts satisfy the parent-children relation.

Claim 9 (Local consistency)**.** *There exists a negligible function* negl *such that for every* $\lambda \in \mathbb{N}$, $n_1 \leq \mathsf{poly}_{n_1}(\lambda)$, $n_2 \leq \mathsf{poly}_{n_2}(\lambda)$, $T := T_{\text{tree-hash}}(n_1, n_2)$, $x \in \{0,1\}^{n_1}$, $z \in \{0,1\}^*$, $i^* \in \{0, \ldots, \ell-1\}$, *and* $\sigma^* \in \{0, \ldots, 2^{i^*} - 1\}$,

$$\Pr\left[\begin{array}{l} b_{\mathsf{V}} = 1 \\ \wedge\ h(\widetilde{\mathsf{node}}^{(\mathrm{L})} \| \widetilde{\mathsf{node}}^{(\mathrm{R})}) \neq \widetilde{\mathsf{node}}^{(\mathrm{P})} \end{array} \middle| (h, \mathsf{rt}, b_{\mathsf{V}}, \widetilde{\mathsf{node}}^{(\mathrm{P})}, \widetilde{\mathsf{node}}^{(\mathrm{L})}, \widetilde{\mathsf{node}}^{(\mathrm{R})}) \leftarrow E(\mathsf{params}, i^*, \sigma^*)\right] \leq \mathsf{negl}(\lambda) \ , \tag{14}$$

where $\mathsf{params} := (1^\lambda, n_1, n_2, x, z)$.

Proof. Note that $b_{\mathsf{V}} = 1$ implies that E obtains π such that $\mathsf{V}(\mathsf{crs}, (x, (h, \mathsf{rt})), \pi) = 1$, which in turn implies that π contains π_{i^*} such that $\mathsf{BARG.V}_{\mathrm{idx}}(\mathsf{crs}^{\mathrm{idx}}, C_{i^*}, \pi_{i^*}) = 1$. Thus, the semi-adaptive somewhere soundness of $\mathsf{BARG}_{\mathrm{idx}}$ implies that when $b_{\mathsf{V}} = 1$, with overwhelming probability there exists $w = (\mathsf{node}^{(0,0)}, \mathsf{cert}^{(0,0)}, \mathsf{node}^{(1,0)}, \mathsf{cert}^{(1,0)}, \mathsf{node}^{(1,1)}, \mathsf{cert}^{(1,1)})$ such that (i) $\mathsf{cert}^{(u,v)}$ is a valid certificate of $\mathsf{node}^{(u,v)}$ w.r.t. hash value rt_{i^*+u} and position $(\lambda(2^u\sigma^* + v), \ldots, \lambda(2^u\sigma^* + v) + \lambda - 1)$ for $\forall (u,v) \in \{(0,0),(1,0),(1,1)\}$, and (ii) $h(\mathsf{node}^{(1,0)} \| \mathsf{node}^{(1,1)}) = \mathsf{node}^{(0,0)}$. Also, note that for such $w = (\mathsf{node}^{(0,0)}, \mathsf{cert}^{(0,0)}, \mathsf{node}^{(1,0)}, \mathsf{cert}^{(1,0)}, \mathsf{node}^{(1,1)}, \mathsf{cert}^{(1,1)})$, the somewhere extractability of SEH guarantees that with overwhelming probability, $\widetilde{\mathsf{node}}^{(\mathrm{P})} = \mathsf{node}^{(0,0)}$, $\widetilde{\mathsf{node}}^{(\mathrm{L})} = \mathsf{node}^{(1,0)}$, and $\widetilde{\mathsf{node}}^{(\mathrm{R})} = \mathsf{node}^{(1,1)}$. Combining these two, we obtain (14) as desired. □

Finally, we prove Claim 4 using the extractor and the above claims.

Proof of Claim 4. Let negl_0 and negl_1 be the negligible functions that are guaranteed to exist by Claim 5 and Claim 6 respectively for a sufficiently large polynomial T_D,[18] and negl_2, negl_3, and negl_4 be the negligible functions that are guaranteed to exist by Claim 7, Claim 8, and Claim 9 respectively. Fix any $\lambda \in \mathbb{N}$, $n_1 \leq \mathsf{poly}_{n_1}(\lambda)$, $n_2 \leq \mathsf{poly}_{n_2}(\lambda)$, $T := T_{\text{tree-hash}}(n_1, n_2)$, $x \in \{0,1\}^{n_1}$, and $z \in \{0,1\}^*$ as stated in the claim, and let $\mathsf{params} := (1^\lambda, n_1, n_2, x, z)$.

First, we show that the extractor extracts the correct tree nodes of $\mathsf{TreeHash}_h(x)$. Concretely, we show that for every $i^* \in \{0, \ldots, \ell - 1\}$ and $\sigma^* \in \{0, \ldots, 2^{i^*} - 1\}$,

$$\Pr\left[\begin{array}{l} b_{\mathsf{V}} = 1 \\ \wedge\ \widetilde{\mathsf{node}}^{(\mathrm{P})} \neq \mathsf{node}_{i^*,\sigma^*} \end{array} \middle| (h, \mathsf{rt}, b_{\mathsf{V}}, \widetilde{\mathsf{node}}^{(\mathrm{P})}, \widetilde{\mathsf{node}}^{(\mathrm{L})}, \widetilde{\mathsf{node}}^{(\mathrm{R})}) \leftarrow E(\mathsf{params}, i^*, \sigma^*)\right] \leq \epsilon(\lambda, i^*) \ , \tag{15}$$

where (i) $\{\mathsf{node}_{i,\sigma}\}_{i \in \{0,\ldots,\ell\}, \sigma \in \{0,\ldots,2^\ell - 1\}}$ is the correct tree nodes of $\mathsf{TreeHash}_h(x)$ and (ii) ϵ is defined as

$$\epsilon(\lambda, i) := 2^{\ell - i}(\mathsf{negl}_1(\lambda) + \mathsf{negl}_2(\lambda) + \mathsf{negl}_4(\lambda)) - 2\mathsf{negl}_1(\lambda) - \mathsf{negl}_4(\lambda) \ .$$

We prove (15) by induction on i^*, where the base case is $i^* = \ell - 1$.

[18] Specifically, T_D is required to be larger than the running time of the reduction algorithms that we implicitly design below using P^*.

Base case: Fix any $\sigma^* \in \{0,\dots,2^{\ell-1}-1\}$. From Claim 7 and the union bound, we have

$$\Pr\left[\begin{array}{l} b_{\mathsf{V}}=1 \\ \wedge\, \exists(\mathrm{X},\sigma^{(\mathrm{X})}) \in \{(\mathrm{L},2\sigma^*),(\mathrm{R},2\sigma^*+1)\} \text{ s.t. } \widetilde{\mathsf{node}}^{(\mathrm{X})} \neq \mathsf{blk}_{\sigma^{(\mathrm{X})}} \end{array}\right] \leq 2\mathsf{negl}_2(\lambda) \ ,$$

where (i) the probability is taken over $(h,\mathsf{rt},b_{\mathsf{V}},\widetilde{\mathsf{node}}^{(\mathrm{P})},\widetilde{\mathsf{node}}^{(\mathrm{L})},\widetilde{\mathsf{node}}^{(\mathrm{R})}) \leftarrow E(\mathsf{params},\ell-1,\sigma^*)$ and (ii) $\{\mathsf{blk}_\sigma\}_{\sigma\in\{0,\dots,2^\ell-1\}}$ are defined as in Claim 7. Thus, from Claim 9 and the union bound, we have

$$\Pr\left[\begin{array}{l} b_{\mathsf{V}}=1 \\ \wedge\, h(\mathsf{blk}_{2\sigma^*} \,\|\, \mathsf{blk}_{2\sigma^*+1}) \neq \widetilde{\mathsf{node}}^{(\mathrm{P})} \end{array}\right] \leq 2\mathsf{negl}_2(\lambda)+\mathsf{negl}_4(\lambda) = \epsilon(\lambda,\ell-1) \ ,$$

where the probability is again taken over $(h,\mathsf{rt},b_{\mathsf{V}},\widetilde{\mathsf{node}}^{(\mathrm{P})},\widetilde{\mathsf{node}}^{(\mathrm{L})},\widetilde{\mathsf{node}}^{(\mathrm{R})}) \leftarrow E(\mathsf{params},\ell-1,\sigma^*)$. Since we have $h(\mathsf{blk}_{2\sigma^*} \,\|\, \mathsf{blk}_{2\sigma^*+1}) = \mathsf{node}_{\ell-1,\sigma^*}$ by the definition of $\mathsf{node}_{\ell-1,\sigma^*}$, we have obtained (15) for $i^* = \ell-1$ as desired.

Inductive step: Assume we have (15) for $i^* = i+1$ and every $\sigma^*_{i+1} \in \{0,\dots,2^{i+1}-1\}$. To prove (15) for $i^* = i$, fix any $\sigma^*_i \in \{0,\dots,2^i-1\}$. For each $(\mathrm{X},\sigma^{(\mathrm{X})}) \in \{(\mathrm{L},2\sigma^*_i),(\mathrm{R},2\sigma^*_i+1)\}$, we use (15) for $i^* = i+1$ and $\sigma^* = \sigma^{(\mathrm{X})} \in \{0,\dots,2^{i+1}-1\}$ to obtain

$$\Pr\left[\begin{array}{l} b_{\mathsf{V}}=1 \\ \wedge\, \widetilde{\mathsf{node}}^{(\mathrm{P})} \neq \mathsf{node}_{i+1,\sigma^{(\mathrm{X})}} \end{array}\middle|\ (h,\mathsf{rt},b_{\mathsf{V}},\widetilde{\mathsf{node}}^{(\mathrm{P})},\widetilde{\mathsf{node}}^{(\mathrm{L})},\widetilde{\mathsf{node}}^{(\mathrm{R})}) \leftarrow E(\mathsf{params},i+1,\sigma^{(\mathrm{X})})\right] \leq \epsilon(\lambda,i+1) \ .$$

Then, we use Claim 6 and the union bound to obtain

$$\Pr\left[\begin{array}{l} b_{\mathsf{V}}=1 \\ \wedge\, \widetilde{\mathsf{node}}^{(\mathrm{X})} \neq \mathsf{node}_{i+1,\sigma^{(\mathrm{X})}} \end{array}\middle|\ (h,\mathsf{rt},b_{\mathsf{V}},\widetilde{\mathsf{node}}^{(\mathrm{P})},\widetilde{\mathsf{node}}^{(\mathrm{L})},\widetilde{\mathsf{node}}^{(\mathrm{R})}) \leftarrow E(\mathsf{params},i,\sigma^*_i)\right] \leq \epsilon(\lambda,i+1)+\mathsf{negl}_1(\lambda) \ .$$

