Monotone-Policy BARGs and More from BARGs and Quadratic Residuosity

Shafik Nassar
UT Austin
shafik@cs.utexas.edu

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

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

Abstract

A tuple of NP statements $(x_1, \ldots, x_k)$ satisfies a monotone policy $P\colon \{0,1\}^k \to \{0,1\}$ if $P(b_1, \ldots, b_k) = 1$, where $b_i = 1$ if and only if x_i is in the NP language. A monotone-policy batch argument (monotone-policy BARG) for NP is a natural extension of regular batch arguments (BARGs) that allows a prover to prove that $x_1, \ldots, x_k$ satisfy a monotone policy P with a proof of size $\mathsf{poly}(\lambda, |\mathcal{R}|, \log k)$, where $|\mathcal{R}|$ is the size of the Boolean circuit computing the NP relation $\mathcal{R}$.

Previously, Brakerski, Brodsky, Kalai, Lombardi, and Paneth (CRYPTO 2023) and Nassar, Waters, and Wu (TCC 2024) showed how to construct monotone-policy BARGs from (somewhere-extractable) BARGs for NP together with a leveled homomorphic encryption scheme (Brakerski et al.) or an additively homomorphic encryption scheme over a *sufficiently-large* group (Nassar et al.). In this work, we improve upon both works by showing that BARGs together with additively homomorphic encryption over *any* group suffices (e.g., over $\mathbb{Z}_2$). For instance, we can instantiate the additively homomorphic encryption with the classic Goldwasser-Micali encryption scheme based on the quadratic residuosity (QR) assumption. Then, by appealing to existing compilers, we also obtain a monotone-policy aggregate signature scheme from any somewhere extractable BARG and the QR assumption.

1 Introduction

A non-interactive batch argument (BARG) for NP allows a prover to construct a short proof attesting that a collection of NP statements $(x_1, \ldots, x_k)$ are all true with a proof whose length scales sublinearly with k. BARGs have proven useful beyond the direct application of minimizing the communication cost of NP verification; they have been used to construct aggregate signatures [WW22, DGKV22, BCJP24, NWW24], delegation for RAM programs [KVZ21, CJJ21b, KLVW23], as well as non-interactive zero-knowledge proofs (NIZKs) [CW23, BKP+24, BWW24]. In recent years, a number of works have shown how to construct BARGs from many standard number-theoretic assumptions, such as the learning with errors (LWE) assumption [CJJ21b], the k-Lin assumption in pairing groups [WW22], the (sub-exponential) decisional Diffie-Hellman (DDH) assumption in pairing-free groups [CGJ+23], or a combination of quadratic residuosity (QR) and LWE or sub-exponential DDH [CJJ21a].

Monotone-policy batch arguments. In a batch argument, the prover's goal is to prove that *all* k statements $x_1, \ldots, x_k$ are true. Suppose instead that a prover wants to publish a proof attesting that a majority of the statements are true, or more generally, that the true statements satisfy some monotone policy such as a (weighted) threshold policy or a monotone Boolean formula. This is the notion of a monotone-policy BARG. Previous works [BCJP24, NWW24] show how to use monotone-policy BARGs to construct *monotone-policy aggregate signatures*, where an aggregator wants to produce a short proof attesting that an authorized quorum of parties have signed a certain message.

A trivial way to build a monotone-policy BARG from a vanilla BARG is to have the prover specify a subset $I \subseteq [t]$ that satisfy the policy and then give a vanilla BARG proof that all of the statements $\{x_i\}_{i \in I}$ are true. The verifier then checks that the subset I satisfies the policy and that the BARG proof verifies. In this case, however, the size of the proof potentially scales *linearly* with the number of statements (it needs to contain the description of the set I). In a *monotone policy BARG* [BBK+23], we require that the size of the proof be sublinear in the number of statements, just as in a vanilla BARG. If we specialize a monotone-policy BARG to the special case of conjunction policies, then we

recover the standard notion of a BARG. Thus, monotone-policy BARGs are a strict generalization of vanilla BARGs. A natural question to ask is whether we can construct monotone-policy BARGs from vanilla BARGs. A recent line of work has shown how to compile a BARG into a monotone-policy BARG using other cryptographic primitives:

- The first work by Brakerski, Brodsky, Kalai, Lombardi and Paneth [BBK+23] relied on BARGs in conjunction with (leveled) homomorphic encryption (which in turn relies either on LWE [Gen09, BV11] or strong tools like indistinguishability obfuscation [CLTV15]).
- Subsequently, Nassar, Waters, and Wu [NWW24] showed that BARGs along with an *additively* homomorphic encryption scheme suffice. Notably, this enabled new instantiations of monotone policy BARGs from pairing-based assumptions and from sub-exponential DDH.

A major caveat in [NWW24] is that the plaintext group for the additively homomorphic encryption must be sufficiently large (e.g., at least $k+1$ where k is the batch size). Unfortunately, this falls short of supporting *any* additively homomorphic encryption. An important example is the classic Goldwasser-Micali encryption scheme [GM82] based on the QR problem. The Goldwasser-Micali scheme is additively homomorphic over $\mathbb{Z}_2$, which is too small to be able to invoke the [NWW24] compiler. Another example is the Benaloh [Ben94] encryption scheme which is additively homomorphic over small groups $\mathbb{Z}_n$. This motivates the question of whether we can reduce the gap between BARGs and monotone-policy BARGs: namely, can we use *any* additively-homomorphic encryption scheme to compile BARGs into monotone-policy BARGs?

1.1 Our Results

In this work, we show how to construct a general monotone-policy BARG from a standard (somewhere-extractable) BARG and *any* additively-homomorphic encryption. In particular, assuming QR and a somewhere-extractable BARG, we obtain a monotone-policy BARG. Our main result can be summarized in the following theorem:

Theorem 1.1 (Informal)**.** *Suppose there exists a somewhere-extractable BARG and an additively homomorphic encryption over any group of size $n > 1$. Then there exists a monotone policy BARG for general monotone policies with non-adaptive soundness.*

Monotone-policy aggregate signatures. The work of [NWW24] also shows how to construct monotone-policy aggregate signatures with static unforgeability from any monotone-policy BARGs with non-adaptive soundness together with a puncturable signature scheme. In a monotone-policy aggregate signature [BCJP24], the aggregator can take a collection of tuples $(\mathsf{vk}_1, m_1, \sigma_1), \ldots, (\mathsf{vk}_k, m_k, \sigma_k)$ of verification key/message/signature triples and aggregate the signatures into a single short signature σ_{agg} with respect to some monotone policy P. The aggregate signature affirms that the aggregator possesses signatures for a subset of the messages that satisfies P.

Corollary 1.2 (Informal)**.** *Suppose there exists a somewhere-extractable BARG, an additively homomorphic encryption over any group of size $n > 1$, and a puncturable signature scheme. Then there exists a monotone-policy aggregate signature scheme satisfying static unforgeability.*

The work of [ADM+24] show how to construct puncturable signatures from any (simulation-sound) non-interactive zero-knowledge (NIZK) proof, which can be built from a wide range of assumptions, including the QR assumption [BFM88, Sah99, DDO+01].[1] In Appendix B, we also show an alternative route to building puncturable signatures from a *unique* signature scheme (i.e., a signature scheme where every message has exactly one signature), or more generally, from an invariant signature [GO92].[2]

[1] Note that the recent implications from BARGs to NIZKs [CW23, BKP+24, BWW24] only yield computationally-sound *arguments*, which do not seem to directly imply puncturable signatures via the [ADM+24] approach.

[2] The construction of invariant signatures from QR from [GO92] also relies on NIZK *proofs*, so this approach does not provide an advantage over the approach of [ADM+24]. We present it primarily to illustrate another approach for building puncturable signatures.

2 Technical Overview

In this section, we explain our techniques for getting a monotone policy BARG from an additively homomorphic encryption over a small group. For ease of exposition, we focus on additively homomorphic *bit* encryptions similar to [GM82].

Zero-fixing hash functions. The work of [NWW24] shows how to compile BARGs to monotone-policy BARGs using a *zero-fixing hash* (ZFH). For an overview of how a ZFH can be used to construct monotone-policy BARGs, we refer the reader to [NWW24]. In this work, we focus on constructing a ZFH, so we start by recalling the definition. In a nutshell, a ZFH is a succinct binding commitment with succinct local openings, similar to a Merkle hash [Mer87], but with an additional property: there is a secret trapdoor that can be used to decide whether a hash value is zero on a predetermined subset of indices. Zero-fixing hash functions can also be viewed as a special case of a function-binding hash function [FWW23] (for substring matching). We start by describing the syntax of a zero-fixing hash function:

- The setup algorithm of the ZFH takes as input a subset $S \subseteq [n]$, and outputs a hash key hk and a secret trapdoor td.
- The hash algorithm works like a regular Merkle hash algorithm: it takes the hash key hk and an input $x \in \{0,1\}^n$ and outputs a succinct digest dig and n succinct local openings $\pi_1, \ldots, \pi_n$.
- There exists a digest-validation algorithm $\mathsf{ValidateDigest}$ that takes as input a digest dig and the hash key hk and outputs 1 if the digest was computed honestly using the hash key hk.
- There exists an extraction algorithm $\mathsf{Extract}$ that given the trapdoor td and a digest dig, outputs either $\mathsf{Matching}$ or $\mathsf{NotMatching}$.

Next, the zero-fixing hash function should satisfy the following properties:

- **Opening correctness:** The opening correctness property states that any honestly generated digest and openings are valid.
- **Succinctness:** Similarly, succinctness is also standard and states the digest and the openings are $\mathrm{polylog}(n)$ bits each.
- **Digest correctness:** The digest correctness property states that for any digest dig and any hash key hk that is zero-fixing on the empty set, if $\mathsf{ValidateDigest}(\mathsf{hk}, \mathsf{dig}) = 1$ then $\mathsf{Extract}(\mathsf{td}, \mathsf{dig}) = \mathsf{Matching}$.
- **Zero-fixing:** The (computational) zero-fixing property requires that for any digest dig, if $\mathsf{Extract}(\mathsf{td}, \mathsf{dig}) = \mathsf{Matching}$, then it is computationally hard to find an opening π_i^* for some $i \in S$ to the value 1. In other words, if the adversary can open a digest dig on some index $i \in S$ to a 1, then the extraction algorithm should declare dig to be $\mathsf{NotMatching}$.
- **Set hiding:** The set-hiding property says that for any two subsets $S_0, S_1 \subseteq [n]$, an adversary that is only given access to hk (sampled to be zero-fixing on either S_0 or S_1) cannot distinguish if hk is zero-fixing on S_0 or S_1.

We remark here that one could also consider the following stronger requirement on $\mathsf{Extract}$: instead of outputting $\mathsf{NotMatching}$, it should output a specific index $i \in S$ for which it is feasible (for the adversary) to find an opening π_i^* to the value 1. Indeed, the work of [BBK+23] goes down this route, however implementing such a primitive seems to require fully-homomorphic encryption. On the other hand, [NWW24] notices that this stronger notion of extraction is unnecessary if we require an additional set-hiding property called *index hiding with extraction*. We elaborate on this property later on.

ZFH from homomorphic encryption. The conceptual idea of [NWW24] to build a ZFH from an additively homomorphic encryption is simple. If we want to hash n-bit inputs, the hash key consists of n ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_n$, one for each index. To hash a string $x \in \{0,1\}^n$, we first view it as a subset $X \subseteq [n]$ in the natural way ($x_i = 1$ if and only if $i \in X$), and take the digest dig_x to be an encryption of $\sum_{i \in X} \mathsf{ct}_i$, which can be computed homomorphically from $\mathsf{ct}_1, \ldots, \mathsf{ct}_n$.

The idea is now as follows: if we want the hash key to be zero-fixing on the set $S \subseteq [n]$, then we sample ct_i as an encryption of 1 if $i \in S$, and as an encryption of 0 if $i \notin S$. If $x_i = 0$ for all $i \in S$, then $X \cap S = \emptyset$, and if there exists some $i \in S$ such that $x_i = 1$, then $X \cap S \neq \emptyset$. This means dig_x decrypts to 0 if and only if x is all 0 on the set S. In this case, the secret decryption key is the extraction trapdoor.

This simplified construction already satisfies some key properties of a ZFH. First, the digest is succinct as it consists of the encryption of a single group element. Second, we have set hiding by the CPA security of the encryption.

The problem with XOR. However, we note that this simple idea already fails if the homomorphic encryption scheme only supports additive homomorphism over a small group. Take $\mathbb{Z}_2$ for example: if x has exactly two non-zero indices in S, then their corresponding ciphertexts will "cancel each other out." In fact, for this idea to work, [NWW24] required a group of size at least $n + 1$. Taking a step back, the homomorphic property with respect to addition is useful in the previous construction because of the fact that there is no going back once 1 is added, and the whole sum would be strictly greater than 0. Once we limit ourselves to a binary XOR operation, it is not clear where the "irreversible" operation would come from. It is worth noting that for *multiplicatively* homomorphic encryption, the previous idea would still work even with small groups: the irreversible operation in this case would be multiplying by 0, and, unlike addition, there is no way to cancel the 0 out using multiplication.

Substituting group elements with vectors. Our first idea is to simulate the irreversible operation by associating each index with a vector of ciphertexts instead of a single ciphertext. The digest now would be the (homomorphic) bitwise XOR of all of the vectors. Namely, imagine that each index $i \in [n]$ is associated with a binary vector $\mathbf{v}_i \in \mathbb{Z}_2^\ell$ such that the vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$ are linearly independent. In this case, once a vector is XORed in, there is no way to remove it since it is linearly independent of the other vectors. Unfortunately, getting n linearly independent vectors over $\mathbb{Z}_2^\ell$, requires $\ell \geq n$. This violates succinctness.

Reducing the vector dimension. Our second idea is to leverage the hiding property of the encryption scheme. In the simplified version of the [NWW24] construction we described above, CPA security is only used for set-hiding. Namely, once the adversary knows the zero-fixing set S, it knows that ct_i is an encryption of 1 for each $i \in S$. But if we use binary vectors instead of a fixed scalar, we can assign a *random* vector $\mathbf{v}_i \xleftarrow{\text{R}} \mathbb{Z}_2^\lambda$ to each $i \in S$, and never reveal the vector. Recall that in the previous construction, the hash key only contained encryptions of the elements, not the elements themselves (in order to satisfy set hiding). Intuitively, if we sample random vectors and only publish their encryptions, then these vectors should be computationally hidden from the view of the adversary. While these vectors are no longer linearly independent (in general, $n > \lambda$), the adversary should not be able to efficiently find a non-trivial linear combination of the non-zero vectors that maps to the zero vector. In more detail, we make the following changes:

- When sampling the hash key, each ciphertext ct_i is replaced with a ciphertext vector $\mathbf{ct}_i$. For each $i \in S$, the Setup algorithm samples a uniform $\mathbf{v}_i \xleftarrow{\text{R}} \mathbb{Z}_2^\lambda \setminus \{\mathbf{0}\}$, where $\mathbf{0}$ is the zero vector, and samples $\mathbf{ct}_i$ to be an encryption of $\mathbf{v}_i$. For each $i \notin S$, the algorithm samples $\mathbf{ct}_i$ as an encryption of $\mathbf{0}$. The trapdoor is still the secret key.
- When hashing a string x, the digest dig_x is an encryption of $\bigoplus_{i \in X} \mathbf{v}_i$. This can be computed by homomorphically evaluating the XOR function on a subset of the encrypted vectors $\mathbf{ct}_1, \ldots, \mathbf{ct}_n$.
- The Extract algorithm outputs Matching if and only if dig_x decrypts to $\mathbf{0}$.

Set hiding follows similarly to before. Digest succinctness also still holds since we have λ ciphertexts, which is independent of n.

Succinct openings. The next question is how to support local openings (i.e., open dig_x in position i to some value). The naïve way is to provide the list of the ciphertexts used to compute the digest dig_x (or equivalently, provide the entire hashed string x). Of course, this is not succinct. The works of [BBK+23, NWW24] used the standard technique of computing the digest via a Merkle-tree structure [Mer87]. Namely, the hash key includes a new ciphertext $\mathbf{ct}_{\mathsf{zero}}$ which is an encryption of $\mathbf{0}$. Given an input $x \in \{0,1\}^n$, the hashing algorithm constructs a complete binary tree with n leaves, where each leaf corresponds to an index $i \in [n]$. Each node i in the tree is associated with a ciphertext $\hat{\mathbf{ct}}_i$ as follows:

- For a leaf $i \in [n]$, if $x_i = 1$ then $\hat{\mathbf{ct}}_i = \mathbf{ct}_i$ and if $x_i = 0$ then $\hat{\mathbf{ct}}_i = \mathbf{ct}_{\mathsf{zero}}$.
- For an internal node i, $\hat{\mathbf{ct}}_i$ is obtained by homomorphically XOR-ing the ciphertexts associated with its children.

By construction, the root ciphertext $\hat{\mathbf{ct}}_{\mathsf{root}}$ is the homomorphic XOR of all of the leaf ciphertexts, which is by definition dig_x. This way, one only needs to provide the ciphertexts along the path to a leaf i in order to open the index i. Note that by construction, $\mathbf{ct}_{\mathsf{zero}}$ does not affect the decrypted value of $\hat{\mathbf{ct}}_{\mathsf{root}}$.