Next, we use the union bound to obtain

$$\Pr\left[\begin{array}{l} b_{\mathsf{V}}=1 \\ \wedge\, \exists(\mathrm{X},\sigma^{(\mathrm{X})}) \in \{(\mathrm{L},2\sigma^*_i),(\mathrm{R},2\sigma^*_i+1)\} \text{ s.t. } \widetilde{\mathsf{node}}^{(\mathrm{X})} \neq \mathsf{node}_{i+1,\sigma^{(\mathrm{X})}} \end{array}\right] \leq 2(\epsilon(\lambda,i+1)+\mathsf{negl}_1(\lambda)) \ ,$$

where the probability is taken over $(h,\mathsf{rt},b_{\mathsf{V}},\widetilde{\mathsf{node}}^{(\mathrm{P})},\widetilde{\mathsf{node}}^{(\mathrm{L})},\widetilde{\mathsf{node}}^{(\mathrm{R})}) \leftarrow E(\mathsf{params},i,\sigma^*_i)$. Next, by using Claim 9 and the union bound, we obtain

$$\Pr\left[\begin{array}{l} b_{\mathsf{V}}=1 \\ \wedge\, h(\mathsf{node}_{i+1,2\sigma^*_i} \,\|\, \mathsf{node}_{i+1,2\sigma^*_i+1}) \neq \widetilde{\mathsf{node}}^{(\mathrm{P})} \end{array}\right] \leq 2(\epsilon(\lambda,i+1)+\mathsf{negl}_1(\lambda))+\mathsf{negl}_4(\lambda) \ ,$$

where the probability is again taken over $(h,\mathsf{rt},b_{\mathsf{V}},\widetilde{\mathsf{node}}^{(\mathrm{P})},\widetilde{\mathsf{node}}^{(\mathrm{L})},\widetilde{\mathsf{node}}^{(\mathrm{R})}) \leftarrow E(\mathsf{params},i,\sigma^*_i)$. Now, observe that we have obtained (15) for $i^* = i$ and $\sigma^* = \sigma^*_i$ since (i) we have $h(\mathsf{node}_{i+1,2\sigma^*_i} \,\|\, \mathsf{node}_{i+1,2\sigma^*_i+1}) = \mathsf{node}_{i,\sigma^*_i}$ by the definition of $\mathsf{node}_{i,\sigma^*_i}$ and (ii) we have

$$\begin{aligned} &2(\epsilon(\lambda,i+1)+\mathsf{negl}_1(\lambda))+\mathsf{negl}_4(\lambda) \\ &= 2\left(2^{\ell-i-1}(\mathsf{negl}_1(\lambda)+\mathsf{negl}_2(\lambda)+\mathsf{negl}_4(\lambda)) - 2\mathsf{negl}_1(\lambda) - \mathsf{negl}_4(\lambda) + \mathsf{negl}_1(\lambda)\right)+\mathsf{negl}_4(\lambda) \\ &= 2^{\ell-i}(\mathsf{negl}_1(\lambda)+\mathsf{negl}_2(\lambda)+\mathsf{negl}_4(\lambda)) - 2\mathsf{negl}_1(\lambda) - \mathsf{negl}_4(\lambda) \\ &= \epsilon(\lambda,i) \ . \end{aligned}$$

Thus, by induction, we indeed have (15) for every $i^* \in \{0,\dots,\ell-1\}$ and $\sigma^* \in \{0,\dots,2^{i^*}-1\}$.

Now, we complete the proof of Claim 4 using what we have shown above. By setting $i^* = 0$ and $\sigma^* = 0$ in (15), we obtain

$$\Pr\left[\begin{array}{l} b_{\mathsf{V}} = 1 \\ \wedge \widetilde{\mathsf{node}}^{(\mathrm{P})} \neq \mathsf{node}_{0,0} \end{array} \middle| (h, \mathsf{rt}, b_{\mathsf{V}}, \widetilde{\mathsf{node}}^{(\mathrm{P})}, \widetilde{\mathsf{node}}^{(\mathrm{L})}, \widetilde{\mathsf{node}}^{(\mathrm{R})}) \leftarrow E(\mathsf{params}, 0, 0)\right] \leq \epsilon(\lambda, 0) \ .$$

From Claim 8 and the union bound, we have

$$\Pr\left[\begin{array}{l} b_{\mathsf{V}} = 1 \\ \wedge \mathsf{rt} \neq \mathsf{node}_{0,0} \end{array} \middle| (h, \mathsf{rt}, b_{\mathsf{V}}, \widetilde{\mathsf{node}}^{(\mathrm{P})}, \widetilde{\mathsf{node}}^{(\mathrm{L})}, \widetilde{\mathsf{node}}^{(\mathrm{R})}) \leftarrow E(\mathsf{params}, 0, 0)\right] \leq \epsilon(\lambda, 0) + \mathsf{negl}_3(\lambda) \ .$$

Since we have $\mathsf{TreeHash}_h(x) = \mathsf{node}_{0,0}$ by the definition of $\mathsf{node}_{0,0}$, we have

$$\Pr\left[\begin{array}{l} b_{\mathsf{V}} = 1 \\ \wedge \mathsf{TreeHash}_h(x) \neq \mathsf{rt} \end{array} \middle| (h, \mathsf{rt}, b_{\mathsf{V}}, \widetilde{\mathsf{node}}^{(\mathrm{P})}, \widetilde{\mathsf{node}}^{(\mathrm{L})}, \widetilde{\mathsf{node}}^{(\mathrm{R})}) \leftarrow E(\mathsf{params}, 0, 0)\right] \leq \epsilon(\lambda, 0) + \mathsf{negl}_3(\lambda) \ .$$

Finally, from Claim 5 and the union bound, we have

$$\Pr\left[\begin{array}{l} \mathsf{V}(\mathsf{crs}, (x, (h, \mathsf{rt})), \pi) = 1 \\ \wedge \mathsf{TreeHash}_h(x) \neq \mathsf{rt} \end{array} \middle| \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda, T, n_1, n_2) \\ ((h, \mathsf{rt}), \pi) \leftarrow \mathsf{P}^*(\mathsf{crs}, x, z) \end{array}\right] \leq \epsilon(\lambda, 0) + \mathsf{negl}_3(\lambda) + \mathsf{negl}_0(\lambda) \ .$$

Since $\epsilon(\lambda, 0) = 2^\ell(\mathsf{negl}_1(\lambda) + \mathsf{negl}_2(\lambda) + \mathsf{negl}_4(\lambda)) - 2\mathsf{negl}_1(\lambda) - \mathsf{negl}_4(\lambda) \leq (n_1/\lambda)(\mathsf{negl}_1(\lambda) + \mathsf{negl}_2(\lambda) + \mathsf{negl}_4(\lambda)) - 2\mathsf{negl}_1(\lambda) - \mathsf{negl}_4(\lambda)$ is negligible due to $n_1 \leq \mathsf{poly}_{n_1}(\lambda)$, we have obtained (13) as desired.

This completes the proof of Claim 4. □

This completes the proof of Lemma 3. □

6 Holographic SNARG for Batch-NP

In this section, we construct a holographic SNARG for batch NP.

Theorem 4. *Assume the existence of the following primitives.*

- *A somewhere extractable hash function family* SEH.
- *A semi-adaptive somewhere-sound publicly verifiable non-interactive BARG for the index language.*
- *A partially adaptive somewhere-sound holographic delegation scheme* SEH-Del *for* SEH.

Then, there exists a publicly verifiable non-interactive BARG for **CSAT** *that is (i) weakly semi-adaptive somewhere sound and (ii) holographic w.r.t. the same encoding algorithm as* SEH-Del. *The scheme is public-coin if the above-listed primitives are public-coin.*

Since all the primitives listed in Theorem 4 exist under the LWE assumption (Theorem 2, Theorem 3, Lemma 2), Theorem 4 implies the following corollary.

Corollary 2. *Under the LWE assumption, there exists a public-coin non-interactive BARG for* **CSAT** *that is (i) weakly semi-adaptive somewhere sound and (ii) LDE-holographic, where for the security parameter λ and an input x of length N, the encoding algorithm* Encode *outputs the LDE of x w.r.t. an arbitrary LDE parameter $(\mathbb{F}, \mathbb{H}, m)$ such that $|\mathbb{F}| \leq \mathsf{poly}(|\mathbb{H}|) \leq \mathsf{poly}(\log N)$, $|\mathbb{H}|^m \leq \mathsf{poly}(N)$, and $m|\mathbb{H}|/|\mathbb{F}| \leq O(1)$.*

In the rest of this section, we prove Theorem 4.

Proof of Theorem 4. Let $\mathsf{SEH} = (\mathsf{SEH.Gen}, \mathsf{SEH.TGen}, \mathsf{SEH.Hash}, \mathsf{SEH.Open}, \mathsf{SEH.Verify}, \mathsf{SEH.Extract})$ be a somewhere extractable hash function family, $\mathsf{BARG}_{\mathrm{idx}} = (\mathsf{BARG.Gen}_{\mathrm{idx}}, \mathsf{BARG.P}_{\mathrm{idx}}, \mathsf{BARG.V}_{\mathrm{idx}})$ be a semi-adaptive somewhere-sound publicly verifiable non-interactive BARG for the index language, and $\mathsf{SEH\text{-}Del} = (\mathsf{SEH\text{-}Del.Gen}, \mathsf{SEH\text{-}Del.P}, \mathsf{SEH\text{-}Del.V}, \mathsf{SEH\text{-}Del.Encode})$ be a partially adaptive somewhere-sound holographic delegation scheme for SEH. Our scheme $\mathsf{BARG} = (\mathsf{Gen}, \mathsf{P}, \mathsf{V})$ is given in Figure 11.

$\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda, 1^n, 1^{M_C}, k)$:

1. Sample $h \leftarrow \mathsf{SEH.Gen}(1^\lambda, nk, 1^n)$.
2. Sample $\mathsf{crs}^{\mathrm{idx}} \leftarrow \mathsf{BARG.Gen}_{\mathrm{idx}}(1^\lambda, 1^{M'_C}, k)$, where $M'_C = \mathsf{poly}(\lambda, |C|, \log k)$ is the size of the circuit C' that is defined in the prover P below.
3. Sample $\mathsf{crs}^{\mathsf{SEH\text{-}Del}} \leftarrow \mathsf{SEH\text{-}Del.Gen}(1^\lambda, nk, 1^n)$.
4. Output $\mathsf{crs} = (1^\lambda, 1^n, 1^{M_C}, k, h, \mathsf{crs}^{\mathrm{idx}}, \mathsf{crs}^{\mathsf{SEH\text{-}Del}})$.

$\pi := \mathsf{P}(\mathsf{crs}, C, \mathbf{x}, \mathbf{w})$:

1. Parse crs as $(1^\lambda, 1^n, 1^{M_C}, k, h, \mathsf{crs}^{\mathrm{idx}}, \mathsf{crs}^{\mathsf{SEH\text{-}Del}})$, $\mathbf{x}$ as $(x_1, \ldots, x_k)$, and $\mathbf{w}$ as $(w_1, \ldots, w_k)$.
2. Compute $\mathsf{rt} := \mathsf{SEH.Hash}(h, \mathbf{x})$ and $\mathsf{cert}_{i,j} := \mathsf{SEH.Open}(h, \mathbf{x}, n(i-1)+j)$ for $\forall i \in [k], j \in [n]$, where $\mathbf{x}$ is viewed as a binary string of length nk.
3. Run $\pi^{\mathrm{idx}} := \mathsf{BARG.P}_{\mathrm{idx}}(\mathsf{crs}^{\mathrm{idx}}, C', \mathbf{w}')$, where $\mathbf{w}' := \{w'_i\}_{i\in[k]}$, $w'_i := (x_i, w_i, \{\mathsf{cert}_{i,j}\}_{j\in[n]})$, and C' is the following circuit.
   - C' has (h, rt) as hardwired inputs, and takes an index $i \in [k]$ and a witness $w'_i = (x_i, w_i, \{\mathsf{cert}_{i,j}\}_{j\in[n]})$ as inputs. First, C' parses x_i as $(x_{i,1}, \ldots, x_{i,n}) \in \{0,1\}^n$. Then, C' outputs 1 iff (i) $\mathsf{SEH.Verify}(h, \mathsf{rt}, x_{i,j}, n(i-1)+j, \mathsf{cert}_{i,j}) = 1$ for $\forall j \in [n]$ and (ii) $C(x_i, w_i) = 1$.
4. Run $\pi^{\mathsf{SEH\text{-}Del}} := \mathsf{SEH\text{-}Del.P}(\mathsf{crs}^{\mathsf{SEH\text{-}Del}}, (\mathbf{x}, (h, \mathsf{rt})))$.
5. Output $\pi := (\mathsf{rt}, \pi^{\mathrm{idx}}, \pi^{\mathsf{SEH\text{-}Del}})$.

$b := \mathsf{V}(\mathsf{crs}, C, \mathbf{x}, \pi)$:

1. Parse crs as $(1^\lambda, 1^n, 1^{M_C}, k, h, \mathsf{crs}^{\mathrm{idx}}, \mathsf{crs}^{\mathsf{SEH\text{-}Del}})$ and π as $(\mathsf{rt}, \pi^{\mathrm{idx}}, \pi^{\mathsf{SEH\text{-}Del}})$. Abort unless $|C| \le M_C$.
2. Output 1 iff all of the following hold.
   (a) $\mathsf{BARG.V}_{\mathrm{idx}}(\mathsf{crs}^{\mathrm{idx}}, C', \pi^{\mathrm{idx}}) = 1$, where C' is defined as in the prover P.
   (b) $\mathsf{SEH\text{-}Del.V}^{\widehat{\mathbf{x}}}(\mathsf{crs}^{\mathsf{SEH\text{-}Del}}, (h, \mathsf{rt}), \pi^{\mathsf{SEH\text{-}Del}}) = 1$, where $\widehat{\mathbf{x}} := \mathsf{SEH\text{-}Del.Encode}(\lambda, \mathbf{x})$.

Figure 11: Holographic publicly verifiable non-interactive BARG BARG.

6.1 Completeness

The completeness can be verified by inspection. Also, it is easy to see that $(\mathsf{Gen}, \mathsf{P}, \mathsf{V})$ is holographic w.r.t. the same encoding algorithm as $\mathsf{SEH\text{-}Del}$. (In particular, the verifier V only uses $\mathbf{x}$ to execute $\mathsf{SEH\text{-}Del.V}^{\widehat{\mathbf{x}}}$ for $\widehat{\mathbf{x}} := \mathsf{SEH\text{-}Del.Encode}(\lambda, \mathbf{x})$ in the last step.)

6.2 Efficiency

The efficiency can be verified by inspection.

6.3 Weakly Semi-Adaptive Somewhere Soundness

To prove the weakly semi-adaptive somewhere soundness, we first give a trapdoor setup algorithm TGen. Let $\mathsf{BARG.TGen}_{\mathrm{idx}}$ be the trapdoor setup algorithm of $\mathsf{BARG}_{\mathrm{idx}}$. Then, TGen is given in Figure 12.

Since the CRS indistinguishability follows directly from the key indistinguishability of SEH and the CRS indistinguishability of $\mathsf{BARG}_{\mathrm{idx}}$, we focus on proving the weakly semi-adaptive somewhere soundness. Assume for contradiction that BARG is not weakly semi-adaptive somewhere sound, i.e., there exist a pair of PPT algorithms $\mathsf{P}^* = (\mathsf{P}^*_1, \mathsf{P}^*_2)$ and a triple of polynomials $\mathsf{poly}_k, \mathsf{poly}_C, \mathsf{poly}_{\mathsf{P}^*}$ such that for infinitely many $\lambda \in \mathbb{N}$, there exist $k \le \mathsf{poly}_k(\lambda)$, $M_C \le \mathsf{poly}_C(\lambda)$, $n \le M_C$, and $z \in \{0,1\}^*$ such that

$$\Pr\left[\begin{array}{l|l} \mathsf{V}(\mathsf{crs}, C, \mathbf{x}, \pi) = 1 & (\mathsf{st}, i^*, \mathbf{x}) \leftarrow \mathsf{P}^*_1(1^\lambda, 1^n, 1^{M_C}, k, z) \\ \wedge\, i^* \in [k] & \mathsf{crs} \leftarrow \mathsf{TGen}(1^\lambda, 1^n, 1^{M_C}, k, i^*) \\ \wedge\, (C, x_{i^*}) \notin \mathbf{CSAT} & (C, \pi) \leftarrow \mathsf{P}^*_2(\mathsf{st}, \mathsf{crs}) \end{array}\right] \ge \frac{1}{\mathsf{poly}_{\mathsf{P}^*}(\lambda)} \ . \tag{16}$$

$\mathsf{crs} \leftarrow \mathsf{TGen}(1^\lambda, 1^n, 1^{M_C}, k, i)$:

1. Sample $(h, \mathsf{td}) \leftarrow \mathsf{SEH.TGen}(1^\lambda, nk, \{n(i-1)+j\}_{j\in[n]})$.
2. Sample $\mathsf{crs}^{\mathrm{idx}} \leftarrow \mathsf{BARG.TGen}_{\mathrm{idx}}(1^\lambda, 1^{M'_C}, k, i)$, where M'_C is defined as in Gen.
3. Sample $\mathsf{crs}^{\mathsf{SEH\text{-}Del}} \leftarrow \mathsf{SEH\text{-}Del.Gen}(1^\lambda, nk, 1^n)$.
4. Output $\mathsf{crs} = (1^\lambda, 1^n, 1^{M_C}, k, h, \mathsf{crs}^{\mathrm{idx}}, \mathsf{crs}^{\mathsf{SEH\text{-}Del}})$.

Figure 12: Trapdoor setup algorithm TGen

$(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{TGen}'(1^\lambda, 1^n, 1^{M_C}, k, i)$:
Identical with TGen except that it additionally outputs td, which is the trapdoor obtained by $\mathsf{SEH.TGen}$.

$b := \mathsf{V}'(\mathsf{crs}, C, \mathbf{x}, \pi, \mathsf{td}, i^*)$:

1. Parse crs as $(1^\lambda, 1^n, 1^{M_C}, k, h, \mathsf{crs}^{\mathrm{idx}}, \mathsf{crs}^{\mathsf{SEH\text{-}Del}})$ and π as $(\mathsf{rt}, \pi^{\mathrm{idx}}, \pi^{\mathsf{SEH\text{-}Del}})$. Abort unless $|C| \le M_C$.
2. Output 1 iff all of the following hold.
   (a) $\mathsf{BARG.V}_{\mathrm{idx}}(\mathsf{crs}^{\mathrm{idx}}, C', \pi^{\mathrm{idx}}) = 1$, where C' is defined as in the prover P.
   (b) $\mathsf{SEH\text{-}Del.V}^{\widehat{\mathbf{x}}}(\mathsf{crs}^{\mathsf{SEH\text{-}Del}}, (h, \mathsf{rt}), \pi^{\mathsf{SEH\text{-}Del}}) = 1$, where $\widehat{\mathbf{x}} := \mathsf{SEH\text{-}Del.Encode}(\lambda, \mathbf{x})$.
   (c) $x_{i^*} = \widetilde{x}_{i^*}$, where $\widetilde{x}_{i^*} := (\widetilde{x}_{i^*,1}, \ldots, \widetilde{x}_{i^*,n}) \in \{0,1\}^n$ for $\{\widetilde{x}_{i^*,j}\}_{j\in[n]} := \mathsf{SEH.Extract}(\mathsf{td}, \mathsf{rt})$.

Figure 13: Hybrid verifier V' and trapdoor setup algorithm TGen'.

$b := \mathsf{V}''(\mathsf{crs}, C, \mathbf{x}, \pi, \mathsf{td}, i^*)$:

1. Parse crs as $(1^\lambda, 1^n, 1^{M_C}, k, h, \mathsf{crs}^{\mathrm{idx}}, \mathsf{crs}^{\mathsf{SEH\text{-}Del}})$ and π as $(\mathsf{rt}, \pi^{\mathrm{idx}}, \pi^{\mathsf{SEH\text{-}Del}})$. Abort unless $|C| \le M_C$.
2. Output 1 iff all of the following hold.
   (a) $\mathsf{BARG.V}_{\mathrm{idx}}(\mathsf{crs}^{\mathrm{idx}}, C', \pi^{\mathrm{idx}}) = 1$, where C' is defined as in the prover P.
   (b) $\mathsf{SEH\text{-}Del.V}^{\widehat{\mathbf{x}}}(\mathsf{crs}^{\mathsf{SEH\text{-}Del}}, (h, \mathsf{rt}), \pi^{\mathsf{SEH\text{-}Del}}) = 1$, where $\widehat{\mathbf{x}} := \mathsf{SEH\text{-}Del.Encode}(\lambda, \mathbf{x})$.
   (c) $x_{i^*} = \widetilde{x}_{i^*}$, where $\widetilde{x}_{i^*} := (\widetilde{x}_{i^*,1}, \ldots, \widetilde{x}_{i^*,n}) \in \{0,1\}^n$ for $\{\widetilde{x}_{i^*,j}\}_{j\in[n]} := \mathsf{SEH.Extract}(\mathsf{td}, \mathsf{rt})$.
   (d) $(C', i^*) \notin \mathbf{CSAT}$.