Validating the hash. To certify that a particular digest dig_x (consisting of a ciphertext $\hat{\mathbf{ct}}_{\mathsf{root}}$) is correctly computed, we follow the blueprint of [NWW24, BBK+23] and use a "hash-and-BARG technique" [CJJ21a]. Namely, the hashing algorithm now also computes a commitment $\mathsf{com}_{\mathsf{dig}}$ to the evaluation tree described above, and attaches a BARG proof that each node was computed honestly, alongside the root ciphertext $\hat{\mathbf{ct}}_{\mathsf{root}}$. In more detail, we define an NP relation parameterized by $\hat{\mathsf{ct}}_{\mathsf{root}}$, $\mathsf{com}_{\mathsf{dig}}$ and the ciphertexts $\mathbf{ct}_1, \ldots, \mathbf{ct}_n, \mathbf{ct}_{\mathsf{zero}}$. Each statement of the relation is an index of a node. The relation checks the following:

- **Leaf nodes:** For a leaf node i, we want to check that the associated ciphertext $\hat{\mathbf{ct}}_i$ is either equal to ct_i or $\mathbf{ct}_{\mathsf{zero}}$. Since the relation does not have access to $\hat{\mathbf{ct}}_i$, and instead only has access to the commitment $\mathsf{com}_{\mathsf{dig}}$, it actually checks that $\mathsf{com}_{\mathsf{dig}}$ opens in positions i to such a $\hat{\mathbf{ct}}_i$. In this case, the NP witness consists of an opening in $\mathsf{com}_{\mathsf{dig}}$ to position i.
- **Non-leaf nodes:** For a non-leaf node i with children $i_{\mathrm{L}}, i_{\mathrm{R}}$, the relation checks that $\mathsf{com}_{\mathsf{dig}}$ opens in positions $i, i_{\mathrm{L}}, i_{\mathrm{R}}$ to ciphertexts $\hat{\mathbf{ct}}_i, \hat{\mathbf{ct}}_{\mathrm{L}}, \hat{\mathbf{ct}}_{\mathrm{R}}$ respectively, where $\hat{\mathbf{ct}}_i$ is the ciphertext obtained by homomorphically XORing $\hat{\mathbf{ct}}_{\mathrm{L}}$ and $\hat{\mathbf{ct}}_{\mathrm{R}}$. In this case, the NP witness consists of the 3 openings in $\mathsf{com}_{\mathsf{dig}}$.
- **Root node:** For the root node, the relation additionally checks that $\mathsf{com}_{\mathsf{dig}}$ opens in the appropriate position to the given ciphertext $\hat{\mathbf{ct}}_{\mathsf{root}}$.

To keep the BARG proof short, we modify the hash key to include a commitment $\mathsf{com}_{\mathsf{hk}}$ of the ciphertexts $\mathbf{ct}_1, \ldots, \mathbf{ct}_n$, and modify the relation to depend on $\mathsf{com}_{\mathsf{hk}}$ instead of $\mathbf{ct}_1, \ldots, \mathbf{ct}_n$. The NP witness for a leaf node i now would need to also include the opening of $\mathsf{com}_{\mathsf{hk}}$ in position i (to the ciphertext $\mathbf{ct}_i$). With these modifications, the digest dig_x for an input $x \in \{0,1\}^\ell$ contains the root ciphertext $\hat{\mathbf{ct}}_{\mathsf{root}}$, the commitment $\mathsf{com}_{\mathsf{dig}}$, and the BARG proof π.

The honest opening to an index $i^* \in [n]$ with a value $b \in \{0,1\}$ is yet another BARG proof π_{open}, where the BARG statements are indices of the ciphertext evaluation tree. The NP relation is almost identical to the hashing relation described above, but is additionally parameterized by a pair $(i^*, b) \in [n] \times \{0,1\}$. The only difference is that for the leaf node i^*, the relation now additionally checks that if $b = 0$ then $\hat{\mathbf{ct}}_i = \mathbf{ct}_{\mathsf{zero}}$ and if $b = 1$ then $\hat{\mathbf{ct}}_i = \mathbf{ct}_i$.

Zero-fixing: first attempt. To argue zero-fixing security, suppose we have an adversary that outputs a digest dig together with an opening of some $i^* \in S$ to the value 1 and moreover, the Extract function declares dig to be Matching (i.e., dig decrypts to $\mathbf{0}$). By somewhere extractability of the BARG, this means that the ciphertext associated with leaf node i^* is an encryption of a non-zero vector $\mathbf{v}_{i^*}$. Since all of the vectors are encrypted, the hope is that the adversary cannot find a linear combination of other vectors $\mathbf{v}_i$ where $\mathbf{v}_{i^*} \oplus \bigoplus_{i \neq i^*} \mathbf{v}_i = \mathbf{0}$. Indeed, any adversary that does so must seemingly know something about the vectors $\mathbf{v}_i$, which of course, would violate CPA security of the encryption scheme. The challenge is in setting up the reduction to CPA security. Namely, in the zero-fixing security game, the adversary is only deemed successful if it produces a digest dig where Extract outputs Matching. However, evaluating the Extract algorithm requires knowledge of the secret key (to decrypt dig and compare the decrypted vector to $\mathbf{0}$). Yet, the reduction algorithm for the CPA security game cannot know the secret key, and thus, cannot determine whether the zero-fixing adversary outputted a valid digest or not.

Naor-Yung to the rescue. To get out of this conundrum, we adopt a Naor-Yung style strategy [NY90] and encrypt twice. Within each pair, we refer to one ciphertext as the "main" one and the other as a "shadow" copy.

- The setup algorithm samples two encryption key pairs: $(\mathsf{pk}^{\mathsf{main}}, \mathsf{sk}^{\mathsf{main}})$ and $(\mathsf{pk}^{\mathsf{shadow}}, \mathsf{sk}^{\mathsf{shadow}})$. For every index $i \in [n]$, we associate two ciphertext vectors: $\mathbf{ct}_i^{\mathsf{main}}$ and $\mathbf{ct}_i^{\mathsf{shadow}}$ under the encryption keys $\mathsf{pk}^{\mathsf{main}}$ and $\mathsf{pk}^{\mathsf{shadow}}$ respectively. Similarly, we also have two encryptions of the zero vector $\mathbf{ct}_{\mathsf{zero}}^{\mathsf{main}}$ and $\mathbf{ct}_{\mathsf{zero}}^{\mathsf{shadow}}$. The hash key is now defined analogously: it contains both public keys, both collections of encrypted vectors, and commitments (and openings) to both collections. The trapdoor is the main secret key $\mathsf{sk}^{\mathsf{main}}$ only.
- The hashing algorithm now computes two evaluation trees, and commits to both. The NP relation additionally requires that the evaluation trees are consistent with one another: for each leaf i, if the commitment of the main tree in position i opens to $\mathbf{ct}_i^{\mathsf{main}}$ then the commitment of the shadow tree also opens to $\mathbf{ct}_i^{\mathsf{shadow}}$, and if the main commitment in position i opens to $\mathbf{ct}_{\mathsf{zero}}^{\mathsf{main}}$ then the commitment of the shadow tree also opens to $\mathbf{ct}_{\mathsf{zero}}^{\mathsf{main}}$. The digest is the commitments and the roots for both trees, as well as the BARG proof π_{hash}.
- The extraction algorithm *only* checks that the root of the *main* tree decrypts to the all zero vector using $\mathsf{sk}^{\mathsf{main}}$.
- The opening is a BARG proof, where the NP relation is the same as the one used for hashing but, similar to before, is parameterized by (i^*, b) and requires that if $b = 0$ then the nodes i^* in both trees should use $\mathbf{ct}_{\mathsf{zero}}^{\mathsf{main}}$ and $\mathbf{ct}_{\mathsf{zero}}^{\mathsf{shadow}}$ respectively, and if $b = 1$ then the nodes should use $\mathbf{ct}_{i^*}^{\mathsf{main}}$ and $\mathbf{ct}_{i^*}^{\mathsf{shadow}}$ respectively.

We now argue our zero-fixing property through a series of hybrids:

1. The first hybrid is the original zero-fixing game where the adversary declares a set $S \subseteq [n]$ and an index $i^* \in S$. The challenger samples hk as described above and sends it to the adversary. The extraction trapdoor is the secret key $\mathsf{sk}^{\mathsf{main}}$. The adversary outputs $\mathsf{dig} = (\hat{\mathbf{ct}}_{\mathsf{root}}^{\mathsf{main}}, \hat{\mathbf{ct}}_{\mathsf{root}}^{\mathsf{shadow}}, \mathsf{com}_{\mathsf{dig}}^{\mathsf{main}}, \mathsf{com}_{\mathsf{dig}}^{\mathsf{shadow}}, \pi_{\mathsf{hash}})$ and an opening π^* of position i^* to value 1, and wins if $\mathsf{Dec}(\mathsf{sk}^{\mathsf{main}}, \hat{\mathbf{ct}}_{\mathsf{root}}^{\mathsf{main}}) = \mathbf{0}$ and π^* is a valid BARG proof for the NP relation with the pair $(i^*, 1)$.
2. In the second hybrid, we substitute the ciphertext associated with leaf i^* in the *shadow* copy only with an encryption of $\mathbf{0}$. Namely, $\mathbf{ct}_{i^*}^{\mathsf{shadow}} \leftarrow \mathsf{Enc}(\mathsf{pk}^{\mathsf{shadow}}, \mathbf{0})$. By CPA security (applied to the shadow copy), we can argue that the adversary behaves the same on this hybrid as it does in the previous one. Note that the challenger in this experiment only needs to know $\mathsf{sk}^{\mathsf{main}}$ (to implement Extract) and *not* $\mathsf{sk}^{\mathsf{shadow}}$. As such, we can rely on CPA security for the shadow copy to conclude that the output of this experiment is computationally indistinguishable from the previous one.
3. In the third hybrid, we change the extraction algorithm to use the shadow tree root instead of the main root. Namely, we check $\mathsf{Dec}(\mathsf{sk}^{\mathsf{shadow}}, \hat{\mathbf{ct}}_{\mathsf{root}}^{\mathsf{shadow}}) = \mathbf{v}_{i^*}$ instead of $\mathsf{Dec}(\mathsf{sk}^{\mathsf{main}}, \hat{\mathbf{ct}}_{\mathsf{root}}^{\mathsf{main}}) = \mathbf{0}$. Here, we appeal to the consistency that is guaranteed by the BARG: for each leaf i, the adversary has to use both $\mathbf{ct}_i^{\mathsf{main}}$ and $\mathbf{ct}_i^{\mathsf{shadow}}$ for the main and shadow copy, or use $\mathbf{ct}_{\mathsf{zero}}^{\mathsf{main}}$ and $\mathbf{ct}_{\mathsf{zero}}^{\mathsf{shadow}}$ for both copies. Since the opening to 1 on position i^* guarantees the commitments $\mathsf{com}_{\mathsf{dig}}^{\mathsf{main}}$ and $\mathsf{com}_{\mathsf{dig}}^{\mathsf{shadow}}$ open on position i^* to $\mathbf{ct}_{i^*}^{\mathsf{main}}$ and $\mathbf{ct}_{i^*}^{\mathsf{shadow}}$ respectively, and the values encrypted by those ciphertexts differ by exactly $\mathbf{v}_{i^*}$, then

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}^{\mathsf{shadow}}, \hat{\mathbf{ct}}_{\mathsf{root}}^{\mathsf{shadow}}) = \mathbf{v}_{i^*}\right] \approx \Pr\left[\mathsf{Dec}(\mathsf{sk}^{\mathsf{main}}, \hat{\mathbf{ct}}_{\mathsf{root}}^{\mathsf{main}}) = \mathbf{0}\right].$$

   Thus, the output of this experiment is computationally indistinguishable from the previous one.
4. For the final hybrid, similar to what we did in the second hybrid, we substitute the ciphertext associated with i^* in the *main* copy with an encryption of $\mathbf{0}$ (i.e., set $\mathbf{ct}_{i^*} \leftarrow \mathsf{Enc}(\mathsf{pk}^{\mathsf{main}}, \mathbf{0})$). In this experiment, the challenger's behavior only needs to know $\mathsf{sk}^{\mathsf{shadow}}$ and *not* $\mathsf{sk}^{\mathsf{main}}$, so the claim follows by CPA security applied to the main ciphertext.

In the final hybrid, the adversary wins if it outputs $\hat{\mathbf{ct}}_{\mathsf{root}}^{\mathsf{shadow}}$ that encrypts $\mathbf{v}_{i^*}$. However, its view is actually *independent* of $\mathbf{v}_{i^*}$, since we removed $\mathbf{v}_{i^*}$ from both the main and shadow copies. Finally, because the challenger samples $\mathbf{v}_{i^*} \xleftarrow{R} \mathbb{Z}_2^\lambda$, the adversary can successfully guess $\mathbf{v}_{i^*}$ only with probability $2^{-\lambda}$, thus completing the proof.

Index-hiding with extracted guess. The zero-fixing hash function of [NWW24] must satisfy an additional security property called index-hiding with extracted guess. Intuitively, this property states that the set on which the hash key is zero-fixing remains hidden, even if we give the adversary oracle access to $\mathsf{Extract}(\mathsf{td}, \cdot)$, as long as the queries made by the adversary do not help it to trivially distinguish between the binding sets. More formally, the game is defined as follows:

1. The adversary chooses a set $S \subseteq [n]$ and an index $i^* \in S$.
2. The challenger samples a random bit $b \xleftarrow{R} \{0,1\}$. If $b = 0$, the challenger samples hk to be zero-fixing on $S \setminus \{i^*\}$ and if $b = 1$, the challenger samples hk to be zero-fixing on S. The challenger gives hk to the adversary.
3. The adversary now outputs a digest dig and an opening σ.
4. The output is 1 if and only if σ is an opening of dig to the value 0 at index i^* and moreover, $\mathsf{Extract}(\mathsf{td}, \mathsf{dig}) = \mathsf{Matching}$.

We say the scheme satisfies index-hiding with extracted guess if for any efficient adversary, the output of the experiment when $b = 0$ is negligibly close to the output when $b = 1$. We can view the output as being extracted from dig, but the adversary is forced to provide an opening σ for dig at index i^* to the value 0. This rules out the trivial strategy of hashing a string $\mathbf{x}$ that is 1 in index i^* and 0 elsewhere. Such a string would be considered $\mathsf{Matching}$ if the hash key was binding on $S \setminus \{i^*\}$ and $\mathsf{NotMatching}$ if the hash key was binding on S.

The construction we provided already satisfies this property. Our argument is similar to that of [NWW24], and follows a Naor-Yung strategy similar to what we used to argue zero-fixing. The only difference between the game with $b = 0$ and $b = 1$ are the ciphertexts $\mathbf{ct}_{i^*}^{\mathsf{main}}$ and $\mathbf{ct}_{i^*}^{\mathsf{shadow}}$. When $b = 0$, these are encryptions of $\mathbf{0}$, and when $b = 1$, these are encryptions of a random vector $\mathbf{v}_{i^*}$.

We define the following series of hybrids, which follows the same templates as the series of hybrids used to argue zero-fixing:

1. The first hybrid is the index-hiding with extracted guess game with $b = 0$. Namely, $\mathbf{ct}^{\mathsf{main}} i^* \leftarrow \mathsf{Enc}(\mathsf{pk}^{\mathsf{main}}, \mathbf{0})$ and $\mathbf{ct}^{\mathsf{shadow}} i^* \leftarrow \mathsf{Enc}(\mathsf{pk}^{\mathsf{shadow}}, \mathbf{0})$.
2. The second hybrid is the same as before, except $\mathbf{ct}_{i^*}^{\mathsf{shadow}} \leftarrow \mathsf{Enc}(\mathsf{pk}^{\mathsf{shadow}}, \mathbf{v}_{i^*})$. Since the security game does not use the secret key $\mathsf{sk}^{\mathsf{shadow}}$, we can use the CPA security of the shadow instance to argue that these two hybrids are computationally close.
3. The third hybrid is the same as before, except the extraction algorithm now uses the shadow instance to extract the guess. Namely, the extraction algorithm outputs $\mathsf{Matching}$ if and only if $\mathsf{Dec}(\mathsf{sk}^{\mathsf{shadow}}, \hat{\mathbf{ct}}_{\mathsf{root}}^{\mathsf{shadow}}) = \mathbf{0}$ instead of $\mathsf{Dec}(\mathsf{sk}^{\mathsf{main}}, \hat{\mathbf{ct}}_{\mathsf{root}}^{\mathsf{main}}) = \mathbf{0}$. The two hybrids are computationally close by the consistency that is guaranteed by the BARG (similar to the zero-fixing argument) and the additional requirement that the hashed string has value 0 on index i^*. Notably, this is where we use the fact that the adversary must produce an opening σ to 0 at index i^*.
4. The fourth hybrid is the same as before, except $\mathbf{ct}_{i^*}^{\mathsf{main}} \leftarrow \mathsf{Enc}(\mathsf{pk}^{\mathsf{main}}, \mathbf{v}_{i^*})$. Since the security game does not use the secret key $\mathsf{sk}^{\mathsf{main}}$ anymore, we can use CPA security of the main instance to argue that these two hybrids are computationally close.
5. The final hybrid is the same as before, except we change back the extraction algorithm to check the main instance. Namely, the extraction algorithm outputs $\mathsf{Matching}$ if and only if $\mathsf{Dec}(\mathsf{sk}^{\mathsf{main}}, \hat{\mathbf{ct}}_{\mathsf{root}}^{\mathsf{main}}) = \mathbf{0}$. This hybrid is computationally close to the previous one by the same argument we used to justify the third hybrid. We note that this is the index-hiding game with $b = 1$ and thus we are done.

3 Preliminaries

Throughout this work, we write λ to denote the security parameter. For $n \in \mathbb{N}$, we write $[n]$ to denote the set $\{1, \ldots, n\}$. For any $m > n$, we write $[n, m]$ to denote the set $\{n, \ldots, m\}$. We write $\mathsf{poly}(\lambda)$ to denote a function that

is bounded by a fixed polynomial in λ, and $\mathsf{negl}(\lambda)$ to denote a function that is $o(\lambda^{-c})$ for all $c \in \mathbb{N}$. For a finite set S, we write $x \xleftarrow{R} S$ to denote that x is a uniformly random element of S. For a distribution $\mathcal{D}$ we write $x \leftarrow \mathcal{D}$ to denote that x is a random drawn from $\mathcal{D}$.