Figure 14: Hybrid verifier V''.

Fix any such P^*, poly_k, poly_C, and $\mathsf{poly}_{\mathsf{P}^*}$. First, we define the hybrid trapdoor setup algorithm TGen' and verifier V' as in Figure 13 and replace TGen and V with TGen' and V' in (16). To show that the probability only decreases by a negligible amount, it suffices to show that the probability that V outputs 1 while V' outputs 0 is negligible. Since V' only differs from V in that V' outputs 0 when $x_{i^*} \neq \widetilde{x}_{i^*}$, it suffices to show that we have $\mathsf{SEH\text{-}Del.V}^{\widehat{\mathbf{x}}}(\mathsf{crs}^{\mathsf{SEH\text{-}Del}}, (h, \mathsf{rt}), \pi^{\mathsf{SEH\text{-}Del}}) = 1 \wedge x_{i^*} \neq \widetilde{x}_{i^*}$ in V' with negligible probability. Since this event indeed only occurs with negligible probability because of the partially adaptive somewhere soundness of $\mathsf{SEH\text{-}Del}$, there exists a negligible function negl_1 such that for infinitely many λ, k, n, M_C, and z as above, we have

$$\Pr\left[\begin{array}{l|l} \mathsf{V}'(\mathsf{crs}, C, \mathbf{x}, \pi, \mathsf{td}, i^*) = 1 & (\mathsf{st}, i^*, \mathbf{x}) \leftarrow \mathsf{P}_1^*(1^\lambda, 1^n, 1^{M_C}, k, z) \\ \wedge\ i^* \in [k] & (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{TGen}'(1^\lambda, 1^n, 1^{M_C}, k, i^*) \\ \wedge\ (C, x_{i^*}) \notin \mathbf{CSAT} & (C, \pi) \leftarrow \mathsf{P}_2^*(\mathsf{st}, \mathsf{crs}) \end{array}\right] \ge \frac{1}{\mathsf{poly}_{\mathsf{P}^*}(\lambda)} - \mathsf{negl}_1(\lambda) \ . \tag{17}$$

Next, we define the (inefficient) hybrid verifier V'' as in Figure 14 and replace V' with V'' in (17). To show that the probability only decreases by a negligible amount, it suffices to show that the event $(C, x_{i^*}) \notin \mathbf{CSAT} \wedge$

$x_{i^*} = \widetilde{x}_{i^*} \wedge (C', i^*) \in \mathbf{CSAT}$ only occurs with negligible probability. (Indeed, unless the event $x_{i^*} = \widetilde{x}_{i^*} \wedge (C', i^*) \in \mathbf{CSAT}$ occurs, the outputs of V' and V'' are identical.) Note that the somewhere extractability of SEH guarantees that with overwhelming probability over the choice of h (sampled in TGen'), there does not exist any certificate that successfully opens the pre-image of rt to a value other than the extracted value $\widetilde{x}_{i^*}$ in position $(n(i^*-1)+1, \ldots, n(i^*-1)+n)$. Thus, with overwhelming probability, $C'(i^*, \cdot)$ outputs 1 only when it is given $\widetilde{x}_{i^*}$ as a part of the witness, and therefore, the event $(C, x_{i^*}) \notin \mathbf{CSAT} \wedge x_{i^*} = \widetilde{x}_{i^*}$ implies $(C', i^*) \notin \mathbf{CSAT}$. Thus, there exists a negligible function negl_2 such that for infinitely many λ, k, n, M_C, and z as above, we have

$$\Pr\left[\begin{array}{l|l} \mathsf{V}''(\mathsf{crs}, C, \mathbf{x}, \pi, \mathsf{td}, i^*) = 1 & (\mathsf{st}, i^*, \mathbf{x}) \leftarrow \mathsf{P}_1^*(1^\lambda, 1^n, 1^{M_C}, k, z) \\ \wedge\, i^* \in [k] & (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{TGen}'(1^\lambda, 1^n, 1^{M_C}, k, i^*) \\ \wedge\, (C, x_{i^*}) \notin \mathbf{CSAT} & (C, \pi) \leftarrow \mathsf{P}_2^*(\mathsf{st}, \mathsf{crs}) \end{array}\right] \geq \frac{1}{\mathsf{poly}_{\mathsf{P}^*}(\lambda)} - \sum_{1 \leq i \leq 2} \mathsf{negl}_i(\lambda) \ . \tag{18}$$

Observe that $\mathsf{V}''(\mathsf{crs}, C, \mathbf{x}, \pi, \mathsf{td}, i^*) = 1$ implies that the proof π contains π^{idx} such that $\mathsf{BARG}.\mathsf{V}_{\mathrm{idx}}(\mathsf{crs}^{\mathrm{idx}}, C', \pi^{\mathrm{idx}}) = 1 \wedge (C', i^*) \notin \mathbf{CSAT}$. Thus, given (18), we can break the semi-adaptive somewhere soundness of $\mathsf{BARG}_{\mathrm{idx}}$. Thus, we obtain a contradiction. □

Remark 5. If the underlying BARG is *somewhere argument of knowledge* [CJJ22], BARG is also somewhere argument of knowledge. ♢

7 Holographic SNARG for P

In this section, we observe that we can obtain a holographic SNARG for P by combining a holographic tree-hash SNARG with a known transformation.

Theorem 5. *Assume the hardness of the LWE assumption. Then, for every Turing machine, there exists a partially adaptive public-coin non-interactive delegation scheme that satisfies the following properties.*

- *The scheme is LDE-holographic, where for the security parameter λ and an input x of length N, the encoding algorithm* Encode *outputs the LDE of x w.r.t. an arbitrary LDE parameter $(\mathbb{F}, \mathbb{H}, m)$ such that $|\mathbb{F}| \leq \mathsf{poly}(|\mathbb{H}|) \leq \mathsf{poly}(\log N)$, $|\mathbb{H}|^m \leq \mathsf{poly}(N)$, and $m|\mathbb{H}|/|\mathbb{F}| \leq O(1)$.*
- *The setup time $T_{\mathsf{Gen}}(\lambda, T, n_1, n_2)$ and the proof length $L_\pi(\lambda, T, n_1, n_2)$ are both at most $\mathsf{poly}(\lambda, \log T, \log n_1, n_2)$.*

As mentioned in Section 2.1, the above theorem is obtained by combining our LWE-based holographic tree-hash SNARG (Corollary 1) with a known transformation [Kiy22a], which uses the LWE-based *RAM delegation scheme* of Choudhuri, Jain, and Jin [CJJ22]. At a high level, the RAM delegation scheme of Choudhuri et al. is a public-coin non-interactive delegation scheme where (i) the prover can prove the correctness of an arbitrary polynomial-time computation and (ii) the verifier only needs to have the Merkle tree-hash of the computation input rather than the input itself. Naturally, their RAM delegation scheme can be converted to a holographic SNARG for P by augmenting it with a holographic tree-hash SNARG, i.e., by requiring the prover to send the tree-hash of the computation input along with a holographic proof about the correctness of the tree-hash. Thus, by using our LWE-based holographic tree-hash SNARG (Corollary 1), we can obtain a holographic SNARG for P as desired. For completeness, we describe our holographic SNARG for P in the appendix (Appendix B).

8 Application: Public-Coin Three-Round Zero-Knowledge

As an application of our holographic SNARGs, we give a public-coin 3-round zero-knowledge argument based on slightly super-polynomial hardness of the LWE assumption and keyless multi-collision-resistant hash functions.

Theorem 6. *For arbitrary super-polynomial functions $\gamma_{\mathsf{LWE}}, \gamma_{mCRH}$ and an arbitrary polynomial K, assume the γ_{LWE}-hardness of the LWE assumption and the existence of a keyless weakly (K, γ_{mCRH})-collision-resistant hash function. Then, there exists a public-coin 3-round zero-knowledge argument for NP.*

Proof. We use a known transformation to obtain a public-coin 3-round ZK argument from a tree-hash delegation scheme. Specifically, we use the transformation shown in [Kiy22a], summarized as follows.

Lemma 4. *Assume the γ_{LWE}-hardness of the LWE assumption and the existence of a keyless weakly (K, γ_{mCRH})-collision-resistant hash function as in Theorem 6. Also, assume the existence of a public-coin non-interactive tree-hash delegation scheme that satisfies the following properties.*

- *The scheme satisfies partial adaptive soundness and is γ_{Del}-secure for a super-polynomial function γ_{Del}.*
- *The scheme is LDE-holographic w.r.t. the encoding algorithm* Encode *such that, for the security parameter λ and an input x of length $N = 2^\ell \lambda$ ($\ell \leq \lfloor \log^2 \lambda \rfloor$), it outputs the LDE of x w.r.t. an LDE parameter $(\mathbb{F}, \mathbb{H}, m)$ such that $2m|\mathbb{H}| < |\mathbb{F}| = \mathsf{poly}(\log \lambda)$ and $N \leq |\mathbb{H}|^m \leq |\mathbb{F}|^m \leq \mathsf{poly}(N)$.*
- *The setup time $T_{\mathsf{Gen}}(\lambda, T, n_1, n_2)$ and the proof length $L_\pi(\lambda, T, n_1, n_2)$ are both at most $\mathsf{poly}(\lambda, \log n_1, n_2)$.*

Then, there exists a public-coin 3-round zero-knowledge argument for NP.

(For details about how we obtain Lemma 4 from [Kiy22a], see Section A.2 in the appendix.) Given Lemma 4, we can prove Theorem 6 by observing that the desired tree-hash delegation scheme can be obtained by straightforwardly adjusting the proofs of Lemma 2 and Lemma 3 (namely, by using the super-polynomial hardness of the LWE assumption rather than the standard polynomial hardness). □

References

[BFLS91] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *23rd ACM STOC*, pages 21–31. ACM Press, May 1991.

[BHK17] Zvika Brakerski, Justin Holmgren, and Yael Tauman Kalai. Non-interactive delegation and batch NP verification from standard computational assumptions. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *49th ACM STOC*, pages 474–482. ACM Press, June 2017.

[BKK+18] Saikrishna Badrinarayanan, Yael Tauman Kalai, Dakshita Khurana, Amit Sahai, and Daniel Wichs. Succinct delegation for low-space non-deterministic computation. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *50th ACM STOC*, pages 709–721. ACM Press, June 2018.

[BKP18] Nir Bitansky, Yael Tauman Kalai, and Omer Paneth. Multi-collision resistance: a paradigm for keyless hash functions. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *50th ACM STOC*, pages 671–684. ACM Press, June 2018.