We say an algorithm is efficient if it runs in probabilistic polynomial time in the length of its input. A non-uniform algorithm $\mathcal{A}$ consists of a pair of algorithms $(\mathcal{A}_1, \mathcal{A}_2)$ where $\mathcal{A}_1$ is a (possibly-unbounded) algorithm that takes as input 1^λ and outputs an advice string ρ_λ of $\mathsf{poly}(\lambda)$ size. Algorithm $\mathcal{A}_2$ is an efficient algorithm. The output of $\mathcal{A}$ on an input $x \in \{0,1\}^\lambda$ is defined as first computing the advice string $\rho_\lambda \leftarrow \mathcal{A}_1(1^\lambda)$ and then outputting $\mathcal{A}_2(x, \rho_\lambda)$. We say two ensembles of distributions $\mathcal{D}_1 = \{\mathcal{D}_{1,\lambda}\}_{\lambda\in\mathbb{N}}$ and $\mathcal{D}_2 = \{\mathcal{D}_{2,\lambda}\}_{\lambda\in\mathbb{N}}$ are computationally indistinguishable if no efficient adversary can distinguish them with non-negligible probability. We say they are statistically indistinguishable if their statistical distance is bounded by $\mathsf{negl}(\lambda)$.

3.1 Cryptographic Building Blocks

In this section, we recall the definition of a few standard cryptographic building blocks we use in this work.

Additively-homomorphic encryption over $\mathbb{Z}_p$. We start by reviewing the notion of additively homomorphic encryption over $\mathbb{Z}_p$.

Definition 3.1 (Additively Homomorphic Encryption over $\mathbb{Z}_p$)**.** An additively homomorphic encryption scheme over $\mathbb{Z}_p$ is a tuple of polynomial-time algorithms $\Pi_{\mathsf{HE}} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Add})$ with the following syntax:

- $\mathsf{Gen}(1^\lambda) \to (\mathsf{sk}, \mathsf{pk})$: On input a security parameter $\lambda \in \mathbb{N}$, the key-generation algorithm outputs a secret key sk and a public key pk.
- $\mathsf{Enc}(\mathsf{pk}, \mathsf{msg}) \to \mathbf{ct}$: On input a public key pk and a message $\mathsf{msg} \in \mathbb{Z}_p^\ell$ of length $\ell \in \mathbb{N}$, the encryption algorithm outputs a ciphertext vector $\mathbf{ct} = (\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$ of length ℓ.
- $\mathsf{Dec}(\mathsf{sk}, \mathbf{ct}) \to \mathsf{msg}$: On input a secret key sk and a ciphertext vector $\mathbf{ct} = (\mathbf{ct}_1, \ldots, \mathbf{ct}_\ell)$ of length $\ell \in \mathbb{N}$, the decryption algorithm either outputs a plaintext $\mathsf{msg} \in \mathbb{Z}_p^\ell$, or a special symbol $\mathsf{msg} = \perp$. The decryption algorithm is deterministic.
- $\mathsf{Add}(\mathsf{pk}, \mathbf{ct}_1, \mathbf{ct}_2) \to \mathbf{ct}'$: On input a public key pk and two ciphertext vectors $\mathbf{ct}_1, \mathbf{ct}_2$ of the same length ℓ, the homomorphic addition algorithm outputs a new ciphertext vector $\mathbf{ct}'$ of length ℓ. The addition algorithm is deterministic.

We require the following properties:

- **Correctness:** For all $\lambda, \ell \in \mathbb{N}$ and all messages $\mathsf{msg} \in \mathbb{Z}_p^\ell$, it holds that:

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}, \mathbf{ct}) = \mathsf{msg} : \begin{array}{c}(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(1^\lambda, 1^n) \\ \mathbf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{msg})\end{array}\right] = 1.$$

- **Evaluation correctness:** For all $\lambda, \ell \in \mathbb{N}$, all $(\mathsf{sk}, \mathsf{pk})$ in the support of $\mathsf{Gen}(1^\lambda)$ and all ciphertext vectors $\mathbf{ct}_1, \mathbf{ct}_2$ of the same length ℓ, where $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}_1) \neq \perp$ and $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}_2) \neq \perp$, it holds that

$$\mathsf{Dec}(\mathsf{sk}, \mathsf{Add}(\mathsf{pk}, \mathsf{ct}_1, \mathsf{ct}_2)) = \mathsf{Dec}(\mathsf{sk}, \mathsf{ct}_1) + \mathsf{Dec}(\mathsf{sk}, \mathsf{ct}_2).$$

- **Compactness:** There exists a polynomial $\mathsf{poly}(\cdot)$ such that for all $\lambda, \ell \in \mathbb{N}$, all $(\mathsf{sk}, \mathsf{pk})$ in the support of $\mathsf{Gen}(1^\lambda)$, all messages $\mathsf{msg}_1, \mathsf{msg}_2 \in \mathbb{Z}_p^\ell$, all ciphertexts $\mathbf{ct}_1, \mathbf{ct}_2$ in the support of $\mathsf{Enc}(\mathsf{pk}, \mathsf{msg}_1)$ and $\mathsf{Enc}(\mathsf{pk}, \mathsf{msg}_2)$ respectively, it holds that

$$|\mathbf{ct}_1|, |\mathbf{ct}_2| \leq \ell \cdot \mathsf{poly}(\lambda) \quad \text{and} \quad |\mathsf{Add}(\mathsf{pk}, \mathbf{ct}_1, \mathbf{ct}_2)| \leq \ell \cdot \mathsf{poly}(\lambda).$$

- **CPA-security:** For an adversary $\mathcal{A}$ and a bit $b \in \{0,1\}$, define the CPA-security experiment $\mathsf{ExptSS}_{\mathcal{A}}(\lambda, b)$ as follows:

1. On input the security parameter 1^λ, the challenger samples a key pair $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(1^\lambda)$ and sends pk to the adversary.
2. The adversary can now make (arbitrarily many) queries on pairs of messages $(\mathsf{msg}_0, \mathsf{msg}_1)$ (where msg_0 and msg_1 are vectors with the same dimension). On each query, the challenger replies with a ciphertext $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{msg}_b)$.
3. After the adversary $\mathcal{A}$ is done making queries, it outputs a guess $b' \in \{0,1\}$.

We say that Π_{HE} is semantically secure if for every efficient adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\left|\Pr[\mathsf{ExptSS}_{\mathcal{A}}(\lambda, 1) = 1] - \Pr[\mathsf{ExptSS}_{\mathcal{A}}(\lambda, 0) = 1]\right| = \mathsf{negl}(\lambda)$.

Fact 3.2 (Additively Homomorphic Encryption over $\mathbb{Z}_p$ [GM82, Ben94])**.** Under the QR assumption, there exists an additively homomorphic encryption scheme over $\mathbb{Z}_2$. For any constant $p > 2$, under the p^{th}-order residuosity assumption, there exists an additively homomorphic encryption scheme over $\mathbb{Z}_p$.

The remaining definitions are copied mostly verbatim from [NWW24].

Vector commitments. Next, we recall the notion of a vector commitment scheme with succinct local openings. Such commitments can be built from any collision-resistant hash function [Mer87].

Definition 3.3 (Vector Commitment)**.** A vector commitment (VC) with local openings is a tuple of efficient algorithms $\Pi_{\mathsf{Com}} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Verify})$ with the following properties:

- $\mathsf{Setup}(1^\lambda, 1^n, 1^\ell) \to \mathsf{crs}$: On input the security parameter $\lambda \in \mathbb{N}$, the block length $n \in \mathbb{N}$, and the vector length $\ell \in \mathbb{N}$, the setup algorithm outputs a common reference string crs. We assume the common reference string implicitly contains the parameters 1^n and 1^ℓ.
- $\mathsf{Commit}(\mathsf{crs}, (x_1, \ldots, x_t)) \to (\mathsf{com}, \sigma_1, \ldots, \sigma_t)$: On input the common reference string crs and a vector of $t \le \ell$ messages $x_1, \ldots, x_t \in \{0,1\}^n$, the commit algorithm outputs a commitment com and openings $\sigma_1, \ldots, \sigma_t$.
- $\mathsf{Verify}(\mathsf{crs}, \mathsf{com}, i, y, \sigma) \to b'$: On input the common reference string crs, the commitment com, an index $i \in [\ell]$, a message $y \in \{0,1\}^n$, and an opening σ, the verification algorithm outputs a bit $b' \in \{0,1\}$.

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

- **Correctness:** For all $\lambda, n, \ell \in \mathbb{N}$, and all positive $t \le \ell$, all $\mathbf{x} = (x_1, \ldots, x_t) \in \{0,1\}^{tn}$, and indices $i \in [t]$,

$$\Pr\left[\mathsf{Verify}(\mathsf{crs}, \mathsf{com}, i, x_i, \sigma_i) = 1 : \begin{array}{c} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^n, 1^\ell), \\ (\mathsf{com}, \sigma_1, \ldots, \sigma_t) \leftarrow \mathsf{Commit}(\mathsf{crs}, \mathbf{x}) \end{array}\right] = 1.$$

- **Computational binding:** For an adversary $\mathcal{A}$, define the computational binding experiment as follows:
  1. On input the security parameter 1^λ, algorithm $\mathcal{A}$ starts by outputting the block length 1^n and vector length 1^ℓ.
  2. The challenger responds with $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^n, 1^\ell)$.
  3. Algorithm $\mathcal{A}$ outputs a commitment com, an index $i \in [\ell]$, and openings (y_0, σ_0) and (y_1, σ_1).
  4. The output of the experiment is $b = 1$ if $\mathsf{Verify}(\mathsf{crs}, \mathsf{com}, i, y_0, \sigma_0) = 1 = \mathsf{Verify}(\mathsf{crs}, \mathsf{com}, i, y_1, \sigma_1)$ and $y_0 \neq y_1$. Otherwise, the output is $b = 0$.

  The commitment scheme is binding if for all efficient adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\Pr[b = 1] = \mathsf{negl}(\lambda)$ in the binding experiment.
- **Succinctness:** There exists a universal polynomial $\mathsf{poly}(\cdot)$ such that for all $\lambda, n, \ell \in \mathbb{N}$, all crs in the support of $\mathsf{Setup}(1^\lambda, \ell)$, and all $(\mathsf{com}, \sigma_1, \ldots, \sigma_\ell)$ in the support of $\mathsf{Commit}(\mathsf{crs}, \cdot)$, the following holds:
  - **Succinct CRS:** $|\mathsf{crs}| = \mathsf{poly}(\lambda + \log n + \log \ell)$.

- **Succinct commitment:** $|\mathsf{com}| = \mathsf{poly}(\lambda + \log n + \log \ell)$.
- **Succinct local opening:** For all $i \in [\ell]$, $|\sigma_i| = \mathsf{poly}(\lambda + \log n + \log \ell)$.

Fact 3.4 (Vector Commitments from Homomorphic Encryption [Mer87, IKO05])**.** If any homomorphic encryption exists, then there exists a vector commitment scheme with local openings.

3.2 Batch Arguments for NP

In this section, we recall the notion of a non-interactive batch argument (BARG) for NP, the special case of a BARG for index languages [CJJ21b] and the notion of a BARG for monotone policy batch NP [BBK+23, NWW24].

Batch arguments for NP. We begin with the notion of a somewhere extractable batch argument for NP. Our presentation follows that of [NWW24], with the syntax where the batch arguments support extraction on up to ℓ indices.

Definition 3.5 (Boolean Circuit Satisfiability)**.** We define the circuit satisfiability language $\mathcal{L}_{\mathsf{CSAT}}$ as

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

Definition 3.6 (BARG)**.** A somewhere-extractable non-interactive batch argument (BARG) for Boolean circuit satisfiability is a tuple of efficient algorithms $\Pi_{\mathsf{BARG}} = (\mathsf{Gen}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{TrapGen}, \mathsf{Extract})$ with the following syntax:

- $\mathsf{Gen}(1^\lambda, 1^k, 1^n, 1^s, 1^\ell) \to (\mathsf{crs}, \mathsf{vk})$: On input the security parameter $\lambda \in \mathbb{N}$, the number of instances $k \in \mathbb{N}$, instance size $n \in \mathbb{N}$, a bound on the size of the Boolean circuit $s \in \mathbb{N}$, and a bound on the size of the extraction set $\ell \in \mathbb{N}$, the generator algorithm outputs a common reference string crs and a verification key vk.
- $\mathsf{Prove}(\mathsf{crs}, C, (x_1, \ldots, x_k), (w_1, \ldots, w_k)) \to \pi$: On input the common reference string crs, a Boolean circuit $C\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$, statements $x_1, \ldots, x_k \in \{0,1\}^k$, and witnesses $w_1, \ldots, w_k \in \{0,1\}^h$, the prove algorithm outputs a proof π.
- $\mathsf{Verify}(\mathsf{vk}, C, (x_1, \ldots, x_k), \pi) \to b$: On input the verification key vk, a Boolean circuit $C\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$, statements $x_1, \ldots, x_k \in \{0,1\}^n$ and a proof π, the verification algorithm outputs a bit $b \in \{0,1\}$.
- $\mathsf{TrapGen}(1^\lambda, 1^k, 1^n, 1^s, 1^\ell, S) \to (\mathsf{crs}, \mathsf{vk}, \mathsf{td})$: On input the security parameter $\lambda \in \mathbb{N}$, the number of instances $k \in \mathbb{N}$, instance size $n \in \mathbb{N}$, a bound on the size of the Boolean circuit $s \in \mathbb{N}$, a bound on the size of the extraction set $\ell \in \mathbb{N}$, and a set $S \subseteq [k]$ of size at most ℓ, the trapdoor generator algorithm outputs a common reference string crs, a verification key vk and an extraction trapdoor td.
- $\mathsf{Extract}(\mathsf{td}, C, (x_1, \ldots, x_k), \pi, i) \to w$. On input the trapdoor td, a Boolean circuit $C\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$, a collection of statements $x_1, \ldots, x_k \in \{0,1\}^n$, a proof π and an index $i \in [k]$, the extraction algorithm outputs a witness w.

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

- **Completeness:** For all $\lambda, k, n, s, \ell \in \mathbb{N}$, all Boolean circuits $C\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$ of size at most s, all statements $\mathbf{x} = (x_1, \ldots, x_k) \in \{0,1\}^{kn}$ and witnesses $\mathbf{w} = (w_1, \ldots, w_k) \in \{0,1\}^{kh}$ where $C(x_i, w_i) = 1$ for all $i \in [k]$,

$$\Pr\left[\mathsf{Verify}(\mathsf{vk}, C, \mathbf{x}, \pi) = 1 : \begin{array}{c} (\mathsf{crs}, \mathsf{vk}) \leftarrow \mathsf{Gen}(1^\lambda, 1^k, 1^n, 1^s, 1^\ell), \\ \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, C, \mathbf{x}, \mathbf{w}) \end{array}\right] = 1.$$

- **Set hiding:** For an adversary $\mathcal{A}$ and a bit $b \in \{0,1\}$, define the set hiding experiment $\mathsf{ExptSH}^{\mathsf{BARG}}_{\mathcal{A}}(\lambda, b)$ as follows:
  1. Algorithm $\mathcal{A}(1^\lambda)$ starts by outputting the number of instances 1^k, the instance size 1^n, the bound on the circuit size 1^s, the bound on the size of the extraction set 1^ℓ, and a set $S \subseteq [k]$ of size at most ℓ.

2. If $b = 0$, the challenger gives $(\mathsf{crs}, \mathsf{vk}) \leftarrow \mathsf{Gen}(1^\lambda, 1^k, 1^n, 1^s, 1^\ell)$ to $\mathcal{A}$. If $b = 1$, the challenger samples $(\mathsf{crs}, \mathsf{vk}, \mathsf{td}) \leftarrow \mathsf{TrapGen}(1^\lambda, 1^k, 1^n, 1^s, 1^\ell, S)$ and gives $(\mathsf{crs}, \mathsf{vk})$ to $\mathcal{A}$.
3. Algorithm $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$, which is the output of the experiment.

Then, Π_{BARG} satisfies set hiding if for every efficient adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that

$$\left|\Pr[\mathsf{ExptSH}^{\mathsf{BARG}}_{\mathcal{A}}(\lambda, 0) = 1] - \Pr[\mathsf{ExptSH}^{\mathsf{BARG}}_{\mathcal{A}}(\lambda, 1) = 1]\right| = \mathsf{negl}(\lambda).$$

- **Somewhere extractable in trapdoor mode:** For an adversary $\mathcal{A}$, define the somewhere extractable security game as follows:

1. Algorithm $\mathcal{A}(1^\lambda)$ starts by outputting the number of instances 1^k, the instance size 1^n, the bound on the circuit size 1^s, a bound on the size of the extraction set 1^ℓ, and a nonempty set $S \subseteq [k]$ of size at most ℓ.
2. The challenger samples $(\mathsf{crs}, \mathsf{vk}, \mathsf{td}) \leftarrow \mathsf{TrapGen}(1^\lambda, 1^k, 1^n, 1^s, 1^\ell, S)$ and gives $(\mathsf{crs}, \mathsf{vk})$ to $\mathcal{A}$.
3. Algorithm $\mathcal{A}$ outputs a Boolean circuit $C\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$ of size at most s, statements $x_1, \ldots, x_m \in \{0,1\}^n$, and a proof π.
4. The output of the game is $b = 1$ if $\mathsf{Verify}(\mathsf{vk}, C, (x_1, \ldots, x_m), \pi) = 1$ and there exists an index $i \in S$ for which $C(x_i, w_i) \neq 1$ where $w_i \leftarrow \mathsf{Extract}(\mathsf{td}, C, (x_1, \ldots, x_k), \pi, i)$. Otherwise, the output is $b = 0$.

Then Π_{BARG} is somewhere extractable in trapdoor mode if for every adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\Pr[b = 1] = \mathsf{negl}(\lambda)$ in the somewhere extractable game.

- **Succinctness:** There exists a fixed polynomial $\mathsf{poly}(\cdot)$ such that for all $\lambda, k, n, s, \ell \in \mathbb{N}$, all crs in the support of $\mathsf{Gen}(1^\lambda, 1^k, 1^n, 1^s, 1^\ell)$, and all Boolean circuits $C\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$ of size at most s, the following properties hold:
  - **Succinct proofs:** The proof π output by $\mathsf{Prove}(\mathsf{crs}, C, \cdot, \cdot)$ satisfies $|\pi| \leq \mathsf{poly}(\lambda + \log k + s + \ell)$.
  - **Succinct CRS:** $|\mathsf{crs}| \leq \mathsf{poly}(\lambda + k + n + \ell) + \mathsf{poly}(\lambda + \log k + s + \ell)$.
  - **Succinct verification key:** $|\mathsf{vk}| \leq \mathsf{poly}(\lambda + \log k + s + \ell)$.

Set hiding with extraction. Following the work of [NWW24], we also require the BARG to satisfy property of *set hiding with extraction*, which we define below. As shown in [NWW24], any somewhere extractable BARG can be modified to satisfy set hiding with extraction.

Definition 3.7 (Set Hiding with Extraction)**.** Let $\Pi_{\mathsf{BARG}} = (\mathsf{Gen}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{TrapGen}, \mathsf{Extract})$ be a somewhere extractable batch argument for Boolean circuit satisfiability (Definition 3.6). For an adversary $\mathcal{A}$ and a bit $b \in \{0,1\}$, define the set hiding with extraction experiment $\mathsf{ExptSHwE}(\lambda, b)$ as follows:

1. Algorithm $\mathcal{A}(1^\lambda)$ starts by outputting the number of instances 1^k, the instance size 1^n, the bound on the circuit size 1^s, the bound on the extraction set 1^ℓ, a set $S \subseteq [k]$ of size at most ℓ, and an index $i^* \in S$.
2. If $b = 0$, the challenger samples $(\mathsf{crs}, \mathsf{vk}, \mathsf{td}) \leftarrow \mathsf{TrapGen}(1^\lambda, 1^k, 1^n, 1^s, 1^\ell, S)$. If $b = 1$, the challenger samples $(\mathsf{crs}, \mathsf{vk}, \mathsf{td}) \leftarrow \mathsf{TrapGen}(1^\lambda, 1^k, 1^n, 1^s, 1^\ell, \{i^*\})$. The challenger replies to $\mathcal{A}$ with $(\mathsf{crs}, \mathsf{vk})$.
3. Algorithm $\mathcal{A}$ outputs a Boolean circuit $C\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$, statements $x_1, \ldots, x_k \in \{0,1\}^n$, and a proof π.
4. If $\mathsf{Verify}(\mathsf{vk}, C, (x_1, \ldots, x_k), \pi) \neq 1$, then the experiment halts with output 0. Otherwise, the challenger replies with $w^* \leftarrow \mathsf{Extract}(\mathsf{td}, C, (x_1, \ldots, x_k), \pi, i^*)$.
5. Algorithm $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$, which is the output of the experiment.

Then, Π_{BARG} satisfies set hiding with extraction if for every efficient adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$\left|\Pr[\mathsf{ExptSHwE}_{\mathcal{A}}(\lambda, 0) = 1] - \Pr[\mathsf{ExptSHwE}_{\mathcal{A}}(\lambda, 1) = 1]\right| = \mathsf{negl}(\lambda).$$

Index BARGs. An index BARG [CJJ21b] is a batch argument for the batch index language where the instance is *always* the tuple $(1, \ldots, k)$. Since the statements are the integers, they have a succinct description, so we can impose a stronger requirement on the running time of the Verify algorithm. We define this below:

Definition 3.8 (Index BARG [CJJ21b])**.** An index BARG is a special case of a BARG where the instances $(x_1, \ldots, x_k)$ are restricted to the integers $(1, \ldots, k)$. In this setting, the Gen algorithm to the index BARG does not separately take in the instance length n as a separate input. Moreover, instead of providing $x_1, \ldots, x_k$ as input to the Prove, Verify, and Extract algorithms, we just give the single index k (in binary). Moreover, we require the *additional* succinctness property on the running time of Verify:

- **Succinct verification time:** There exists a fixed polynomial $\mathsf{poly}(\cdot)$ such that for all $\lambda, k, n, s, \ell \in \mathbb{N}$, all $(\mathsf{crs}, \mathsf{vk})$ in the support of $\mathsf{Gen}(1^\lambda, 1^k, 1^s, 1^\ell)$ and all Boolean circuits $C\colon [k] \times \{0,1\}^h \to \{0,1\}$ of size at most s, the running time of $\mathsf{Verify}(\mathsf{vk}, C, k, \cdot)$ is bounded by $\mathsf{poly}(\lambda + \log k + s + \ell)$.

3.3 Zero-Fixing Hash Functions

In this section, we recall the notion of a zero-fixing hash function [NWW24]. As shown in [NWW24], a zero-fixing hash function can be combined with any vanilla BARG to obtain a monotone policy BARG. Recall that a zero-fixing hash function is a keyed hash function that supports succinct local openings. Moreover, the hash key is associated with a set of indices $S \subseteq [n]$, where n is the input length. Moreover, there is a trapdoor td associated with the hash key hk that can be used to decide whether a hash digest dig is Matching or NotMatching on the set S. The zero-fixing security requirement then says that if the extractor outputs Matching for a digest dig, it must be computationally hard to open dig to a 1 on any index $i \in S$. We now give the formal definition:

Definition 3.9 (Zero-Fixing Hash Function)**.** A *zero-fixing hash function* is a tuple of polynomial-time algorithms $\Pi_{\mathsf{H}} = (\mathsf{Setup}, \mathsf{Hash}, \mathsf{ProveOpen}, \mathsf{VerOpen}, \mathsf{Extract}, \mathsf{ValidateDigest})$ with the following syntax:

- $\mathsf{Setup}(1^\lambda, 1^n, S) \to (\mathsf{hk}, \mathsf{vk}, \mathsf{td})$: On input a security parameter λ, an input length n, and a set $S \subseteq [n]$, the setup algorithm outputs a hash key hk, a verification key vk and a trapdoor td. We implicitly assume that hk includes λ and n.
- $\mathsf{Hash}(\mathsf{hk}, x) \to \mathsf{dig}$: On input a hash key hk and a string $x \in \{0,1\}^n$, the hash algorithm outputs a digest dig. This algorithm is deterministic.
- $\mathsf{ValidateDigest}(\mathsf{vk}, \mathsf{dig}) \to b$: On input a hash key vk and a digest dig, the digest validation algorithm outputs a bit $b \in \{0,1\}$. This algorithm is deterministic.
- $\mathsf{ProveOpen}(\mathsf{hk}, x, i) \to \sigma$: On input a hash key hk, a string $x \in \{0,1\}^n$ and an index $i \in [n]$, the prove algorithm outputs an opening σ.
- $\mathsf{VerOpen}(\mathsf{vk}, \mathsf{dig}, i, b, \sigma) \to b'$: On input a hash key vk, a digest dig, an index $i \in [n]$, a bit $b \in \{0,1\}$ and an opening σ, the verification algorithm outputs a bit $b' \in \{0,1\}$. The verification algorithm is deterministic.
- $\mathsf{Extract}(\mathsf{td}, \mathsf{dig}) \to m$: On input a trapdoor td and a digest dig, the extraction algorithm outputs a value $m \in \{\mathsf{Matching}, \mathsf{NotMatching}\}$. This algorithm is deterministic.

We require Π_{H} satisfy the following efficiency and correctness properties:

- **Succinctness:** There exists a universal polynomial $\mathsf{poly}(\cdot)$ such that for all parameters $\lambda, n \in \mathbb{N}$, all $(\mathsf{hk}, \mathsf{vk}, \mathsf{td})$ in the support of $\mathsf{Setup}(1^\lambda, 1^n, \cdot)$, all inputs $x \in \{0,1\}^n$ and all indices $i \in [n]$, the following properties hold:
  - **Succinct verification key:** $|\mathsf{vk}| \le \mathsf{poly}(\lambda + \log n)$.
  - **Succinct digest:** The digest dig output by $\mathsf{Hash}(\mathsf{hk}, x)$ satisfies $|\mathsf{dig}| \le \mathsf{poly}(\lambda + \log n)$.
  - **Succinct openings:** The opening σ output by $\mathsf{ProveOpen}(\mathsf{hk}, x, i)$ satisfies $|\sigma| \le \mathsf{poly}(\lambda + \log n)$.
  - **Succinct verification:** The running time of $\mathsf{VerOpen}(\mathsf{vk}, \cdot)$ is $\mathsf{poly}(\lambda + \log n)$.

- **Correctness:** For all $\lambda, n \in \mathbb{N}$, every $x \in \{0,1\}^n$, and every $i \in [n]$, the following properties hold:
  - **Opening correctness:**

$$\Pr\left[\mathsf{VerOpen}(\mathsf{vk}, \mathsf{dig}, i, x_i, \sigma) = 1 \;:\; \begin{array}{c}(\mathsf{hk}, \mathsf{vk}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, \varnothing) \\ \mathsf{dig} \leftarrow \mathsf{Hash}(\mathsf{hk}, x) \\ \sigma \leftarrow \mathsf{ProveOpen}(\mathsf{hk}, x, i)\end{array}\right] = 1.$$

  - **Digest correctness:**

$$\Pr\left[\mathsf{ValidateDigest}(\mathsf{vk}, \mathsf{dig}) = 1 : \begin{array}{c}(\mathsf{hk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, \varnothing) \\ \mathsf{dig} \leftarrow \mathsf{Hash}(\mathsf{hk}, x)\end{array}\right] = 1.$$

We additionally require the following security properties:

- **Set hiding:** For a bit $b \in \{0,1\}$ and an adversary $\mathcal{A}$, we define the set hiding game $\mathsf{ExptSH}_{\mathcal{A}}(\lambda, b)$ as follows:
  1. On input 1^λ, the adversary $\mathcal{A}$ outputs 1^n and a set $S \subseteq [n]$.
  2. If $b = 0$, the challenger samples $(\mathsf{hk}, \mathsf{vk}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, \varnothing)$ and if $b = 1$, the challenger samples $(\mathsf{hk}, \mathsf{vk}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, S)$. It gives $(\mathsf{hk}, \mathsf{vk})$ to $\mathcal{A}$.
  3. Algorithm $\mathcal{A}$ outputs a bit b' which is the output of the experiment.

  The hash function satisfies set binding if for all efficient adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that

$$\left|\Pr[\mathsf{ExptSH}_{\mathcal{A}}(\lambda, 0) = 1] - \Pr[\mathsf{ExptSH}_{\mathcal{A}}(\lambda, 1) = 1]\right| = \mathsf{negl}(\lambda).$$

- **Index hiding with extracted guess:** For an adversary $\mathcal{A}$ and a bit $b \in \{0,1\}$, we define the index hiding with extracted guess game $\mathsf{ExptIHE}_{\mathcal{A}}(\lambda, b)$ as follows:
  1. On input 1^λ, algorithm $\mathcal{A}$ outputs 1^n, a set $S \subseteq [n]$, and an index $i^* \in S$.
  2. If $b = 0$, the challenger samples $(\mathsf{hk}, \mathsf{vk}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, S \setminus \{i^*\})$. Otherwise, it samples $(\mathsf{hk}, \mathsf{vk}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, S)$. The challenger sends $(\mathsf{hk}, \mathsf{vk})$ to $\mathcal{A}$.
  3. Algorithm $\mathcal{A}$ outputs a digest dig and an opening σ.
  4. The output of the experiment is 1 if $\mathsf{VerOpen}(\mathsf{hk}, \mathsf{dig}, i^*, 0, \sigma) = 1$ and $\mathsf{Extract}(\mathsf{td}, \mathsf{dig})$ outputs Matching. Otherwise, the output is 0.

  The hash function satisfies index hiding with extracted guess if for all efficient adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that

$$\left|\Pr[\mathsf{ExptIHE}_{\mathcal{A}}(\lambda, 0) = 1] - \Pr[\mathsf{ExptIHE}_{\mathcal{A}}(\lambda, 1) = 1]\right| = \mathsf{negl}(\lambda).$$

- **Selective zero fixing:** For an adversary $\mathcal{A}$, we define the adaptive zero-fixing game $\mathsf{ExptZF}_{\mathcal{A}}(\lambda)$ as follows:
  1. On input 1^λ, algorithm $\mathcal{A}$ outputs 1^n, a set $S \subseteq [n]$ and an index $i \in S$.
  2. The challenger samples $(\mathsf{hk}, \mathsf{vk}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, S)$ and gives $(\mathsf{hk}, \mathsf{vk})$ to $\mathcal{A}$.
  3. Algorithm $\mathcal{A}$ outputs a digest dig and an opening σ.
  4. The output of the experiment is 1 if $\mathsf{Extract}(\mathsf{td}, \mathsf{dig})$ outputs Matching and $\mathsf{VerOpen}(\mathsf{hk}, \mathsf{dig}, i, 1, \sigma) = 1$. Otherwise, the output is 0.

  The hash function satisfies zero-fixing if for all efficient adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\Pr[\mathsf{ExptZF}_{\mathcal{A}}(\lambda) = 1] = \mathsf{negl}(\lambda)$.

- **Extractor validity:** For an adversary $\mathcal{A}$, we define the extractor validity game $\mathsf{ExptEV}_{\mathcal{A}}(\lambda)$ as follows:

1. On input 1^λ, the adversary $\mathcal{A}$ outputs 1^n.
2. The challenger samples $(\mathsf{hk}, \mathsf{vk}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, \varnothing)$ and sends hk to the adversary.
3. Algorithm $\mathcal{A}$ outputs a digest dig.
4. The output of the experiment is 1 if $\mathsf{ValidateDigest}(\mathsf{hk}, \mathsf{dig}) = 1$ and $\mathsf{Extract}(\mathsf{td}, \mathsf{dig}) = \mathsf{NotMatching}$. Otherwise, the output is 0.

The hash function satisfies extractor validity if for every efficient adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\Pr[\mathsf{ExptEV}_{\mathcal{A}}(\lambda) = 1] = \mathsf{negl}(\lambda)$.

Remark 3.10 (Adaptive Zero-Fixing Security)**.** We can define a stronger adaptive notion of zero-fixing security where the adversary outputs the index $i \in S$ with the digest and the opening, instead of at the beginning of the security game (i.e., after seeing hk and vk). As argued in [NWW24], those two notions are equivalent. When constructing zero-fixing hash (as in Construction 4.2), it is easier to work with the simpler selective definition.

One-sided index hiding. For our application, it suffices to consider a weaker notion of "one-sided" index hiding where we only require that the adversary's advantage cannot increase (but could decrease). Proving one-sided security is often easier than proving two-sided security, so we define the simpler notion here:

Definition 3.11 (One-Sided Index-Hiding with Extracted Guess)**.** We say a zero-fixing hash function Π_{H} satisfies *one-sided* index-hiding with extracted guess security if for all efficient adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that

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

4 Construction of Zero-Fixing Hash Functions

In this section, we show how to construct a zero-fixing hash function by combining an index BARG (Definition 3.8), an additively homomorphic encryption scheme over $\mathbb{Z}_p$ (Definition 3.1), and a vector commitment scheme with succinct local openings (Definition 3.3).

Binary tree indexing. Similar to [NWW24], we will work with complete binary trees. Following [NWW24], we use the following procedure to associate a unique index with each node in the binary tree:

Definition 4.1 (Binary Tree Indexing)**.** Let $\mathcal{T}$ be a complete binary tree with $n = 2^k$ leaves. Then $\mathcal{T}$ contains exactly $2n - 1$ nodes. We associate a unique index $i \in [2n - 1]$ via the following procedure:

- First, associate the value $v = 1$ to the root node.
- If v is the value associated with a node, then associate values $2v$ and $2v+1$ with its left and right child. Recursively apply this process to assign a value to every node in the tree.
- The index i associated with a node is defined to be $2n - v$, where v is the value associated with the node.

By design, Definition 4.1 has the following properties:

- The leaf nodes are indexed 1 through n and the root node is indexed $2n - 1$.
- The index of every non-leaf node is greater than the index of its children.
- Given the index of any non-leaf node, we can efficiently compute the indices of its left and right child.

Construction 4.2 (Zero-Fixing Hash Function)**.** Our construction will rely on the following building blocks:

- Let $\Pi_{\mathsf{BARG}} = (\mathsf{BARG.Gen}, \mathsf{BARG.Prove}, \mathsf{BARG.Verify}, \mathsf{BARG.TrapGen}, \mathsf{BARG.Extract})$ be a somewhere extractable index BARG (Definition 3.8).

- Take any constant $p \in \mathbb{N}$. Let $\Pi_{\mathsf{HE}} = (\mathsf{HE.Gen}, \mathsf{HE.Enc}, \mathsf{HE.Dec}, \mathsf{HE.Add})$ be an additively homomorphic encryption scheme over $\mathbb{Z}_p$ (Definition 3.1). For a security parameter λ, let $\ell_{\mathsf{ct}}(\lambda)$ be a bound on the length of the ciphertexts output by either $\mathsf{HE.Enc}(\mathsf{pk}, \cdot)$ or $\mathsf{HE.Add}(\mathsf{pk}, \cdot, \cdot)$ for any $(\mathsf{sk}, \mathsf{pk})$ in the support of $\mathsf{HE.Gen}(1^\lambda)$.
- Let $\Pi_{\mathsf{Com}} = (\mathsf{Com.Setup}, \mathsf{Com.Commit}, \mathsf{Com.Verify})$ be a vector commitment scheme with succinct local openings (Definition 3.3).