[BR22] Liron Bronfman and Ron D. Rothblum. PCPs and Instance Compression from a Cryptographic Lens. In Mark Braverman, editor, *13th Innovations in Theoretical Computer Science Conference (ITCS 2022)*, volume 215 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 30:1–30:19, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *Journal of the ACM*, 51(4):557—594, July 2004.

[CHM+20] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Psi Vesely, and Nicholas P. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 738–768. Springer, Heidelberg, May 2020.

[CJJ21a] Arka Rai Choudhuri, Abhishek Jain, and Zhengzhong Jin. Non-interactive batch arguments for NP from standard assumptions. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part IV*, volume 12828 of *LNCS*, pages 394–423, Virtual Event, August 2021. Springer, Heidelberg.

[CJJ21b] Arka Rai Choudhuri, Abhishek Jain, and Zhengzhong Jin. SNARGs for $\mathcal{P}$ from LWE. Cryptology ePrint Archive, Report 2021/808, Version 20211108:181325, 2021. https://eprint.iacr.org/2021/808. An extended version of [CJJ22].

[CJJ22] Arka Rai Choudhuri, Abhishek Jain, and Zhengzhong Jin. SNARGs for $\mathcal{P}$ from LWE. In *62nd FOCS*, pages 68–79. IEEE Computer Society Press, February 2022.

[CKLR11] Kai-Min Chung, Yael Tauman Kalai, Feng-Hao Liu, and Ran Raz. Memory delegation. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 151–168. Springer, Heidelberg, August 2011.

[COS20] Alessandro Chiesa, Dev Ojha, and Nicholas Spooner. Fractal: Post-quantum and transparent recursive proofs from holography. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 769–793. Springer, Heidelberg, May 2020.

[DGKV22] Lalita Devadas, Rishab Goyal, Yael Kalai, and Vinod Vaikuntanathan. Rate-1 non-interactive arguments for batch-NP and applications. In *63rd FOCS*, pages 1057–1068. IEEE Computer Society Press, October / November 2022.

[GKR15] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: Interactive proofs for muggles. *Journal of the ACM*, 62(4):27:1–27:64, 2015.

[GR17] Tom Gur and Ron D. Rothblum. A hierarchy theorem for interactive proofs of proximity. In Christos H. Papadimitriou, editor, *ITCS 2017*, volume 4266, pages 39:1–39:43, 67, January 2017. LIPIcs.

[HJKS22] James Hulett, Ruta Jawale, Dakshita Khurana, and Akshayaram Srinivasan. SNARGs for P from sub-exponential DDH and QR. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 520–549. Springer, Heidelberg, May / June 2022.

[HLR21a] Justin Holmgren, Alex Lombardi, and Ron D. Rothblum. Fiat-Shamir via list-recoverable codes (or: Parallel repetition of GMW is not zero-knowledge). Cryptology ePrint Archive, Report 2021/286, Version 20210307:022349, 2021. https://eprint.iacr.org/2021/286. An extended version of [HLR21b].

[HLR21b] Justin Holmgren, Alex Lombardi, and Ron D. Rothblum. Fiat–Shamir via list-recoverable codes (or: parallel repetition of GMW is not zero-knowledge). In Samir Khuller and Virginia Vassilevska Williams, editors, *53rd ACM STOC*, page 750–760. ACM Press, June 2021.

[HR04] Chun-Yuan Hsiao and Leonid Reyzin. Finding collisions on a public road, or do secure hash functions need secret coins? In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 92–105. Springer, Heidelberg, August 2004.

[HR18] Justin Holmgren and Ron Rothblum. Delegating computations with (almost) minimal time and space overhead. In Mikkel Thorup, editor, *59th FOCS*, pages 124–135. IEEE Computer Society Press, October 2018.

[HW15] Pavel Hubacek and Daniel Wichs. On the communication complexity of secure function evaluation with long output. In Tim Roughgarden, editor, *ITCS 2015*, pages 163–172. ACM, January 2015.

[JJ22] Abhishek Jain and Zhengzhong Jin. Indistinguishability obfuscation via mathematical proofs of equivalence. In *63rd FOCS*, pages 1023–1034. IEEE Computer Society Press, 2022.

[JKKZ21] Ruta Jawale, Yael Tauman Kalai, Dakshita Khurana, and Rachel Yun Zhang. SNARGs for bounded depth computations and PPAD hardness from sub-exponential LWE. In Samir Khuller and Virginia Vassilevska Williams, editors, *53rd ACM STOC*, pages 708–721. ACM Press, June 2021.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *24th ACM STOC*, pages 723–732. ACM Press, May 1992.

[Kiy22a] Susumu Kiyoshima. Public-coin 3-round zero-knowledge from learning with errors and keyless multi-collision-resistant hash. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part I*, volume 13507 of *LNCS*, pages 444–473. Springer, Heidelberg, August 2022.

[Kiy22b] Susumu Kiyoshima. Public-coin 3-round zero-knowledge from learning with errors and keyless multi-collision-resistant hash. Cryptology ePrint Archive, Report 2022/820, Version 20220624:060657, 2022. https://eprint.iacr.org/2022/820. An extended version of [Kiy22a].

[KP16] Yael Tauman Kalai and Omer Paneth. Delegating RAM computations. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 91–118. Springer, Heidelberg, October / November 2016.

[KPY19a] Yael Kalai, Omer Paneth, and Lisa Yang. How to delegate computations publicly. Cryptology ePrint Archive, Report 2019/603, Version 20190602:113205, 2019. https://eprint.iacr.org/2019/603. An extended version of [KPY19b].

[KPY19b] Yael Tauman Kalai, Omer Paneth, and Lisa Yang. How to delegate computations publicly. In Moses Charikar and Edith Cohen, editors, *51st ACM STOC*, pages 1115–1124. ACM Press, June 2019.

[KR15] Yael Tauman Kalai and Ron D. Rothblum. Arguments of proximity - [extended abstract]. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 422–442. Springer, Heidelberg, August 2015.

[KRR22] Yael Tauman Kalai, Ran Raz, and Ron D Rothblum. How to Delegate Computations: The Power of No-Signaling Proofs. *Journal of the ACM*, 69(1):1–82, February 2022.

[KVZ21] Yael Tauman Kalai, Vinod Vaikuntanathan, and Rachel Yun Zhang. Somewhere statistical soundness, post-quantum security, and SNARGs. In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part I*, volume 13042 of *LNCS*, pages 330–368. Springer, Heidelberg, November 2021.

[PP22] Omer Paneth and Rafael Pass. Incrementally verifiable computation via rate-1 batch arguments. In *63rd FOCS*, pages 1045–1056. IEEE Computer Society Press, October / November 2022.

[PS19] Chris Peikert and Sina Shiehian. Noninteractive zero knowledge for NP from (plain) learning with errors. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 89–114. Springer, Heidelberg, August 2019.

[RVW13] Guy N. Rothblum, Salil P. Vadhan, and Avi Wigderson. Interactive proofs of proximity: delegating computation in sublinear time. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 793–802. ACM Press, June 2013.

[WW22] Brent Waters and David J. Wu. Batch arguments for NP and more from standard bilinear group assumptions. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 433–463. Springer, Heidelberg, August 2022.

A Additional Remarks

A.1 Remark About Theorem 1

Remark 6 (On the condition on t_λ in Theorem 1). As mentioned in [HLR21a, Section 5.1], the condition $t_\lambda \geq \lambda/\log(1/\rho(\lambda))$ in Theorem 1 can be weakened to $t_\lambda \geq \lambda^\delta/\log(1/\rho(\lambda))$ for an arbitrarily small constant $\delta > 0$. Roughly speaking, this can be observed as follows. At a high level, the correlation-intractable hash function

family $\mathcal{H}$ of Theorem 1 is obtained by combining an information-theoretical object[19] with a hash function family $\mathcal{H}'$ that is correlation intractable for *T'-time searchable relation ensembles* for a sufficiently large polynomial upper bound T'. (A relation ensemble $\mathbf{R}' = \{\mathbf{R}'_\lambda\}_{\lambda\in\mathbb{N}}$ is T'-time searchable if for every $\lambda \in \mathbb{N}$, there exists a circuit C_λ of size $T'(\lambda)$ such that if $(x, y) \in \mathbf{R}'_\lambda$ then $y = C_\lambda(x)$.) Now, a more accurate description of the condition for t_λ is $t_\lambda \geq k^2 \log Q/\log(1/\rho(\lambda))$, where k is the output length of $\mathcal{H}'$ and $Q \leq (k|Y_\lambda|\rho(\lambda))^4 = \mathsf{poly}(k, \lambda)$ (cf. [HLR21a, Proof of Lemma 5.1]). Thus, if the output length of $\mathcal{H}'$ is sufficiently short, it suffices to have $t_\lambda \geq \lambda^\delta/\log(1/\rho(\lambda))$. Fortunately, in [HLR21b], the hash function family $\mathcal{H}'$ is instantiated by the LWE-based correlation-intractable hash family of Peikert and Shiehian [PS19], and its output length can be shortened to $\lambda^{\delta'}$ for an arbitrarily small constant $\delta' > 0$ by appropriately setting its parameters (e.g., by using it with a scaled-down security parameter).

If the LWE assumption holds against super-polynomial-time adversaries, the condition can be further weakened to $t_\lambda \geq \lambda^{1/\tau(\lambda)}/\log(1/\rho(\lambda))$ for a super-constant function $\tau(\lambda) = \omega(1)$. To see this, let us assume the T_{LWE}-hardness of the LWE problem for an arbitrarily small super-polynomial $T_{\mathrm{LWE}}(\lambda) = \lambda^{\tau'(\lambda)}$. Then, when the hash function $\mathcal{H}'$ in the above is used with an appropriate parameter setting (in particular with a scaled-down security parameter $\lambda^{1/\tau'(\lambda)}$), its output length is shortened to $\mathsf{poly}(\lambda^{1/\tau'(\lambda)})$, and its correlation extractability holds against $\mathsf{poly}(\lambda)$-time adversaries for $T'(\lambda)$-time searchable relations, where T' is an arbitrary predetermined polynomial upper bound.[20] Thus, there exists a super-constant function τ such that for t_λ to satisfy the condition $t_\lambda \geq k^2 \log Q/\log(1/\rho(\lambda))$ in the above, it suffices to have $t_\lambda \geq \lambda^{1/\tau(\lambda)}/\log(1/\rho(\lambda))$. ♢

A.2 Remark About Lemma 4

Remark 7. Concretely speaking, Lemma 4 is obtained by combining the proofs of [Kiy22b, Lemma 1, Lemma 6, Lemma7, Theorem 3], where a tree-hash delegation scheme as described in Lemma 4 is gradually upgraded to stronger delegation schemes and then used to obtain a public-coin 3-round ZK argument. (Although each of the proofs of [Kiy22b, Lemma 1, Lemma 6, Lemma7, Theorem 3] is given assuming the sub-exponential hardness of the LWE assumption, the sub-exponential hardness is only used to obtain a desired tree-hash delegation scheme and thus slightly super-polynomial hardness is sufficient in the other parts of the proofs.[21]) As a minor detail, we note that holographic tree-hash delegation schemes in [Kiy22a] are formulated based on a different definition than ours. However, schemes under our definition (Definition 20) trivially satisfy the definition of [Kiy22a] (see Appendix C for details). ♢

B Details About Holographic SNARG for P

We describe the holographic SNARG for P that is sketched in Section 7. (This section is largely taken from [Kiy22b, Section 5].)

B.1 Preliminary: RAM Delegation

We recall the definition of publicly verifiable non-interactive RAM delegation schemes from [KPY19b, CJJ22].

A RAM machine R with word size ℓ is modeled as a deterministic machine with random access to a memory of at most 2^ℓ bits and a local state of $O(\ell)$ bits. At every step, the machine reads or writes a single memory bit and updates its state. For simplicity, we use the security parameter λ as the word size. Also, for convenience, we consider a slightly more general model than [KPY19b, CJJ22] and think of a RAM machine that has access to a memory of at most 2^ℓ bits and additionally takes a short input. (In [KPY19b, CJJ22], a RAM machine has access to a memory of 2^ℓ bits and takes no input other than the memory and the initial local state.) In this paper,

[19] Namely, a family of list-recoverable codes.

[20] The output length of $\mathcal{H}'$ (when instantiated by the LWE-based correlation-intractable hash family of Peikert and Shiehian [PS19]) is independent of T' (cf. [PS19, Proof of Theorem 7]). Thus, the output length of $\mathcal{H}'$ can be $\lambda^{1/\omega(1)}$ even when the correlation intractability is required to hold for $\mathsf{poly}(\lambda)$-time searchable relations.

[21] More precisely, the sub-exponential hardness is also used to obtain a logarithmic-depth collision-resistant hash function family since the tree-hash delegation scheme in [Kiy22a] is only shown to work for such hash function families. Since by default we assume that tree-hash delegation schemes work for arbitrary hash functions (and our tree-hash delegation scheme in Corollary 1 indeed works for arbitrary hash functions), the sub-exponential hardness is no longer needed for this purpose.

the memory and state of a RAM machine at a given time-step are referred to as its *memory-state pair*.[22] For any RAM machine R, let U_R denote the language such that $(\ell, x, \mathsf{ms}, \mathsf{ms}', T) \in U_R$ iff R with word size ℓ and on input x transitions from memory-state pair ms to memory-state pair ms' in T steps.

Definition 21. *For any RAM machine R, a **publicly verifiable non-interactive RAM delegation scheme** for R consists of four algorithms $(\mathsf{Gen}, \mathsf{Mem}, \mathsf{Prove}, \mathsf{Ver})$ that have the following syntax.*

Syntax.

- $(\mathsf{pk}, \mathsf{vk}, \mathsf{dk}) \leftarrow \mathsf{Gen}(1^\lambda, T)$*:* Gen *is a probabilistic algorithm that takes as input a security parameter 1^λ and a time bound T, and it outputs a triple of public keys: a prover key* pk*, a verifier key* vk*, and a digest key* dk*.*
- $\mathsf{digest} := \mathsf{Mem}(\mathsf{dk}, \mathsf{ms})$*:* Mem *is a deterministic algorithm that takes as input a digest key* dk *and a memory-state pair* ms*, and it outputs a digest* digest *of the memory-state pair.*
- $\pi := \mathsf{Prove}(\mathsf{pk}, x, \mathsf{ms}, \mathsf{ms}')$*:* Prove *is a deterministic algorithm that takes as input a prover key* pk*, an input x to R, source and destination memory-state pairs* $\mathsf{ms}, \mathsf{ms}'$*, and it outputs a proof π.*
- $b := \mathsf{Ver}(\mathsf{vk}, x, \mathsf{digest}, \mathsf{digest}', \pi)$*:* Ver *is a deterministic algorithm that takes as input a verifier key* vk*, an input x to R, source and destination digests* $\mathsf{digest}, \mathsf{digest}'$*, and a proof π, and it outputs a bit b.*

Efficiency. *For any functions $T_{\mathsf{Gen}} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ and $L_\pi : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, a publicly verifiable non-interactive RAM delegation scheme is said to have **setup time** T_{Gen} **and proof length** L_π if for every $\lambda, T \in \mathbb{N}$ such that $T \leq 2^\lambda$ and for every $x, \mathsf{ms}, \mathsf{ms}' \in \{0,1\}^*$ such that $(\lambda, x, \mathsf{ms}, \mathsf{ms}', T) \in U_R$:*

- $\mathsf{Gen}(1^\lambda, T)$ *runs in time $T_{\mathsf{Gen}}(\lambda, T)$.*
- $\mathsf{Mem}(\mathsf{dk}, \mathsf{ms})$ *runs in time $|\mathsf{ms}| \cdot \mathsf{poly}(\lambda)$ and outputs a digest of length λ.*
- $\mathsf{Prove}(\mathsf{pk}, x, \mathsf{ms}, \mathsf{ms}')$ *runs in time $\mathsf{poly}(\lambda, T, |x|, |\mathsf{ms}|)$ and outputs a proof of length $L_\pi(\lambda, T, |x|)$.*
- $\mathsf{Ver}(\mathsf{vk}, x, \mathsf{digest}, \mathsf{digest}', \pi)$ *runs in time $O(L_\pi(\lambda, T, |x|)) + \mathsf{poly}(\lambda, |x|)$.*

Security. *A publicly verifiable non-interactive RAM delegation scheme is called **sound** if it satisfies the following.*

- ***Correctness.*** *For every $\lambda, T \in \mathbb{N}$ such that $T \leq 2^\lambda$ and for every $x, \mathsf{ms}, \mathsf{ms}' \in \{0,1\}^*$ such that $(\lambda, x, \mathsf{ms}, \mathsf{ms}', T) \in U_R$,*

$$\Pr\left[\mathsf{Ver}(\mathsf{vk}, x, \mathsf{digest}, \mathsf{digest}', \pi) = 1 \;\middle|\; \begin{array}{l}(\mathsf{pk}, \mathsf{vk}, \mathsf{dk}) \leftarrow \mathsf{Gen}(1^\lambda, T) \\ \mathsf{digest} := \mathsf{Mem}(\mathsf{dk}, \mathsf{ms}) \\ \mathsf{digest}' := \mathsf{Mem}(\mathsf{dk}, \mathsf{ms}') \\ \pi := \mathsf{Prove}(\mathsf{pk}, x, \mathsf{ms}, \mathsf{ms}')\end{array}\right] = 1 \ .$$

- ***Collision resistance.*** *For every PPT algorithm $\mathcal{A}$ and polynomial poly_T, there exists a negligible function negl such that for every $\lambda \in \mathbb{N}$, $T \leq \mathsf{poly}_T(\lambda)$, and $z \in \{0,1\}^*$,*

$$\Pr\left[\begin{array}{l}\mathsf{ms} \neq \mathsf{ms}' \\ \wedge\, \mathsf{Mem}(\mathsf{dk}, \mathsf{ms}) = \mathsf{Mem}(\mathsf{dk}, \mathsf{ms}')\end{array} \;\middle|\; \begin{array}{l}(\mathsf{pk}, \mathsf{vk}, \mathsf{dk}) \leftarrow \mathsf{Gen}(1^\lambda, T) \\ (\mathsf{ms}, \mathsf{ms}') \leftarrow \mathcal{A}(\mathsf{pk}, \mathsf{vk}, \mathsf{dk}, z)\end{array}\right] \leq \mathsf{negl}(\lambda) \ .$$

- ***Soundness.*** *For every PPT algorithm $\mathcal{A}$ and polynomial poly_T, there exists a negligible function negl such that for every $\lambda \in \mathbb{N}$, $T \leq \mathsf{poly}_T(\lambda)$, and $z \in \{0,1\}^*$,*

$$\Pr\left[\begin{array}{l}\mathsf{Ver}(\mathsf{vk}, x, \mathsf{digest}, \mathsf{digest}', \pi) = 1 \\ \wedge\, (\lambda, x, \mathsf{ms}, \mathsf{ms}', T) \in U_R \\ \wedge\, \mathsf{digest} = \mathsf{Mem}(\mathsf{dk}, \mathsf{ms}) \\ \wedge\, \mathsf{digest}' \neq \mathsf{Mem}(\mathsf{dk}, \mathsf{ms}')\end{array} \;\middle|\; \begin{array}{l}(\mathsf{pk}, \mathsf{vk}, \mathsf{dk}) \leftarrow \mathsf{Gen}(1^\lambda, T) \\ (x, \mathsf{ms}, \mathsf{ms}', \mathsf{digest}, \mathsf{digest}', \pi) \leftarrow \mathcal{A}(\mathsf{pk}, \mathsf{vk}, \mathsf{dk}, z)\end{array}\right] \leq \mathsf{negl}(\lambda) \ .$$

[22] Unlike [KPY19b, CJJ22], we refrain from using the term "configuration" to refer to the memory and state since we allow RAM machines to additionally have inputs.

A publicly verifiable non-interactive RAM delegation scheme is called ***public-coin*** *if the setup algorithm* Gen *is public-coin, i.e., it just outputs a triple of strings that are sampled uniformly randomly.*

As in [Kiy22a], the following prior result [CJJ22] (with straightforward adaptation) is used in this work.

Theorem 7. *Let R be any RAM machine. Under the hardness of the LWE assumption, there exists a publicly verifiable non-interactive RAM delegation scheme for R with setup time $T_{\mathsf{Gen}}(\lambda, T) = \mathsf{poly}(\lambda, \log T)$ and proof length $L_\pi(\lambda, T, |x|) = \mathsf{poly}(\lambda, \log T, |x|)$. Furthermore, this scheme is public-coin, and (i) the setup algorithm* Gen *outputs a hash function as a digest key, and (ii) the digest algorithm* Mem*, on input a digest key* dk *and a memory-state pair* $\mathsf{ms} = (\mathsf{DB}, \mathsf{st})$*, outputs a triple* $\mathsf{digest} = (\mathsf{st}, \mathsf{rt}, |\mathsf{DB}|)$ *that consists of the local state* st*, the tree-hash* $\mathsf{rt} := \mathsf{TreeHash}_{\mathsf{dk}}(\mathsf{DB})$ *of the memory* DB*, and the memory length* $|\mathsf{DB}|$.

Two remarks about Theorem 7 are given below.