We construct a zero-fixing hash $\Pi_{\mathsf{H}} = (\mathsf{Setup}, \mathsf{Hash}, \mathsf{ProveOpen}, \mathsf{VerOpen}, \mathsf{Extract}, \mathsf{ValidateDigest})$. In the following description, we assume without loss of generality that the bound on the input length $n \in \mathbb{N}$ is a power of two (i.e., $n = 2^k$ for some integer $k \in \mathbb{N}$). Next, we define the following NP relation which we will be using in our construction. In what follows, all of the ciphertext vectors have length λ.

Statement: index $i \in [n]$
Witness: ciphertext vectors $\hat{\mathbf{ct}}^{(0)}, \hat{\mathbf{ct}}^{(1)}$, openings $\sigma^{(0)}, \sigma^{(1)}$, and an auxiliary witness $\tilde{w}$
Hardcoded: the common reference string $\mathsf{crs}_{\mathsf{Com}}$ for Π_{Com}, an index $i^* \in [n] \cup \{\perp\}$, a value $y \in \{0, 1, \perp\}$, and for each $b \in \{0, 1\}$, a public key pk_b for Π_{HE}, commitments $\mathsf{com}_{\mathsf{hk}}^{(b)}$ and $\mathsf{com}^{(b)}$ and two ciphertext vectors $\mathbf{ct}_{\mathsf{zero}}^{(b)}, \mathbf{ct}_{\mathsf{root}}^{(b)}$

On input a statement $i \in [n]$ and a witness $\left(\hat{\mathbf{ct}}^{(0)}, \hat{\mathbf{ct}}^{(1)}, \sigma^{(0)}, \sigma^{(1)}, \tilde{w}\right)$:

- If $i \in [n]$, then parse $\tilde{w} = \left(\tilde{\mathbf{ct}}^{(0)}, \tilde{\mathbf{ct}}^{(1)}, \sigma_{\mathsf{hk}}^{(0)}, \sigma_{\mathsf{hk}}^{(1)}\right)$. Output 1 if the following conditions hold:
  1. **Opening to ciphertext:** for $b \in \{0, 1\}$, $\mathsf{Com.Verify}\left(\mathsf{crs}_{\mathsf{Com}}, \mathsf{com}_b, i, \hat{\mathbf{ct}}^{(b)}, \sigma^{(b)}\right) = 1$.
  2. **Opening to ciphertext in** hk**:** for $b \in \{0, 1\}$, $\mathsf{Com.Verify}\left(\mathsf{crs}_{\mathsf{Com}}, \mathsf{com}_{\mathsf{hk}}^{(b)}, i, \tilde{\mathbf{ct}}^{(b)}, \sigma_{\mathsf{hk}}^{(b)}\right) = 1$.
  3. **Consistent choice of ciphertexts:** $\left(\hat{\mathbf{ct}}^{(0)} = \mathbf{ct}_{\mathsf{zero}}^{(0)} \wedge \hat{\mathbf{ct}}^{(1)} = \mathbf{ct}_{\mathsf{zero}}^{(1)}\right)$ or $\left(\hat{\mathbf{ct}}^{(0)} = \tilde{\mathbf{ct}}^{(0)} \wedge \hat{\mathbf{ct}}^{(1)} = \tilde{\mathbf{ct}}^{(1)}\right)$.
  4. **Validity of ciphertext at target index:** If $i = i^*$, then *additionally* check that:
  $$\hat{\mathbf{ct}}^{(b)} = \begin{cases} \mathbf{ct}_{\mathsf{zero}}^{(b)} & \text{if } y = 0 \\ \tilde{\mathbf{ct}}^{(b)} & \text{if } y = 1. \end{cases}$$

  If any of these conditions are not satisfied, output 0.
- If $i \in [n+1, 2n-1]$, then parse $\tilde{w} = (\tilde{w}_{\mathrm{L}}, \tilde{w}_{\mathrm{R}})$, where $\tilde{w}_d = \left(\hat{\mathbf{ct}}_d^{(0)}, \hat{\mathbf{ct}}_d^{(1)}, \sigma_d^{(0)}, \sigma_d^{(1)}\right)$ for $d \in \{\mathrm{L}, \mathrm{R}\}$. Output 1 if all of the following conditions hold for all $b \in \{0, 1\}$.
  1. **Opening to ciphertext:** $\mathsf{Com.Verify}\left(\mathsf{crs}_{\mathsf{Com}}, \mathsf{com}_b, i, \hat{\mathbf{ct}}^{(b)}, \sigma^{(b)}\right) = 1$.
  2. **Opening to child ciphertexts:** $\mathsf{Com.Verify}(\mathsf{crs}_{\mathsf{Com}}, \mathsf{com}_b, i_{\mathrm{L}}, \hat{\mathbf{ct}}_{\mathrm{L}}^{(b)}, \sigma_{\mathrm{L}}^{(b)}) = 1$ and $\mathsf{Com.Verify}(\mathsf{crs}_{\mathsf{Com}}, \mathsf{com}_b, i_{\mathrm{R}}, \hat{\mathbf{ct}}_{\mathrm{R}}^{(b)}, \sigma_{\mathrm{R}}^{(b)}) = 1$, where i_{L} and i_{R} are the indices of the left and right child of i (according to the indexing scheme from Definition 4.1).
  3. **Correctness of evaluation:** $\hat{\mathbf{ct}}^{(b)} = \mathsf{Add}\left(\mathsf{pk}_b, \hat{\mathbf{ct}}_{\mathrm{L}}^{(b)}, \hat{\mathbf{ct}}_{\mathrm{R}}^{(b)}\right)$.
  4. **Validity of root:** If $i = 2n - 1$ then $\hat{\mathbf{ct}}^{(b)} = \mathbf{ct}_{\mathsf{root}}^{(b)}$.

  If any of these conditions are not satisfied, output 0.

Figure 1: The relation $\mathcal{R}\left[\mathsf{crs}_{\mathsf{Com}}, \left\{\mathsf{pk}_b, \mathsf{com}_{\mathsf{hk}}^{(b)}, \mathsf{com}_b, \mathbf{ct}_{\mathsf{zero}}^{(b)}, \mathbf{ct}_{\mathsf{root}}^{(b)}\right\}_{b \in \{\mathsf{main}, \mathsf{shadow}\}}, i^*, y\right]$.

We describe our construction below:

- $\mathsf{Setup}(1^\lambda, 1^n, S)$: On input a security parameter λ, the input length $n = 2^k$ and a set $S \subseteq [n]$, the setup algorithm start by sampling the following:

– Sample two key pairs: $(\mathsf{sk}_{\mathsf{main}}, \mathsf{pk}_{\mathsf{main}}) \leftarrow \mathsf{HE.Gen}(1^\lambda, 1^n)$ and $(\mathsf{sk}_{\mathsf{shadow}}, \mathsf{pk}_{\mathsf{shadow}}) \leftarrow \mathsf{HE.Gen}(1^\lambda, 1^n)$.
– Sample the CRS for the commitment scheme with block length $\lambda \cdot \ell_{\mathsf{ct}}(\lambda)$ and up to $2n-1$ blocks: $\mathsf{crs}_{\mathsf{Com}} \leftarrow \mathsf{Com.Setup}(1^\lambda, 1^{\lambda \cdot \ell_{\mathsf{ct}}(\lambda)}, 1^{2n-1})$.
– Sample the CRS for an index BARG (that supports extractability on up to 3 positions): $(\mathsf{crs}_{\mathsf{BARG}}, \mathsf{vk}_{\mathsf{BARG}}) \leftarrow \mathsf{BARG.Gen}(1^\lambda, 1^{2n-1}, 1^s, 1^3)$, where s is a bound on the size of the circuit computing the index relation from Fig. 1.

Next, for each $b \in \{\mathsf{main}, \mathsf{shadow}\}$, construct an encryption of 0: $\mathbf{ct}^{(b)}_{\mathsf{zero}} \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_b, \mathbf{0})$ where $\mathbf{0}$ is a zero vector of length λ. For all $i \in [n]$, sample a random $\mathbf{v}_i \xleftarrow{R} \{0,1\}^\lambda \setminus \{\mathbf{0}\}$. For each $i \in [n]$ and $b \in \{\mathsf{main}, \mathsf{shadow}\}$, compute the following:

– If $i \in S$, compute $\mathbf{ct}^{(b)}_i \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_b, \mathbf{v}_i)$.
– If $i \notin S$, compute $\mathbf{ct}^{(b)}_i \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_b, \mathbf{0})$.

Next, the setup algorithm constructs a commitment to the ciphertexts associated with the hash key. Specifically, for each $b \in \{\mathsf{main}, \mathsf{shadow}\}$, it computes

$$\left(\mathsf{com}^{(b)}_{\mathsf{hk}}, \sigma^{(b)}_{\mathsf{hk},1}, \ldots, \sigma^{(b)}_{\mathsf{hk},n}\right) \leftarrow \mathsf{Com.Commit}\left(\mathsf{crs}_{\mathsf{Com}}, \left(\mathbf{ct}^{(b)}_1, \ldots, \mathbf{ct}^{(b)}_n\right)\right).$$

Finally, the setup algorithm constructs the hash key hk, the verification key vk, and the trapdoor td as follows:

$$\mathsf{hk} = \left(\mathsf{crs}_{\mathsf{Com}}, \mathsf{crs}_{\mathsf{BARG}}, \left\{\mathsf{pk}_b, \mathbf{ct}^{(b)}_{\mathsf{zero}}, \mathbf{ct}^{(b)}_1, \ldots, \mathbf{ct}^{(b)}_n, \sigma^{(b)}_{\mathsf{hk},1}, \ldots, \sigma^{(b)}_{\mathsf{hk},n}\right\}_{b \in \{\mathsf{main},\mathsf{shadow}\}}\right) \tag{4.1}$$

$$\mathsf{vk} = \left(\mathsf{crs}_{\mathsf{Com}}, \mathsf{vk}_{\mathsf{BARG}}, \left\{\mathsf{pk}_b, \mathbf{ct}^{(b)}_{\mathsf{zero}}, \mathsf{com}^{(b)}_{\mathsf{hk}}\right\}_{b \in \{\mathsf{main},\mathsf{shadow}\}}\right) \tag{4.2}$$

$$\mathsf{td} = \mathsf{sk}_{\mathsf{main}}. \tag{4.3}$$

- $\mathsf{Hash}(\mathsf{hk}, x)$: On input a hash key hk (parsed as in Eq. (4.1)) and a string $x \in \{0,1\}^n$, the hashing algorithm proceeds as follows:

  – Construct two complete binary trees $\mathcal{T}_{\mathsf{main}}, \mathcal{T}_{\mathsf{shadow}}$, each with n leaves. For each tree $\mathcal{T}_b$, we assign a ciphertext vector $\hat{\mathbf{ct}}^{(b)}_i$ to each node $i \in [2s-1]$ in the tree as follows (where the nodes are indexed using Definition 4.1):
    * If $i \in [n]$, let $\hat{\mathbf{ct}}^{(b)}_i \leftarrow \mathbf{ct}^{(b)}_{\mathsf{zero}}$ if $x_i = 0$ and $\hat{\mathbf{ct}}^{(b)}_i \leftarrow \mathbf{ct}^{(b)}_i$ if $x_i = 1$.
    * For each internal node $i \in [n, 2n-1]$, let $\hat{\mathbf{ct}}^{(b)}_i = \mathsf{HE.Add}\left(\mathsf{pk}_b, \hat{\mathbf{ct}}^{(b)}_{i_{\mathrm{L}}}, \hat{\mathbf{ct}}^{(b)}_{i_{\mathrm{R}}}\right)$, where i_{L} and i_{R} are the indices associated with the left and right child of node i under the canonical tree indexing scheme (Definition 4.1).
  – For $b \in \{\mathsf{main}, \mathsf{shadow}\}$, construct commitments to the ciphertexts associated with $\mathcal{T}_b$:

$$\left(\mathsf{com}_b, \sigma^{(b)}_1, \ldots, \sigma^{(b)}_{2n-1}\right) \leftarrow \mathsf{Com.Commit}\left(\mathsf{crs}_{\mathsf{Com}}, \left(\hat{\mathbf{ct}}^{(b)}_1, \ldots, \hat{\mathbf{ct}}^{(b)}_{2n-1}\right)\right)$$

  – For $b \in \{\mathsf{main}, \mathsf{shadow}\}$, let $\mathbf{ct}^{(b)}_{\mathsf{root}} = \hat{\mathbf{ct}}^{(b)}_{2n-1}$ (i.e., the ciphertext vector associated with the root of $\mathcal{T}_b$). Let $C_\perp$ be the circuit that computes the following instantiation of the relation from Fig. 1:

$$\mathcal{R}\left[\mathsf{crs}_{\mathsf{Com}}, \left\{\mathsf{pk}_b, \mathsf{com}^{(b)}_{\mathsf{hk}}, \mathsf{com}_b, \mathbf{ct}^{(b)}_{\mathsf{zero}}, \mathbf{ct}^{(b)}_{\mathsf{root}}\right\}_{b \in \{\mathsf{main},\mathsf{shadow}\}}, \perp, \perp\right].$$

  – For each $i \in [2n-1]$, let $\tau_i = \left(\hat{\mathbf{ct}}^{\mathsf{main}}_i, \hat{\mathbf{ct}}^{\mathsf{shadow}}_i, \sigma^{\mathsf{main}}_i, \sigma^{\mathsf{shadow}}_i\right)$ be the opening for the ciphertext vectors associated with node i in $\mathcal{T}_{\mathsf{main}}$ and $\mathcal{T}_{\mathsf{shadow}}$. Then, for each $i \in [2s-1]$, define the auxiliary witness $\tilde{w}_i$ to be
    * If $i \in [n]$ then $\tilde{w}_i = \left(\mathbf{ct}^{\mathsf{main}}_i, \mathbf{ct}^{\mathsf{shadow}}_i, \sigma^{\mathsf{main}}_{\mathsf{hk},i}, \sigma^{\mathsf{shadow}}_{\mathsf{hk},i}\right)$.

* If $i \in [n+1, 2n-1]$ then $\tilde{w}_i = (\tau_{i_L}, \tau_{i_R})$ where i_L, i_R are the indices of the left and right child of node i, respectively.

Finally, $\forall i \in [2n-1]$ let $\hat{w}_i = (\tau_i, \tilde{w}_i)$. Compute $\pi_{\mathsf{BARG}} \leftarrow \mathsf{BARG.Prove}(\mathsf{crs}_{\mathsf{BARG}}, C_\perp, 2n-1, (\hat{w}_1, \ldots, \hat{w}_{2n-1}))$.

– Output the digest

$$\mathsf{dig} = \left(\mathbf{ct}^{\mathsf{main}}_{\mathsf{root}}, \mathbf{ct}^{\mathsf{shadow}}_{\mathsf{root}}, \mathsf{com}_{\mathsf{main}}, \mathsf{com}_{\mathsf{shadow}}, \pi_{\mathsf{BARG}}\right).$$

- $\mathsf{ProveOpen}(\mathsf{hk}, x, i^*)$: On input a hash key hk (parsed as in Eq. (4.1)), a string $x \in \{0,1\}^n$ and an index $i^* \in [n]$, the opening algorithm proceeds as follows:

  – Let $C_{i^*, x_{i^*}}$ be the circuit the following instantiation of the relation from Fig. 1:

  $$\mathcal{R}\left[\mathsf{crs}_{\mathsf{Com}}, \left\{\mathsf{pk}_b, \mathsf{com}^{(b)}_{\mathsf{hk}}, \mathsf{com}_b, \mathbf{ct}^{(b)}_{\mathsf{zero}}, \mathbf{ct}^{(b)}_{\mathsf{root}}\right\}_{b\in\{\mathsf{main},\mathsf{shadow}\}}, i^*, x_{i^*}\right].$$

  – Compute the witnesses $\hat{w}_i$ for each $i \in [2n-1]$ using the same procedure as in the Hash algorithm.

  – Output the opening $\sigma \leftarrow \mathsf{BARG.Prove}(\mathsf{crs}_{\mathsf{BARG}}, C_{i^*, x_{i^*}}, 2n-1, (\hat{w}_1, \ldots, \hat{w}_{2n-1}))$

- $\mathsf{VerOpen}(\mathsf{vk}, \mathsf{dig}, i, \beta, \sigma)$: On input the verification key vk (parsed according to Eq. (4.2)), a digest $\mathsf{dig} = \left(\mathbf{ct}^{(0)}_{\mathsf{root}}, \mathbf{ct}^{(1)}_{\mathsf{root}}, \mathsf{com}_0, \mathsf{com}_1, \pi_{\mathsf{BARG}}\right)$, an index $i^* \in [n]$, a bit $\beta \in \{0,1\}$ and an opening σ, the verification algorithm outputs $\mathsf{BARG.Verify}(\mathsf{crs}_{\mathsf{BARG}}, C_{i^*,\beta}, 2n-1, \sigma)$ where $C_{i^*,\beta}$ is the circuit computing the following relation from Fig. 1:

  $$\mathcal{R}\left[\mathsf{crs}_{\mathsf{Com}}, \left\{\mathsf{pk}_b, \mathsf{com}^{(b)}_{\mathsf{hk}}, \mathsf{com}_b, \mathbf{ct}^{(b)}_{\mathsf{zero}}, \mathbf{ct}^{(b)}_{\mathsf{root}}\right\}_{b\in\{\mathsf{main},\mathsf{shadow}\}}, i^*, \beta\right].$$

- $\mathsf{Extract}(\mathsf{td}, \mathsf{dig})$: On input a trapdoor $\mathsf{td} = \mathsf{sk}_{\mathsf{main}}$ and a digest $\mathsf{dig} = \left(\mathbf{ct}^{\mathsf{main}}_{\mathsf{root}}, \mathbf{ct}^{\mathsf{shadow}}_{\mathsf{root}}, \mathsf{com}_{\mathsf{main}}, \mathsf{com}_{\mathsf{shadow}}, \pi_{\mathsf{BARG}}\right)$, the extraction algorithm outputs Matching if $\mathsf{HE.Dec}(\mathsf{sk}_{\mathsf{main}}, \mathbf{ct}^{\mathsf{main}}_{\mathsf{root}}) = \mathbf{0}$. Otherwise, it outputs NotMatching.

- $\mathsf{ValidateDigest}(\mathsf{vk}, \mathsf{dig})$: On input the verification key vk (parsed according to Eq. (4.2)) and a digest $\mathsf{dig} = \left(\mathbf{ct}^{\mathsf{main}}_{\mathsf{root}}, \mathbf{ct}^{\mathsf{shadow}}_{\mathsf{root}}, \mathsf{com}_{\mathsf{main}}, \mathsf{com}_{\mathsf{shadow}}, \pi_{\mathsf{BARG}}\right)$, the digest-validation algorithm outputs

  $$\mathsf{BARG.Verify}(\mathsf{vk}_{\mathsf{BARG}}, C_\perp, 2n-1, \pi_{\mathsf{BARG}}),$$

  where $C_\perp$ is the circuit computing the following relation from Fig. 1:

  $$\mathcal{R}\left[\mathsf{crs}_{\mathsf{Com}}, \left\{\mathsf{pk}_b, \mathsf{com}^{(b)}_{\mathsf{hk}}, \mathsf{com}_b, \mathbf{ct}^{(b)}_{\mathsf{zero}}, \mathbf{ct}^{(b)}_{\mathsf{root}}\right\}_{b\in\{\mathsf{main},\mathsf{shadow}\}}, \perp, \perp\right].$$

Theorem 4.3 (Correctness). *Construction 4.2 is correct.*

Proof. Take any $\lambda, n \in \mathbb{N}$ and $x \in \{0,1\}^n$. Suppose $(\mathsf{hk}, \mathsf{vk}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, \varnothing)$. Parse

$$\mathsf{hk} = \left(\mathsf{crs}_{\mathsf{Com}}, \mathsf{crs}_{\mathsf{BARG}}, \left\{\mathsf{pk}_b, \mathsf{ct}^{(b)}_{\mathsf{zero}}, \mathbf{ct}^{(b)}_1, \ldots, \mathbf{ct}^{(b)}_n, \sigma^{(b)}_{\mathsf{hk},1}, \ldots, \sigma^{(b)}_{\mathsf{hk},n}\right\}_{b\in\{\mathsf{main},\mathsf{shadow}\}}\right)$$
$$\mathsf{vk} = \left(\mathsf{crs}_{\mathsf{Com}}, \mathsf{vk}_{\mathsf{BARG}}, \left\{\mathsf{pk}_b, \mathbf{ct}^{(b)}_{\mathsf{zero}}, \mathsf{com}^{(b)}_{\mathsf{hk}}\right\}_{b\in\{\mathsf{main},\mathsf{shadow}\}}\right)$$
$$\mathsf{td} = \mathsf{sk}_{\mathsf{main}}.$$

We show each property individually.

Digest validity. Let $\mathsf{dig} \leftarrow \mathsf{Hash}(\mathsf{hk}, x)$. By construction, $\mathsf{dig} = (\mathbf{ct}^{\mathsf{main}}_{\mathsf{root}}, \mathbf{ct}^{\mathsf{shadow}}_{\mathsf{root}}, \mathsf{com}_{\mathsf{main}}, \mathsf{com}_{\mathsf{shadow}}, \pi_{\mathsf{BARG}})$ where $\pi_{\mathsf{BARG}} \leftarrow \mathsf{BARG.Prove}(\mathsf{crs}_{\mathsf{BARG}}, C_\perp, 2n-1, (\hat{w}_1, \ldots, \hat{w}_{2n-1}))$ and $C_\perp$ is the circuit computing the relation

$$\mathcal{R}\left[\mathsf{crs}_{\mathsf{Com}}, \left\{\mathsf{pk}_b, \mathsf{com}^{(b)}_{\mathsf{hk}}, \mathsf{com}_b, \mathbf{ct}^{(b)}_{\mathsf{zero}}, \mathbf{ct}^{(b)}_{\mathsf{root}}\right\}_{b\in\{\mathsf{main},\mathsf{shadow}\}}, \perp, \perp\right]$$

from Fig. 1. Parse $\hat{w}_i = (\tau_i, \tilde{w}_i)$ where $\tau_i = (\hat{\mathbf{ct}}^{\mathsf{main}}_i, \hat{\mathbf{ct}}^{\mathsf{shadow}}_i, \sigma^{\mathsf{main}}_i, \sigma^{\mathsf{shadow}}_i)$. We prove that $C_\perp(i, \hat{w}_i) = 1$ for each $i \in [2n-1]$:

- **Leaf nodes:** Suppose $i \in [n]$. Then by construction of Hash, we have $\hat{\mathbf{ct}}_i^{\mathsf{main}} = \mathbf{ct}_i^{\mathsf{main}}$ and $\hat{\mathbf{ct}}_i^{\mathsf{shadow}} = \mathbf{ct}_i^{\mathsf{shadow}}$ and $\tilde{w}_i = (\mathbf{ct}_i^{\mathsf{main}}, \mathbf{ct}_i^{\mathsf{shadow}}, \sigma_{\mathsf{hk},i}^{\mathsf{main}}, \sigma_{\mathsf{hk},i}^{\mathsf{shadow}})$. Consider each of the checks:
  1. **Opening to ciphertext:** by construction of Hash, for each $b \in \{\mathsf{main}, \mathsf{shadow}\}$, the commitment com_b for each $b \in \{\mathsf{main}, \mathsf{shadow}\}$ is a vector commitment to $(\mathbf{ct}_1^{(b)}, \ldots, \mathbf{ct}_{2n-1}^{(b)})$ and the opening $\sigma_i^{(b)}$ is a valid opening for position i. Therefore the check passes.
  2. **Opening to ciphertext in** hk**:** By construction of Setup, for each $b \in \{\mathsf{main}, \mathsf{shadow}\}$, the commitment $\mathsf{com}_{\mathsf{hk}}^{(b)}$ is a vector commitment to $(\mathbf{ct}_1^{(b)}, \ldots, \mathbf{ct}_n^{(b)})$ and the opening $\sigma_{\mathsf{hk},i}^{(b)}$ is a valid opening for position i. Therefore the check passes.
  3. **Consistent choice of ciphertexts:** By construction of Hash, we have that for each $b \in \{\mathsf{main}, \mathsf{shadow}\}$, it holds that $\mathbf{v}_i^{(b)}$ is either $\mathbf{ct}_i^{(b)}$ or $\mathbf{ct}_{\mathsf{zero}}^{(b)}$ depending on the value of x_i. Therefore they are consistent and the check passes.
  4. **Validity of ciphertext at target index:** Since the hash relation does not define a target index, the check passes trivially.
- **Non-leaf nodes:** Suppose $i \in [n+1, 2n-1]$. Then $\tilde{w}_i = (\tau_{i_{\mathsf{L}}}, \tau_{i_{\mathsf{R}}})$, Consider each of the checks:
  1. **Opening to ciphertext:** This follows by the same reason as above.
  2. **Opening to child ciphertexts:** This follows similarly from the fact that for each $b \in \{\mathsf{main}, \mathsf{shadow}\}$, the commitment com_b is a vector commitment to $(\mathbf{ct}_1^{(b)}, \ldots, \mathbf{ct}_{2n-1}^{(b)})$ with openings $\sigma_1^{(b)}, \ldots, \sigma_{2n-1}^{(b)}$.
  3. **Correctness of evaluation:** By construction of Hash, for all $b \in \{\mathsf{main}, \mathsf{shadow}\}$, and all non-leaf nodes, we have that $\mathbf{ct}_i^{(b)} = \mathsf{HE.Add}\big(\mathsf{pk}_b, \mathbf{ct}_{i_{\mathsf{L}}}^{(b)}, \mathbf{ct}_{i_{\mathsf{R}}}^{(b)}\big)$, and so the checks pass (since HE.Add is deterministic).
  4. **Validity of root:** By construction of Hash, for all $b \in \{\mathsf{main}, \mathsf{shadow}\}$ we have that $\mathbf{ct}_{\mathsf{root}}^{(b)} = \mathbf{ct}_{2n-1}^{(b)}$, so the check trivially passes.

Since $C_\perp(i, \hat{w}_i) = 1$ for each $i \in [2n-1]$, then all of the witnesses are correct and π_{BARG} cause BARG.Verify (and by correspondence ValidateDigest) to accept by the completeness of Π'_{BARG}.

Opening correctness. Let $i^* \in [n]$, and suppose $\sigma \leftarrow \mathsf{ProveOpen}(\mathsf{hk}, x, i^*)$. We show that $\mathsf{VerOpen}(\mathsf{vk}, \mathsf{dig}, i, x_{i^*}, \sigma)$ accepts. This follows by an analogous argument, with the one difference being that the BARG proof σ is now computed with respect to the circuit $C_{i^*, x_{i^*}}$ that computes the relation

$$\mathcal{R}\left[\mathsf{crs}_{\mathsf{Com}}, \left\{\mathsf{pk}_b, \mathsf{com}_{\mathsf{hk}}^{(b)}, \mathsf{com}_b, \mathbf{ct}_{\mathsf{zero}}^{(b)}, \mathbf{ct}_{\mathsf{root}}^{(b)}\right\}_{b \in \{\mathsf{main},\mathsf{shadow}\}}, i^*, x_{i^*}\right]$$

from Fig. 1. In other words, the only difference now is that the verification algorithm additionally checks validity at target index. Consider $\hat{w}_{i^*} = (\tau_{i^*}, \tilde{w}_{i^*})$, where τ_{i^*} and $\tilde{w}_{i^*}$ are defined as before. By construction of ProveOpen, for each $b \in \{\mathsf{main}, \mathsf{shadow}\}$, it holds that $\hat{\mathbf{ct}}_{i^*}^{(b)} = \mathbf{ct}_{\mathsf{zero}}^{(b)}$ if $x_{i^*} = 0$ and $\hat{\mathbf{ct}}_{i^*}^{(b)} = \mathbf{ct}_{i^*}^{(b)}$ if $x_{i^*} = 1$, therefore the validity at target index check passes as well. The claim now follows by the completeness of Π'_{BARG} similar to before.

□

Theorem 4.4 (Succinctness). *Construction 4.2 is succinct.*

Proof. Take any $\lambda, n \in \mathbb{N}$ and $x \in \{0,1\}^n$. Let $s \in \mathbb{N}$ be a bound on the size of the circuits computing the relation in Fig. 1. Let $i \in [n]$ be an index. Suppose $(\mathsf{hk}, \mathsf{vk}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, \varnothing)$, $\mathsf{dig} \leftarrow \mathsf{Hash}(\mathsf{hk}, x)$ and $\pi_{\mathsf{open}} \leftarrow \mathsf{ProveOpen}(\mathsf{hk}, x, i)$. Parse

$$\begin{aligned}
\mathsf{hk} &= \left(\mathsf{crs}_{\mathsf{Com}}, \mathsf{crs}_{\mathsf{BARG}}, \left\{\mathsf{pk}_b, \mathbf{ct}_{\mathsf{zero}}^{(b)}, \mathbf{ct}_1^{(b)}, \ldots, \mathbf{ct}_n^{(b)}, \sigma_{\mathsf{hk},1}^{(b)}, \ldots, \sigma_{\mathsf{hk},n}^{(b)}\right\}_{b \in \{\mathsf{main},\mathsf{shadow}\}}\right) \\
\mathsf{vk} &= \left(\mathsf{crs}_{\mathsf{Com}}, \mathsf{vk}_{\mathsf{BARG}}, \left\{\mathsf{pk}_b, \mathbf{ct}_{\mathsf{zero}}^{(b)}, \mathsf{com}_{\mathsf{hk}}^{(b)}\right\}_{b \in \{\mathsf{main},\mathsf{shadow}\}}\right) \\
\mathsf{td} &= \mathsf{sk}_{\mathsf{main}} \\
\mathsf{dig} &= \left(\mathbf{ct}_{\mathsf{root}}^{\mathsf{main}}, \mathbf{ct}_{\mathsf{root}}^{\mathsf{shadow}}, \mathsf{com}_{\mathsf{main}}, \mathsf{com}_{\mathsf{shadow}}, \pi_{\mathsf{dig}}\right)
\end{aligned}$$

All ciphertexts are encryptions of vectors of dimension λ. By the compactness of Π_{HE}, the size of the ciphertexts and the public keys is $\mathsf{poly}(\lambda)$. By the succinctness of Π_{Com}, it holds that $\mathsf{crs}_{\mathsf{Com}}, \mathsf{com}_{\mathsf{hk}}^{\mathsf{main}}, \mathsf{com}_{\mathsf{hk}}^{\mathsf{shadow}}, \mathsf{com}_{\mathsf{main}}$ and $\mathsf{com}_{\mathsf{shadow}}$ all have length $\mathsf{poly}(\lambda + \log n)$. It remains to bound the parameters of the BARG. To do so, we bound s. The relation in Fig. 1 requires a constant number of openings for the ciphertext checks. Each of these can be implemented by a circuit of size $\mathsf{poly}(\lambda)$. Similarly, the correctness of the homomorphic evaluation check and the constant number of ciphertext comparisons also require a circuit of size $\mathsf{poly}(\lambda + \log n)$. Thus, the size s of the circuit in Fig. 1 is bounded by $\mathsf{poly}(\lambda + \log n)$. By succinctness of Π'_{BARG}, it holds that the length of the verification key $\mathsf{vk}_{\mathsf{BARG}}$ and the proofs π_{dig} and π_{open} have size $\mathsf{poly}(\lambda + \log n)$. In total, everything is polynomial in $\mathsf{poly}(\lambda + \log n)$ and therefore all of the succinctness requirements are satisfied by Construction 4.2. □

Security. In the subsequent sections, we prove each of the required security properties on Construction 4.2. Instantiating the underlying additively homomorphic encryption scheme with the Goldwasser-Micali construction [GM82] over $\mathbb{Z}_2$, we obtain the following corollary:

Corollary 4.5 (Zero-Fixing Hash Functions)**.** *Assuming the quadratic residuosity assumption and a somewhere extractable BARG, there exists a zero-fixing hash function.*

In combination with the compiler from [NWW24], this yields Theorem 1.1.

4.1 Set Hiding

We start by showing Construction 4.2 satisfies set hiding. This follows immediately from CPA-security of the underlying encryption scheme. Recall that in Construction 4.2, the only difference between a hash key that binds to the empty set $\varnothing$ versus the set S is that some of the ciphertexts in the hash key switch from encryptions of zero vectors (when binding to the empty set) to an encryptions of non-zero vectors (when binding to the set S). We formalize this below:

Theorem 4.6 (Set Hiding)**.** *If Π_{HE} is CPA-secure, then Construction 4.2 satisfies set hiding.*

Proof. Let $\mathcal{A}$ be an efficient adversary for the set hiding game. For ease of exposition, we treat main and shadow from Construction 4.2 as 0 and 1 respectively. Define the games Hyb_β for each $\beta \in \{0, 1, 2\}$ as follows:

1. On input 1^λ, algorithm $\mathcal{A}$ outputs the input length 1^n and a set $S \subseteq [n]$.
2. The challenger samples the following quantities:
   - $(\mathsf{sk}_0, \mathsf{pk}_0) \leftarrow \mathsf{HE.Gen}(1^\lambda)$ and $(\mathsf{sk}_1, \mathsf{pk}_1) \leftarrow \mathsf{HE.Gen}(1^\lambda)$.
   - $(\mathsf{crs}_{\mathsf{BARG}}, \mathsf{vk}_{\mathsf{BARG}}) \leftarrow \mathsf{Gen}(1^\lambda, 1^{2n}, 1^s, 1^3)$.
   - $\mathsf{crs}_{\mathsf{Com}} \leftarrow \mathsf{Com.Setup}(1^\lambda, 1^{\lambda \cdot \ell_{\mathsf{ct}}(\lambda)}, 2n-1)$.
   - $\mathbf{ct}_{\mathsf{zero}}^{(b)} \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_b, \mathbf{0})$ for all $b \in \{0, 1\}$.
   - For all $i \in [n]$, sample a random $\mathbf{v}_i \xleftarrow{\mathrm{R}} \{0,1\}^\lambda \setminus \{\mathbf{0}\}$.
   - For all $i \in [n], b \in \{0,1\}$, if $i \in S$ and $b < \beta$, the challenger samples $\mathbf{ct}_i^{(b)} \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_b, \mathbf{v}_i)$. Otherwise, if $i \notin S$ or $b \geq \beta$, the challenger samples $\mathbf{ct}_i^{(b)} \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_b, \mathbf{0})$.
   - For each $b \in \{0,1\}$ let $\left(\mathsf{com}_{\mathsf{hk}}^{(b)}, \sigma_{\mathsf{hk},1}^{(b)}, \ldots, \sigma_{\mathsf{hk},n}^{(b)}\right) \leftarrow \mathsf{Com.Commit}\left(\mathsf{crs}_{\mathsf{Com}}, (\mathbf{ct}_1^{(b)}, \ldots, \mathbf{ct}_n^{(b)})\right)$.
3. The challenger constructs the hash key hk and the verification vk as defined in Eqs. (4.1) and (4.2):

$$\mathsf{hk} = \left(\mathsf{crs}_{\mathsf{Com}}, \mathsf{crs}_{\mathsf{BARG}}, \left\{\mathsf{pk}_b, \mathbf{ct}_{\mathsf{zero}}^{(b)}, \mathbf{ct}_1^{(b)}, \ldots, \mathbf{ct}_n^{(b)}, \sigma_{\mathsf{hk},1}^{(b)}, \ldots, \sigma_{\mathsf{hk},n}^{(b)}\right\}_{b \in \{0,1\}}\right)$$
$$\mathsf{vk} = \left(\mathsf{crs}_{\mathsf{Com}}, \mathsf{vk}_{\mathsf{BARG}}, \left\{\mathsf{pk}_b, \mathbf{ct}_{\mathsf{zero}}^{(b)}, \mathsf{com}_{\mathsf{hk}}^{(b)}\right\}_{b \in \{0,1\}}\right)$$

   and gives $(\mathsf{hk}, \mathsf{vk})$ to $\mathcal{A}$.