1. The first part of Theorem 7 differs from what is shown in [CJJ22] in that RAM machines are defined in a model where a RAM machine has access to a memory of at most 2^ℓ bits (rather than exactly 2^ℓ bits) and takes a short input. Still, the first part of Theorem 7 can be easily obtained from [CJJ22]. In particular, the analysis given in [CJJ22] can be easily extended for memories of at most 2^ℓ bits by appending the length $|\mathsf{DB}|$ of the memory to the digest digest so that the verification algorithm Verify can learn $|\mathsf{DB}|$ and (ii) for RAM machines that take additional short inputs by allowing the proof length to be polynomial in the input length (but still polylogarithmic in the computation-time bound T).[23]

2. Regarding the furthermore part of Theorem 7, the public-coin property is implicitly mentioned in [CJJ22].[24] The properties of Gen and Mem can be easily verified by inspecting the scheme description in [CJJ21b, Figure 5].[25]

B.2 Holographic SNARG for P

Fix any two-input Turing machine M. Let R be the following RAM machine.

- R is given as input a string χ_2 and given as memory a string χ_1. Then, R internally executes $M(\chi_1, \chi_2)$ while using the memory χ_1 to emulate the working tape of M. (We assume that χ_1 contains a padding string that can be used to emulate the working tape, and M is designed to ignore the padding part.) When M terminates, R writes (b, t) at the beginning of the memory and terminates, where b is the output of M and t is the running time of M.

Without loss of generality, we assume that there exists a (non-decreasing) polynomial poly_R such that when the running time of $M(\chi_1, \chi_2)$ is t, the running time of $R^{\chi_1}(\chi_2)$ is $\mathsf{poly}_R(t)$, and $R^{\chi_1}(\chi_2)$ only reads and writes the first $\mathsf{poly}_R(t)$ bits of χ_1. (Hence, we assume that the length of χ_1, including the padding part, is at most $\mathsf{poly}_R(t)$.) Let $\mathsf{RDel} = (\mathsf{RDel.Gen}, \mathsf{RDel.Mem}, \mathsf{RDel.P}, \mathsf{RDel.V})$ be the public-coin non-interactive RAM delegation scheme given by Theorem 7 for the RAM machine R. Recall that $\mathsf{RDel.Gen}$ outputs as a digest key a hash function that is sampled from a collision-resistant hash family. Let $\mathsf{TH\text{-}Del} = (\mathsf{TH\text{-}Del.Gen}, \mathsf{TH\text{-}Del.P}, \mathsf{TH\text{-}Del.V}, \mathsf{TH\text{-}Del.Encode})$ be the public-coin non-interactive tree-hash delegation scheme given by Corollary 1.

Our publicly verifiable holographic non-interactive delegation scheme for M is described in Figure 15. The correctness and efficiency can be verified by inspection. The partially adaptive soundness can be proved by following the soundness proof given in [Kiy22b, Section 5]. Namely, it is first argued that because of the partially adaptive soundness of $\mathsf{TM\text{-}Del}$, if a cheating prover successfully provides an accepting proof for a false statement $((\chi_1, \chi_2), T)$, the proof must contain the correct tree-hash rt of χ_1, implying that $\mathsf{digest} = (\mathsf{st}_{\mathrm{START}}, \mathsf{rt}, |\chi_1|)$ is the digest of the initial memory-state pair of $R^{\chi_1}(\chi_2)$. On the other hand, since the statement $((\chi_1, \chi_2), T)$ is false (implying that either the RAM machine $R^{\chi_1}(\chi_2)$ does not terminate at step $T_R = \mathsf{poly}_R(T)$ or it terminates

[23] For those familiar with the RAM delegation of [CJJ22], we note that we allow the statements of the Batch-NP argument to contain the input of the RAM machine.

[24] Technically, the public-coin property can be verified by observing that under the LWE assumption, all the components of the scheme of [CJJ22] can be made public-coin by using, e.g., an FHE scheme with pseudorandom public keys and ciphertexts.

[25] Actually, Mem in [CJJ21b, Figure 5] outputs a pair $\mathsf{digest} = (\mathsf{st}, \mathsf{rt})$, but as noted above, we consider an extended version that additionally includes $|\mathsf{DB}|$ in digest.

$\underline{\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda, T, n_1, n_2)\text{:}}$

1. Sample $(\mathsf{pk}, \mathsf{vk}, \mathsf{dk}) \leftarrow \mathsf{RDel.Gen}(1^\lambda, T_R)$, where $T_R := \mathsf{poly}_R(T)$.
2. Sample $\mathsf{crs}^{\mathsf{TH\text{-}Del}} \leftarrow \mathsf{TH\text{-}Del.Gen}(1^\lambda, T^{\mathsf{TH\text{-}Del}}, n_1, |\mathsf{dk}| + \lambda)$, where $T^{\mathsf{TH\text{-}Del}} := T_{\text{tree-hash}}(n_1, |\mathsf{dk}| + \lambda)$ for the function $T_{\text{tree-hash}}$ in Definition 20.
3. Output $\mathsf{crs} = (1^\lambda, T, \mathsf{crs}^{\mathsf{TH\text{-}Del}}, \mathsf{pk}, \mathsf{vk}, \mathsf{dk})$.

$\underline{\pi := \mathsf{P}(\mathsf{crs}, (\chi_1, \chi_2))\text{:}}$

1. Parse crs as $(1^\lambda, T, \mathsf{crs}^{\mathsf{TH\text{-}Del}}, \mathsf{pk}, \mathsf{vk}, \mathsf{dk})$. Let $T_R := \mathsf{poly}_R(T)$.
2. Run $R^{\chi_1}(\chi_2)$. If $R^{\chi_1}(\chi_2)$ does not terminate in T_R steps, abort. Otherwise, let χ_1' denote the content of the memory at the termination of $R^{\chi_1}(\chi_2)$.
3. Compute $\pi^{\mathsf{RDel}} := \mathsf{RDel.P}(\mathsf{pk}, \chi_2, \mathsf{ms}, \mathsf{ms}')$ for $\mathsf{ms} := (\chi_1, \mathsf{st}_{\text{START}})$ and $\mathsf{ms}' := (\chi_1', \mathsf{st}_{\text{END}})$, where $\mathsf{st}_{\text{START}}$ and st_{END} are the initial and the terminating states of R.
4. Compute $\pi^{\mathsf{TH\text{-}Del}} := \mathsf{TH\text{-}Del.P}(\mathsf{crs}, (\chi_1, (\mathsf{dk}, \mathsf{rt}))$ for $\mathsf{rt} := \mathsf{TreeHash}_{\mathsf{dk}}(\chi_1)$.
5. Let (b', t') be the prefix of χ_1' that R wrote at the time of its termination, rt' be the tree-hash that is obtained by $\mathsf{rt}' := \mathsf{TreeHash}_{\mathsf{dk}}(\chi_1')$, and $\mathsf{cert}^{\mathsf{TreeHash}}$ be the local opening for (b', t') w.r.t. rt'.
6. Output $\pi := (\mathsf{rt}, \mathsf{rt}', (b', t'), \pi^{\mathsf{RDel}}, \pi^{\mathsf{TH\text{-}Del}}, \mathsf{cert}^{\mathsf{TreeHash}})$.

$\underline{b := \mathsf{V}(\mathsf{crs}, (\chi_1, \chi_2), \pi)\text{:}}$

1. Parse crs as $(1^\lambda, T, \mathsf{crs}^{\mathsf{TH\text{-}Del}}, \mathsf{pk}, \mathsf{vk}, \mathsf{dk})$ and π as $(\mathsf{rt}, \mathsf{rt}', (b', t'), \pi^{\mathsf{RDel}}, \pi^{\mathsf{TH\text{-}Del}}, \mathsf{cert}^{\mathsf{TreeHash}})$. Let $T_R := \mathsf{poly}_R(T)$, $\mathsf{digest} := (\mathsf{st}_{\text{START}}, \mathsf{rt}, |\chi_1|)$ and $\mathsf{digest}' := (\mathsf{st}_{\text{END}}, \mathsf{rt}', |\chi_1|)$.
2. Output 1 if all of the following hold.
   (a) $b' = 1$ and $t' \leq T$.
   (b) $\mathsf{RDel.V}(\mathsf{vk}, \chi_2, \mathsf{digest}, \mathsf{digest}', \pi^{\mathsf{RDel}}) = 1$.
   (c) $\mathsf{cert}^{\mathsf{TreeHash}}$ is a valid local opening for (b', t') w.r.t. rt'.
   (d) $\mathsf{TH\text{-}Del.V}(\mathsf{crs}, (\chi_1, (\mathsf{dk}, \mathsf{rt})), \pi^{\mathsf{TH\text{-}Del}}) = 1$.

Figure 15: Our publicly verifiable non-interactive delegation scheme for M.

with a memory content $((b, t), \ldots)$ such that $b = 0$ or $t > T$), the destination memory hash rt' in the proof cannot be correct, i.e., $\mathsf{digest}' := (\mathsf{st}_{\text{END}}, \mathsf{rt}', |\chi_1|)$ cannot be the digest of the memory-state pair that $R^{\chi_1}(\chi_2)$ has at step T_R. These two imply that the proof that the cheating prover provides must contain a proof that breaks the soundness of RDel, and thus, we obtain a contradiction. Since the formal proof is quite similar to that given in [Kiy22b, Section 5], the formal proof is omitted.

C Details About the Definition of Holographic Tree-Hash Delegation in [Kiy22a]

We provide necessary details about *publicly verifiable weak tree-hash oracle memory delegation schemes*, a different formulation of holographic tree-hash delegation sachems that is given in [Kiy22a] and mentioned in Section 8.

C.1 Definition

First, we recall the definition of publicly verifiable weak tree-hash oracle memory delegation schemes. The following definition is obtained by combining [Kiy22b, Definition 8, Definition 9, Lemma 2].

Definition 22. *Let $\mathcal{H}$ be a hash function family such that each $h \in \mathcal{H}_\lambda$ is a length-halving hash function $h : \{0,1\}^{2\lambda} \to \{0,1\}^\lambda$. Then, a **publicly verifiable (2-round) weak tree-hash oracle memory delegation***

***scheme** for $\mathcal{H}$ consists of five algorithms* $(\mathsf{Mem}, \mathsf{Query}_1, \mathsf{Prove}, \mathsf{Query}_2, \mathsf{Ver})$ *that have the following syntax and efficiency.*

Syntax.