4. Algorithm $\mathcal{A}$ outputs a bit b' which is the output of the experiment.

Let $\mathsf{Hyb}_\beta(\mathcal{A})$ be the output of Hyb_β with adversary $\mathcal{A}$. Note that by construction $\mathsf{ExptSH}_{\mathcal{A}}(\lambda, 0) \equiv \mathsf{Hyb}_0(\mathcal{A})$ and $\mathsf{ExptSH}_{\mathcal{A}}(\lambda, 1) \equiv \mathsf{Hyb}_2(\mathcal{A})$. We now argue that each adjacent pair of hybrid distributions are computationally indistinguishable.

Claim 4.7. *If Π_{HE} is CPA-secure, then there exists a negligible function $\mathsf{negl}(\cdot)$ such that*

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

Proof. Suppose that $\left|\Pr[\mathsf{Hyb}_1(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_0(\mathcal{A}) = 1]\right| \geq \varepsilon$ for some non-negligible ε. We use $\mathcal{A}$ to construct an efficient CPA-security adversary $\mathcal{B}$ against Π_{HE} as follows:

1. On input 1^λ, algorithm $\mathcal{B}$ runs $\mathcal{A}$ to obtain the input length 1^n and the set $S \subseteq [n]$. Denote $S = \{i_1, \ldots, i_t\}$, where $t = |S|$.
2. The challenger sends the public key pk_0 to $\mathcal{B}$.
3. Algorithm $\mathcal{B}$ samples the following:
   - $(\mathsf{sk}_1, \mathsf{pk}_1) \leftarrow \mathsf{HE.Gen}(1^\lambda)$, $\mathsf{crs}_{\mathsf{Com}} \leftarrow \mathsf{Com.Setup}(1^\lambda, 1^{\lambda \cdot \ell_{\mathsf{ct}}(\lambda)}, 2n-1)$.
   - $(\mathsf{crs}_{\mathsf{BARG}}, \mathsf{vk}_{\mathsf{BARG}}) \leftarrow \mathsf{BARG.Gen}(1^\lambda, 1^{2n-1}, 1^s, 1^3)$.
   - $\mathsf{crs}_{\mathsf{Com}} \leftarrow \mathsf{Com.Setup}(1^\lambda, 1^{\lambda \cdot \ell_{\mathsf{ct}}(\lambda)}, 2n-1)$.
   - For all $i \in [n]$, sample a random $\mathbf{v}_i \xleftarrow{\mathrm{R}} \{0,1\}^\lambda \setminus \{\mathbf{0}\}$.
4. Then, for each $i \in [n]$, algorithm $\mathcal{B}$ does the following:
   - If $i \in S$, then make an encryption query on the pair $(\mathbf{0}, \mathbf{v}_i)$ and receive the ciphertext $\mathbf{ct}_i^*$. Set $\mathbf{ct}_i^{(0)} = \mathsf{ct}_i^*$.
   - If $i \notin S$, set $\mathbf{ct}_i^{(0)} \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_1, \mathbf{0})$.
   - Compute $\mathbf{ct}_i^{(1)} \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_1, \mathbf{0})$.
5. For $b \in \{0,1\}$, algorithm $\mathcal{B}$ computes $\left(\mathsf{com}_{\mathsf{hk}}^{(b)}, \sigma_{\mathsf{hk},1}^{(b)}, \ldots, \sigma_{\mathsf{hk},n}^{(b)}\right) \leftarrow \mathsf{Com.Commit}\left(\mathsf{crs}_{\mathsf{Com}}, (\mathbf{ct}_1^{(b)}, \ldots, \mathbf{ct}_n^{(b)})\right)$.
6. Algorithm $\mathcal{B}$ constructs hk and vk according to Eqs. (4.1) and (4.2):

$$\mathsf{hk} = \left(\mathsf{crs}_{\mathsf{Com}}, \mathsf{crs}_{\mathsf{BARG}}, \left\{\mathsf{pk}_b, \mathbf{ct}_{\mathsf{zero}}^{(b)}, \mathbf{ct}_1^{(b)}, \ldots, \mathbf{ct}_n^{(b)}, \sigma_{\mathsf{hk},1}^{(b)}, \ldots, \sigma_{\mathsf{hk},n}^{(b)}\right\}_{b \in \{0,1\}}\right)$$
$$\mathsf{vk} = \left(\mathsf{crs}_{\mathsf{Com}}, \mathsf{vk}_{\mathsf{BARG}}, \left\{\mathsf{pk}_b, \mathbf{ct}_{\mathsf{zero}}^{(b)}, \mathsf{com}_{\mathsf{hk}}^{(b)}\right\}_{b \in \{0,1\}}\right)$$

   and give $(\mathsf{hk}, \mathsf{vk})$ to $\mathcal{A}$.
7. Algorithm $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$, which $\mathcal{B}$ also outputs.

Observe that if the $\mathbf{ct}_i^*$ are encryptions of $\mathbf{0}$ then $\mathcal{B}$ perfectly simulates Hyb_0. If $\mathbf{ct}_i^*$ are encryptions of $\mathbf{v}_i$, then $\mathcal{B}$ perfectly simulates Hyb_1 for $\mathcal{A}$. We conclude that the advantage of $\mathcal{B}$ is ε. In addition, if $\mathcal{A}$ is efficient then so is $\mathcal{B}$, therefore ε is negligible by the CPA security of Π_{HE}. □

Claim 4.8. *If Π_{HE} is CPA-secure, then there exists a negligible function $\mathsf{negl}(\cdot)$ such that*

$$\left|\Pr[\mathsf{Hyb}_2(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_1(\mathcal{A}) = 1]\right| = \mathsf{negl}(\lambda).$$

Proof. Follows by an analogous argument as the proof of Claim 4.7. The only difference is the reduction algorithm $\mathcal{B}$ sets pk_1 and the ciphertexts $\mathbf{ct}_i^{(1)}$ for $i \in S$ to be the public key and challenge ciphertexts it receives for the CPA challenger, whereas $\mathbf{ct}_i^{(0)}$ is set to be an encryption of $\mathbf{v}_i$ if $i \in S$, or an encryption of $\mathbf{0}$ if $i \notin S$. □

Theorem 4.6 now follows by combining Claims 4.7 and 4.8. □

4.2 Additive Invariants on Ciphertexts

Similar to [NWW24], the remaining security properties of the zero-fixing hash function (zero fixing, extractor validity, and index hiding with extracted guess) will rely on reasoning about various properties on the ciphertext vector associated with the root node in our tree of ciphertexts (i.e., the hash digest). The general strategy to prove these properties is similar. We first establish a certain invariant on the leaf ciphertexts by relying on the fact that they are *honestly* generated by the setup algorithm. Then, we appeal to the security of the BARG and the vector commitment to "propagate" the invariant to the root ciphertext.

We start by recalling the invariants introduced by [NWW24] and extend them in two ways: (1) we define the invariants with respect to a vector of ciphertexts (as opposed to a single ciphertext); and (2) we pass auxiliary input which corresponds to the view of the challenger in the security games.

Definition 4.9 (Tree-Based Additive Invariant on Ciphertext Vectors)**.** Let n be a power of two and let $\Pi_{\mathsf{HE}} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Add})$ be an additively homomorphic encryption scheme over $\mathbb{Z}_p$. We say that an efficiently-computable predicate $P\colon \{0,1\}^* \to \{0,1\}$ is a tree-based additive invariant for Π_{HE} if for all $\lambda, n \in \mathbb{N}$, all key-pairs $(\mathsf{sk}_0, \mathsf{pk}_0)$, $(\mathsf{sk}_1, \mathsf{pk}_1)$ in the support of $\mathsf{Gen}(1^\lambda, 1^n)$, all indices $j, j_{\mathrm{L}}, j_{\mathrm{R}} \in [2n-1]$ where j_{L} and j_{R} are the children of j according to the indexing scheme in Definition 4.1, all ciphertext vectors $\left(\mathbf{ct}_{\mathrm{L}}^{(0)}, \mathbf{ct}_{\mathrm{L}}^{(1)}\right)$, $\left(\mathbf{ct}_{\mathrm{R}}^{(0)}, \mathbf{ct}_{\mathrm{R}}^{(1)}\right)$, and all auxiliary input $z \in \{0,1\}^*$ where

$$P\left(\mathbf{ct}_{\mathrm{L}}^{(0)}, \mathbf{ct}_{\mathrm{L}}^{(1)}, \mathsf{sk}_0, \mathsf{sk}_1, j_{\mathrm{L}}, z\right) = 1 \quad \text{and} \quad P\left(\mathbf{ct}_{\mathrm{R}}^{(0)}, \mathbf{ct}_{\mathrm{R}}^{(1)}, \mathsf{sk}_0, \mathsf{sk}_1, j_{\mathrm{R}}, z\right) = 1,$$

it holds that

$$P\left(\mathbf{ct}_{\mathsf{sum}}^{(0)}, \mathbf{ct}_{\mathsf{sum}}^{(1)}, \mathsf{sk}_0, \mathsf{sk}_1, j, z\right) = 1,$$

where $\mathbf{ct}_{\mathsf{sum}}^{(0)} = \mathsf{Add}\left(\mathsf{pk}_0, \mathbf{ct}_{\mathrm{L}}^{(0)}, \mathbf{ct}_{\mathrm{R}}^{(0)}\right)$ and $\mathbf{ct}_{\mathsf{sum}}^{(1)} = \mathsf{Add}\left(\mathsf{pk}_1, \mathbf{ct}_{\mathrm{L}}^{(1)}, \mathbf{ct}_{\mathrm{R}}^{(1)}\right)$. This implies that if P holds for the two children of a node, then it also holds for the parent node.

One way to view the tree-based invariant is that if an adversary can "break" the invariant on some non-leaf node, then the adversary can also break the invariant on one of children of that node.

Predicate propagation experiment. We now recall the definition of the general predicate propagation experiment from [NWW24], which we use in the analysis of Construction 4.2. This is a general experiment specification that captures the structure of the security definitions for a zero-fixing hash function.

Definition 4.10 (Predicate Propagation Experiment)**.** The predicate propagation experiment for Construction 4.2 is parameterized by the following two components:

- A tree-based additive invariant P (Definition 4.9) for the homomorphic encryption scheme Π_{HE}.
- An efficiently-computable "challenge-derivation" function $\mathsf{DeriveChal}(S, i)$ that takes as input a set $S \subseteq [n]$ and an index $i \in [n]$ and outputs two sets $S_0, S_1 \subseteq [n]$ and an index idx that is either a pair (i^*, y^*) or $\perp$. In the predicate propagation experiment, the sets S_0 and S_1 will determine the distribution of the ciphertexts in the common reference string. The index idx will determine the verification check. Each of the security properties (i.e., zero fixing, extractor validity, and index hiding with extracted guess) will induce a different choice of DeriveChal (to be specified in their respective proofs).

We now define the predicate propagation experiment $\mathsf{Expt}[P, \mathsf{DeriveChal}]$ between a challenger and an adversary $\mathcal{A}$:

1. On input the security parameter 1^λ, algorithm $\mathcal{A}_1$ outputs the input length 1^n, a set $S \subseteq [n]$, and an index $i^* \in S$ (or a special symbol $\perp$).
2. The challenger computes $(S_{\mathsf{main}}, S_{\mathsf{shadow}}, \mathsf{idx}) \leftarrow \mathsf{DeriveChal}(S, i^*)$.
3. The challenger samples the following quantities as in Setup:
   - $(\mathsf{sk}_{\mathsf{main}}, \mathsf{pk}_{\mathsf{main}}) \leftarrow \mathsf{HE.Gen}(1^\lambda, 1^n)$, $(\mathsf{sk}_{\mathsf{shadow}}, \mathsf{pk}_{\mathsf{shadow}}) \leftarrow \mathsf{HE.Gen}(1^\lambda, 1^n)$.
   - $(\mathsf{crs}_{\mathsf{BARG}}, \mathsf{vk}_{\mathsf{BARG}}) \leftarrow \mathsf{BARG.Gen}(1^\lambda, 1^{2n}, 1^s, 1^3)$.

- $\mathsf{crs}_{\mathsf{Com}} \leftarrow \mathsf{Com.Setup}(1^\lambda, 1^{\lambda \cdot \ell_{\mathsf{ct}}(\lambda)}, 2n-1)$.
- $\mathbf{ct}^{(b)}_{\mathsf{zero}} \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_b, \mathbf{0})$ for all $b \in \{\mathsf{main}, \mathsf{shadow}\}$.
- For all $i \in [n]$, sample a random $\mathbf{v}_i \xleftarrow{R} \{0,1\}^\lambda \setminus \{\mathbf{0}\}$.
- For all $i \in [n], b \in \{\mathsf{main}, \mathsf{shadow}\}$, if $i \in S_b$ then sample $\mathbf{ct}^{(b)}_i \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_b, \mathbf{v}_i)$. Otherwise, sample $\mathbf{ct}^{(b)}_i \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_b, \mathbf{0})$.
- For all $b \in \{\mathsf{main}, \mathsf{shadow}\}$, $\left(\mathsf{com}^{(b)}_{\mathsf{hk}}, \sigma^{(b)}_{\mathsf{hk},1}, \ldots, \sigma^{(b)}_{\mathsf{hk},n}\right) \leftarrow \mathsf{Com.Commit}\left(\mathsf{crs}_{\mathsf{Com}}, (\mathbf{ct}^{(b)}_1, \ldots, \mathbf{ct}^{(b)}_n)\right)$.

4. The challenger constructs hk and vk as defined in Eqs. (4.1) and (4.2):

$$\mathsf{hk} = \left(\mathsf{crs}_{\mathsf{Com}}, \mathsf{crs}_{\mathsf{BARG}}, \left\{\mathsf{pk}_b, \mathbf{ct}^{(b)}_{\mathsf{zero}}, \mathbf{ct}^{(b)}_1, \ldots, \mathbf{ct}^{(b)}_n, \sigma^{(b)}_{\mathsf{hk},1}, \ldots, \sigma^{(b)}_{\mathsf{hk},n}\right\}_{b \in \{\mathsf{main},\mathsf{shadow}\}}\right)$$
$$\mathsf{vk} = \left(\mathsf{crs}_{\mathsf{Com}}, \mathsf{vk}_{\mathsf{BARG}}, \left\{\mathsf{pk}_b, \mathbf{ct}^{(b)}_{\mathsf{zero}}, \mathsf{com}^{(b)}_{\mathsf{hk}}\right\}_{b \in \{\mathsf{main},\mathsf{shadow}\}}\right)$$

   The challenger gives $(\mathsf{hk}, \mathsf{vk})$ to $\mathcal{A}$.

5. Algorithm $\mathcal{A}$ outputs a digest $\mathsf{dig} = (\mathbf{ct}^{\mathsf{main}}_{\mathsf{root}}, \mathbf{ct}^{\mathsf{shadow}}_{\mathsf{root}}, \mathsf{com}_{\mathsf{main}}, \mathsf{com}_{\mathsf{shadow}}, \pi_{\mathsf{dig}})$ and a proof π.

6. The output of the experiment is 1 if all of the following conditions hold, and 0 otherwise:

   (a) $\mathsf{BARG.Verify}(\mathsf{crs}_{\mathsf{BARG}}, C_{\mathsf{idx}}, (1, \ldots, 2n-1), \pi) = 1$.

   (b) $P(\mathbf{ct}^{\mathsf{main}}_{\mathsf{root}}, \mathbf{ct}^{\mathsf{shadow}}_{\mathsf{root}}, \mathsf{sk}_{\mathsf{main}}, \mathsf{sk}_{\mathsf{shadow}}, 2n-1, (\mathbf{v}_1, \ldots, \mathbf{v}_n, \mathsf{idx})) = 0$.

   Here, the circuit C_{idx} computes the relation from Fig. 1:

   - If $\mathsf{idx} = (i, y)$, then C_{idx} computes the relation

$$\mathcal{R}\left[\mathsf{crs}_{\mathsf{Com}}, \left\{\mathsf{pk}_b, \mathsf{com}^{(b)}_{\mathsf{hk}}, \mathsf{com}_b, \mathbf{ct}^{(b)}_{\mathsf{zero}}, \mathbf{ct}^{(b)}_{\mathsf{root}}\right\}_{b \in \{\mathsf{main},\mathsf{shadow}\}}, i, y\right].$$

   - If $\mathsf{idx} = \perp$, then C_{idx} computes the relation

$$\mathcal{R}\left[\mathsf{crs}_{\mathsf{Com}}, \left\{\mathsf{pk}_b, \mathsf{com}^{(b)}_{\mathsf{hk}}, \mathsf{com}_b, \mathbf{ct}^{(b)}_{\mathsf{zero}}, \mathbf{ct}^{(b)}_{\mathsf{root}}\right\}_{b \in \{\mathsf{main},\mathsf{shadow}\}}, \perp, \perp\right].$$

   In words, the adversary "wins" the game if it produces a proof π that verifies, but the digest does not satisfy the tree-based additive invariant P.