- $\widehat{\mathsf{DB}} \coloneqq \mathsf{Mem}(1^\lambda, \mathsf{DB})$*:* Mem *is a deterministic polynomial-time algorithm that takes as input a security parameter λ (in unary) and a memory* DB*, and it outputs an encoding* $\widehat{\mathsf{DB}}$ *of the memory.*
- $(q, \sigma) \leftarrow \mathsf{Query}_1(1^\lambda)$*:* Query_1 *is a probabilistic polynomial-time algorithm that takes as input a security parameter λ (in unary), and it outputs a query q and a random string σ.*
- $\pi \coloneqq \mathsf{Prove}(\mathsf{DB}, (M, t, y), q)$*:* Prove *is a deterministic algorithm that takes as input a memory* DB*, a deterministic Turing machine M (possibly with some hardwired inputs), a time bound t, an output y, and a query q, and it outputs a proof π.*
- $I \coloneqq \mathsf{Query}_2(L_{\mathsf{DB}}, \sigma, \pi)$*:* Query_2 *is a deterministic algorithm that takes as input a length parameter L_{DB}, a random string σ, and a proof π, and it outputs a set $I \subseteq \mathbb{N}$ of oracle queries.*
- $b \coloneqq \mathsf{Ver}^{(\cdot)}(L_{\mathsf{DB}}, (M, t, y), q, \sigma, \pi)$*:* Ver *is a deterministic oracle algorithm that takes as input a length parameter L_{DB}, a deterministic Turing machine M (possibly with some hardwired inputs), a time bound t, an output y, a query q, a random string σ, and a proof π, and it outputs a bit b.*

Efficiency. *For any polynomial p, there exists polynomials $\mathsf{poly}_P, \mathsf{poly}_V$ such that for every $\lambda \in \mathbb{N}$, $(M, t, y) \in \{0,1\}^{p(\lambda)}$, and $\mathsf{DB} \in \{0,1\}^*$ such that $M(\mathsf{DB})$ outputs y within t steps and $|\mathsf{DB}| \leq t \leq \lambda^{\log \lambda}$,*

- $\mathsf{Prove}(\mathsf{DB}, (M, t, y), q)$ *runs in time* $\mathsf{poly}_P(\lambda, t)$*, and*
- $\mathsf{Ver}^{(\cdot)}(|\mathsf{DB}|, (M, t, y), q, \sigma, \pi)$ *runs in time* $\mathsf{poly}_V(\lambda)$*.*

Security. *A publicly verifiable 2-round tree-hash oracle memory delegation scheme is required to satisfy the following.*

- ***Correctness.** For any hash function $h : \{0,1\}^{2\lambda} \to \{0,1\}^\lambda$, let M_h be a Turing machine that takes as input a string $\mathsf{DB} \in \{0,1\}^{2^\ell \lambda}$ ($\ell \in \mathbb{N}$) and outputs $\mathsf{TreeHash}_h(\mathsf{DB})$. Then, for every $\lambda \in \mathbb{N}$, $h \in \mathcal{H}_\lambda$, $(M_h, t, y) \in \{0,1\}^{\mathsf{poly}(\lambda)}$, and $\mathsf{DB} \in \{0,1\}^{2^\ell \lambda}$ ($\ell \in \mathbb{N}$) such that $M_h(\mathsf{DB})$ outputs y within t steps and $|\mathsf{DB}| \leq t \leq 2^\lambda$,*

$$\Pr\left[\mathsf{Ver}^{\widehat{\mathsf{DB}}|_I}(|\mathsf{DB}|, (M_h, t, y), q, \sigma, \pi) = 1 \;\middle|\; \begin{array}{l} \widehat{\mathsf{DB}} \coloneqq \mathsf{Mem}(1^\lambda, \mathsf{DB}) \\ (q, \sigma) \leftarrow \mathsf{Query}_1(1^\lambda) \\ \pi \coloneqq \mathsf{Prove}(\mathsf{DB}, (M_h, t, y), q) \\ I \coloneqq \mathsf{Query}_2(|\mathsf{DB}|, \sigma, \pi) \end{array}\right] = 1 \ .$$

- ***Weak soundness.** There exist a deterministic polynomial-time algorithm* Decode *and a predicate* Valid *such that for every pair of* PPT *adversaries $(\mathcal{A}_1, \mathcal{A}_2)$ and every polynomial-size advice $\{z_\lambda\}_{\lambda \in \mathbb{N}}$, there exists a negligible function* negl *such that for every $\lambda \in \mathbb{N}$ and $h \in \mathcal{H}_\lambda$,*

$$\Pr\left[\begin{array}{l} \mathsf{rt} \neq \mathsf{TreeHash}_h(\widetilde{\mathsf{DB}}) \\ \wedge\ \mathsf{Ver}^{\widehat{\mathsf{DB}}|_I}(L_{\mathsf{DB}}, (M_h, t_{L_{\mathsf{DB}}}, \mathsf{rt}), q, \sigma, \pi) = 1 \\ \wedge\ \mathsf{Valid}(\widehat{\mathsf{DB}}, L_{\mathsf{DB}}) = 1 \end{array} \;\middle|\; \begin{array}{l} (\widehat{\mathsf{DB}}, L_{\mathsf{DB}}, \mathsf{st}) \leftarrow \mathcal{A}_1(1^\lambda, z_\lambda) \\ (q, \sigma) \leftarrow \mathsf{Query}_1(1^\lambda) \\ (\mathsf{rt}, \pi) \leftarrow \mathcal{A}_2(h, q, \sigma, \mathsf{st}) \\ I \coloneqq \mathsf{Query}_2(L_{\mathsf{DB}}, \sigma, \pi) \\ \widetilde{\mathsf{DB}} \leftarrow \mathsf{Decode}(\widehat{\mathsf{DB}}, L_{\mathsf{DB}}) \end{array}\right] \leq \mathsf{negl}(\lambda) \ ,$$

  where $t_{L_{\mathsf{DB}}}$ is the running time of M_h for inputs of length L_{DB}, and $\mathsf{Decode}(\cdot, L_{\mathsf{DB}})$ always outputs an L_{DB}-bit string (or $\perp$).

*A publicly verifiable 2-round oracle memory delegation scheme is called **public-coin** if the query algorithm* Query_1 *is public-coin, i.e., it just outputs a string that is sampled uniformly randomly.*

$\widehat{\mathsf{DB}} := \mathsf{Mem}(1^\lambda, \mathsf{DB})$:

1. Output $\widehat{\mathsf{DB}} := \mathsf{Encode}(\lambda, \mathsf{DB})$

$(q, \sigma) \leftarrow \mathsf{Query}_1(1^\lambda)$:

1. For each $i \in [\lambda]$, run $\mathsf{crs}_i \leftarrow \mathsf{Gen}(1^\lambda, T_i, n_{i,1}, n_{i,2})$ for $T_i := \mathsf{poly}_T(2^i\lambda)$, $n_{i,1} := 2^i\lambda$, and $n_{i,2} := \mathsf{poly}_h(\lambda)+\lambda$.
2. Output $(q, \sigma) := (\{\mathsf{crs}_i\}_{i\in[\lambda]}, \bot)$.

$\pi := \mathsf{Prove}(\mathsf{DB}, (M_h, t, y), q)$:

1. Parse q as $\{\mathsf{crs}_i\}_{i\in[\lambda]}$ and find i^* such that $|\mathsf{DB}| = 2^{i^*}\lambda$.
2. Compute $\pi' := \mathsf{P}(\mathsf{crs}_{i^*}, (\mathsf{DB}, (h, y)))$.
3. Run the first part of the holographic verifier $\mathsf{V}(\mathsf{crs}_{i^*}, (|\mathsf{DB}|, (h, y)), \pi')$ to obtain a query set I (cf. Definition 16).
4. Output $\pi := (\pi', I)$.

$I := \mathsf{Query}_2(L_{\mathsf{DB}}, \sigma, \pi)$:

1. Parse π as (π', I) and output I.

$b := \mathsf{Ver}^{\widehat{\mathsf{DB}}|_I}(L_{\mathsf{DB}}, (M_h, t, y), q, \sigma, \pi)$:

1. Parse q as $\{\mathsf{crs}_i\}_{i\in[\lambda]}$ and π as (π', I). Also, find i^* such that $L_{\mathsf{DB}} = 2^{i^*}\lambda$.
2. Run the first part of the holographic verifier $\mathsf{V}(\mathsf{crs}_{i^*}, (L_{\mathsf{DB}}, (h, y)), \pi')$. If the verifier immediately outputs 0 or the query set that the verifier outputs is not equal to I, output 0 and terminate.
3. Output 1 iff $\widehat{\mathsf{DB}}|_I = Z$, where Z is the set of expected responses that the first part of the holographic verifier outputs along with I (cf. Definition 16).

Figure 16: Publicly verifiable weak tree-hash oracle memory delegation scheme.

C.2 Relation with Publicly Verifiable Non-Interactive Tree-Hash Delegation Schemes

Next, we observe that partially adaptive publicly verifiable LDE-holographic non-interactive tree-hash delegation schemes (Definition 20) can straightforwardly be viewed as publicly verifiable weak tree-hash oracle memory delegation schemes (Definition 22). Let $\mathsf{TH\text{-}Del} = (\mathsf{Gen}, \mathsf{P}, \mathsf{V})$ be any partially adaptive publicly verifiable LDE-holographic non-interactive tree-hash delegation scheme with setup time $T_{\mathsf{Gen}}(\lambda, T, n_1, n_2) = \mathsf{poly}(\lambda, \log n_1, n_2)$ and proof length $L_\pi(\lambda, T, n_1, n_2) = \mathsf{poly}(\lambda, \log n_1, n_2)$, and let Encode be the encoding algorithm that is guaranteed to exist by the holographic property. Let $\mathcal{H}$ be a hash function family such that each $h \in \mathcal{H}_\lambda$ is a length-halving hash function $h : \{0,1\}^{2\lambda} \to \{0,1\}^\lambda$. Let $\mathsf{poly}_h, \mathsf{poly}_T$ be any polynomials such that for every $\lambda \in \mathbb{N}$ and $h \in \mathcal{H}_\lambda$, the description size of h (as a bit string) is at most $\mathsf{poly}_h(\lambda)$ and the running time of the Turing machine M_h (as defined in Definition 22) is at most $\mathsf{poly}_T(L)$ for an input of length L.

Then, $\mathsf{TH\text{-}Del}$ can be viewed as a publicly verifiable weak tree-hash oracle memory delegation scheme for $\mathcal{H}$ as described in Figure 16. It is easy to check that the scheme satisfies the desired efficiency and security requirements. (For the weak soundness requirement, we consider as Valid an algorithm that checks whether the input is the LDE of an L_{DB}-bit string w.r.t. the LDE parameter $(\mathbb{F}, \mathbb{H}, m)$ that the encoding algorithm Encode of $\mathsf{TH\text{-}Del}$ would use, and we consider as Decode an algorithm that applies LDE decoding to the input.) The security holds against $\mathsf{poly}(\gamma)$-time adversaries for a super-polynomial γ if $\mathsf{TH\text{-}Del}$ is γ-secure.