The goal now is to show that if specific "pre-conditions" are met, then for all efficient adversaries $\mathcal{A}$, the probability that $\mathsf{Expt}[P, \mathsf{DeriveChal}]$ outputs 1 is negligible. These pre-conditions capture properties of the leaf nodes of the tree. To that end, we now define the predicate propagation hybrid experiment $\mathsf{Expt}_j[P, \mathsf{DeriveChal}]$ between a challenger and an adversary $\mathcal{A}$:

Definition 4.11 (Predicate Propagation Hybrid Experiment)**.** Let $j \in \mathbb{N}$ be an index. For a tree-based additive invariant P and a challenge-derivation function DeriveChal, we define the predicate propagation hybrid experiment between a challenger and an adversary $\mathcal{A}$, which we denote by $\mathsf{Expt}_j[P, \mathsf{DeriveChal}]$, as follows:

1. On input the security parameter 1^λ, algorithm $\mathcal{A}_1$ outputs the input length 1^n, a set $S \subseteq [n]$, and an index $i^* \in S$ (or a special symbol $\perp$).

2. The challenger computes $(S_{\mathsf{main}}, S_{\mathsf{shadow}}, \mathsf{idx}) \leftarrow \mathsf{DeriveChal}(S, i^*)$.

3. The challenger samples the following quantities as in Setup:

   - $(\mathsf{sk}_{\mathsf{main}}, \mathsf{pk}_{\mathsf{main}}) \leftarrow \mathsf{HE.Gen}(1^\lambda, 1^n)$, $(\mathsf{sk}_{\mathsf{shadow}}, \mathsf{pk}_{\mathsf{shadow}}) \leftarrow \mathsf{HE.Gen}(1^\lambda, 1^n)$.
   - $(\mathsf{crs}_{\mathsf{BARG}}, \mathsf{vk}_{\mathsf{BARG}}, \mathsf{td}_{\mathsf{BARG}}) \leftarrow \mathsf{TrapGen}(1^\lambda, 1^{2n}, 1^s, 1^3, \{j\})$.

- $\mathsf{crs}_{\mathsf{Com}} \leftarrow \mathsf{Com.Setup}(1^\lambda, 1^{\lambda \cdot \ell_{\mathsf{ct}}(\lambda)}, 2n-1)$.
- $\mathbf{ct}^{(b)}_{\mathsf{zero}} \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_b, \mathbf{0})$ for all $b \in \{\mathsf{main}, \mathsf{shadow}\}$.
- For all $i \in [n]$, sample a random $\mathbf{v}_i \xleftarrow{\mathrm{R}} \{0,1\}^\lambda \setminus \{\mathbf{0}\}$.
- For all $i \in [n], b \in \{\mathsf{main}, \mathsf{shadow}\}$, if $i \in S_b$ then sample $\mathbf{ct}^{(b)}_i \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_b, \mathbf{v}_i)$. Otherwise, sample $\mathbf{ct}^{(b)}_i \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_b, \mathbf{0})$.
- $\left(\mathsf{com}^{(b)}_{\mathsf{hk}}, \sigma^{(b)}_{\mathsf{hk},1}, \ldots, \sigma^{(b)}_{\mathsf{hk},n}\right) \leftarrow \mathsf{Com.Commit}\left(\mathsf{crs}_{\mathsf{Com}}, (\mathbf{ct}^{(b)}_1, \ldots, \mathbf{ct}^{(b)}_n)\right)$ for all $b \in \{\mathsf{main}, \mathsf{shadow}\}$.

4. The challenger constructs hk and vk as defined in Eqs. (4.1) and (4.2):

$$\mathsf{hk} = \left(\mathsf{crs}_{\mathsf{Com}}, \mathsf{crs}_{\mathsf{BARG}}, \left\{\mathsf{pk}_b, \mathbf{ct}^{(b)}_{\mathsf{zero}}, \mathbf{ct}^{(b)}_1, \ldots, \mathbf{ct}^{(b)}_n, \sigma^{(b)}_{\mathsf{hk},1}, \ldots, \sigma^{(b)}_{\mathsf{hk},n}\right\}_{b \in \{\mathsf{main},\mathsf{shadow}\}}\right)$$
$$\mathsf{vk} = \left(\mathsf{crs}_{\mathsf{Com}}, \mathsf{vk}_{\mathsf{BARG}}, \left\{\mathsf{pk}_b, \mathbf{ct}^{(b)}_{\mathsf{zero}}, \mathsf{com}^{(b)}_{\mathsf{hk}}\right\}_{b \in \{\mathsf{main},\mathsf{shadow}\}}\right)$$

   The challenger gives $(\mathsf{hk}, \mathsf{vk})$ to $\mathcal{A}$.

5. Algorithm $\mathcal{A}$ outputs a digest $\mathsf{dig} = (\mathbf{ct}^{\mathsf{main}}_{\mathsf{root}}, \mathbf{ct}^{\mathsf{shadow}}_{\mathsf{root}}, \mathsf{com}_{\mathsf{main}}, \mathsf{com}_{\mathsf{shadow}}, \pi_{\mathsf{dig}})$ and a proof π.

6. The challenger computes $(\hat{\mathbf{ct}}^{\mathsf{main}}_j, \hat{\mathbf{ct}}^{\mathsf{shadow}}_j, \sigma^{\mathsf{main}}_j, \sigma^{\mathsf{shadow}}_j, \tilde{w}_j) \leftarrow \mathsf{BARG.Extract}(\mathsf{td}_{\mathsf{BARG}}, \pi, j)$.

7. The output of the experiment is 1 if all of the following conditions hold, and 0 otherwise:

   (a) $\mathsf{BARG.Verify}(\mathsf{crs}_{\mathsf{BARG}}, C_{\mathsf{idx}}, (1, \ldots, 2n-1), \pi) = 1$.

   (b) $C_{\mathsf{idx}}(j, (\hat{\mathbf{ct}}^{\mathsf{main}}_j, \hat{\mathbf{ct}}^{\mathsf{shadow}}_j, \sigma^{\mathsf{main}}_j, \sigma^{\mathsf{shadow}}_j, \tilde{w}_j)) = 1$.

   (c) $P(\hat{\mathbf{ct}}^{\mathsf{main}}_j, \hat{\mathbf{ct}}^{\mathsf{shadow}}_j, \mathsf{sk}_{\mathsf{main}}, \mathsf{sk}_{\mathsf{shadow}}, j, (\mathbf{v}_1, \ldots, \mathbf{v}_n, \mathsf{idx})) = 0$.

   Here, the circuit C_{idx} computes the relation from Fig. 1:

   - If $\mathsf{idx} = (i, y)$, then C_{idx} computes the relation

$$\mathcal{R}\left[\mathsf{crs}_{\mathsf{Com}}, \left\{\mathsf{pk}_b, \mathsf{com}^{(b)}_{\mathsf{hk}}, \mathsf{com}_b, \mathbf{ct}^{(b)}_{\mathsf{zero}}, \mathbf{ct}^{(b)}_{\mathsf{root}}\right\}_{b \in \{\mathsf{main},\mathsf{shadow}\}}, i, y\right].$$

   - If $\mathsf{idx} = \bot$, then C_{idx} computes the relation

$$\mathcal{R}\left[\mathsf{crs}_{\mathsf{Com}}, \left\{\mathsf{pk}_b, \mathsf{com}^{(b)}_{\mathsf{hk}}, \mathsf{com}_b, \mathbf{ct}^{(b)}_{\mathsf{zero}}, \mathbf{ct}^{(b)}_{\mathsf{root}}\right\}_{b \in \{\mathsf{main},\mathsf{shadow}\}}, \bot, \bot\right].$$

   In words, the adversary "wins" the game if it produces a proof π that verifies, the challenger extracts a correct witness for instance j but the extracted witness does not satisfy the tree-based additive invariant P.

Theorem 4.12 (Predicate Propagation)**.** *Let P be a tree-based additive invariant and let* DeriveChal *be a challenge-derivation function. Suppose Π_{Com} satisfies computational binding and Π_{BARG} satisfies set hiding with extraction, set hiding, and somewhere extractability. Let $\mathcal{A}$ be any efficient adversary for the predicate propagation experiment. Suppose that for every index $j \in [n]$ (where $n = n(\lambda)$ is the input length chosen by $\mathcal{A}$), there exists a negligible function $\varepsilon_j(\cdot)$ such that*

$$\Pr[\mathsf{Expt}_j[P, \mathsf{DeriveChal}](\mathcal{A}) = 1] = \varepsilon_j(\lambda).$$

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

$$\Pr[\mathsf{Expt}[P, \mathsf{DeriveChal}](\mathcal{A}) = 1] = \mathsf{negl}(\lambda).$$

Remark 4.13 (Comparison with [NWW24, Theorem 5.9])**.** Despite the similarities between Theorem 4.12 and [NWW24, Theorem 5.9], there are two reasons we cannot use [NWW24, Theorem 5.9] as a black box. First, while they use additive homomorphic encryption (which captures the scheme Π_{HE}) on a single element, we apply the homomorphic operation on a vector of elements rather. Second, we allow the invariant P to depend on the view of the challenger, by giving it auxiliary input. We give the formal proof of Theorem 4.12 in Appendix A (which shares the same structure as the corresponding proof from [NWW24]).

4.3 Zero Fixing

In this section, we show that Construction 4.2 satisfies zero-fixing security. In the selective zero-fixing game, the adversary chooses a set $S \subseteq [n]$ and an index $i^* \in S$. Then the hash key in Construction 4.2 is chosen to bind to a set S. This means that the ciphertexts in the hash key associated with the set S are replaced by encryptions of non-zero vectors. The adversary is then required to produce a digest dig that is Matching together with an opening of the index i^* to 1.

Intuitively, the BARG in Construction 4.2 guarantees that the digest and opening are computed honestly for some string, so we assume this to be the case in the following discussion. If the size of S is sufficiently large, then there exists a subset $S' \subseteq S$ for which the corresponding vectors are linearly dependent (i.e., they sum to $\mathbf{0}$), and moreover, $i^* \in S'$ with non-negligible probability. Thus, if the adversary knows S', then it can easily construct an "honest" digest and opening that would win the zero-fixing game: choose $x = x_1 \dots x_n$ such that $x_i = 1$ whenever $i \in S'$ (and set $x_i = 0$ otherwise). The adversary can then compute $\mathsf{dig} \leftarrow \mathsf{Hash}(\mathsf{hk}, x)$ and $\sigma \leftarrow \mathsf{ProveOpen}(\mathsf{hk}, x, i^*)$. It is easy to see that $\mathsf{Extract}(\mathsf{vk}, \mathsf{dig}) = \mathsf{Matching}$ since the vectors in S' sum to $\mathbf{0}$ by construction. However, if the vectors $\mathbf{v}_{i^*}$ corresponding to i^* are computationally hidden from the adversary, then it should be infeasible for the adversary to identify a non-trivial set of linearly-dependent vectors. Thus, we show this by relying on CPA-security of the underlying encryption scheme. As noted in Section 2, we use a Naor-Yung approach (with double encryption) for the analysis.

Specifically, starting from the selective zero-fixing game, we first switch to a hybrid where $\mathbf{ct}_{i^*}^{\mathsf{shadow}}$ is an encryption of $\mathbf{0}$ instead of $\mathbf{v}_{i^*}$. Recall that the extraction algorithm ignores the shadow ciphertexts, so these two experiment are computationally indistinguishable. Next, we observe that this erasure of $\mathbf{v}_{i^*}$ gives us an additive invariant: for all nodes in the evaluation tree that do not include i^* in their sub-tree, the main and shadow ciphertexts encrypt the same vector, but for all nodes in the evaluation tree that include i^*, the difference between the vectors encrypted by the main and shadow ciphertexts is $\mathbf{v}_{i^*}$. The consistency condition in the relation guarantees the invariant holds for the leaves, and using Theorem 4.12, we can propagate this invariant to the root node, which includes i^* in its sub-tree. Therefore we can move to another hybrid in which the extraction algorithm outputs Matching if and only if $\mathsf{Dec}(\mathsf{sk}_{\mathsf{shadow}}, \mathbf{ct}_{\mathsf{root}}^{\mathsf{shadow}}) = \mathbf{v}_{i^*}$. Finally, we again use the security of the encryption scheme this time to switch $\mathbf{ct}_{i^*}^{\mathsf{main}}$ to be an encryption of $\mathbf{0}$. In the final experiment, the adversary's view is independent of $\mathbf{v}_{i^*}$, but in order to win, it is required to produce a ciphertext that decrypts to $\mathbf{v}_{i^*}$. The claim holds information theoretically at this point over the random choice of $\mathbf{v}_{i^*}$. We now give the formal proof:

Theorem 4.14 (Zero-Fixing Security)**.** *Suppose Π_{Com} is binding, Π_{BARG} satisfies set hiding with extraction, set hiding and is somewhere extractable, and Π_{HE} is CPA-secure. Then Construction 4.2 satisfies selective zero-fixing.*

Proof. Let $\mathcal{A}$ be an efficient adversary for the zero-fixing game. We define the following hybrid sequence:

- Hyb_0: This is the selective zero-fixing game
  1. On input 1^λ, algorithm $\mathcal{A}_1$ outputs 1^n, a set $S \subseteq [n]$, an index $i^* \in S$ and a state $\mathsf{st}_{\mathcal{A}}$.
  2. The challenger samples the following quantities as in Setup:
     - $(\mathsf{sk}_{\mathsf{main}}, \mathsf{pk}_{\mathsf{main}}) \leftarrow \mathsf{HE.Gen}(1^\lambda, 1^n)$ and $(\mathsf{sk}_{\mathsf{shadow}}, \mathsf{pk}_{\mathsf{shadow}}) \leftarrow \mathsf{HE.Gen}(1^\lambda, 1^n)$.
     - $(\mathsf{crs}_{\mathsf{BARG}}, \mathsf{vk}_{\mathsf{BARG}}) \leftarrow \mathsf{Gen}(1^\lambda, 1^{2n}, 1^s, 1^3)$.
     - $\mathsf{crs}_{\mathsf{Com}} \leftarrow \mathsf{Com.Setup}(1^\lambda, 1^{\lambda \cdot \ell_{\mathsf{ct}}(\lambda)}, 2n-1)$.
     - $\mathbf{ct}_{\mathsf{zero}}^{(b)} \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_b, \mathbf{0})$ for all $b \in \{\mathsf{main}, \mathsf{shadow}\}$.
     - For all $i \in [n]$, sample a random $\mathbf{v}_i \xleftarrow{R} \{0,1\}^\lambda \setminus \{\mathbf{0}\}$.
     - For all $i \in [n]$, if $i \in S$ then sample $\mathbf{ct}_i^{\mathsf{main}} \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_{\mathsf{main}}, \mathbf{v}_i)$. Otherwise sample $\mathbf{ct}_i^{\mathsf{main}} \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_{\mathsf{main}}, \mathbf{0})$.
     - For all $i \in [n]$, if $i \in S$ then sample $\mathbf{ct}_i^{\mathsf{shadow}} \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_{\mathsf{shadow}}, \mathbf{v}_i)$. Otherwise sample $\mathbf{ct}_i^{\mathsf{shadow}} \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_{\mathsf{shadow}}, \mathbf{0})$.
     - $\left(\mathsf{com}_{\mathsf{hk}}^{(b)}, \sigma_{\mathsf{hk},1}^{(b)}, \dots, \sigma_{\mathsf{hk},n}^{(b)}\right) \leftarrow \mathsf{Com.Commit}\left(\mathsf{crs}_{\mathsf{Com}}, (\mathbf{ct}_1^{(b)}, \dots, \mathbf{ct}_n^{(b)})\right)$ for all $b \in \{\mathsf{main}, \mathsf{shadow}\}$.
  3. The challenger constructs the hash key hk and the verification vk as defined in Eqs. (4.1) and (4.2):

$$\mathsf{hk} = \left(\mathsf{crs}_{\mathsf{Com}}, \mathsf{crs}_{\mathsf{BARG}}, \left\{\mathsf{pk}_b, \mathbf{ct}_{\mathsf{zero}}^{(b)}, \mathbf{ct}_1^{(b)}, \dots, \mathbf{ct}_n^{(b)}, \sigma_{\mathsf{hk},1}^{(b)}, \dots, \sigma_{\mathsf{hk},n}^{(b)}\right\}_{b \in \{\mathsf{main},\mathsf{shadow}\}}\right)$$

$$\mathsf{vk} = \left(\mathsf{crs}_{\mathsf{Com}}, \mathsf{vk}_{\mathsf{BARG}}, \left\{\mathsf{pk}_b, \mathbf{ct}_{\mathsf{zero}}^{(b)}, \mathsf{com}_{\mathsf{hk}}^{(b)}\right\}_{b \in \{\mathsf{main},\mathsf{shadow}\}}\right)$$

and gives $(\mathsf{hk}, \mathsf{vk})$ to $\mathcal{A}$.

4. Algorithm $\mathcal{A}$ outputs a digest $\mathsf{dig} = (\mathbf{ct}^{\mathsf{main}}_{\mathsf{root}}, \mathbf{ct}^{\mathsf{shadow}}_{\mathsf{root}}, \mathsf{com}_{\mathsf{main}}, \mathsf{com}_{\mathsf{shadow}}, \pi_{\mathsf{dig}})$ and a proof π.
5. The output of the experiment is 1 if all of the following conditions hold, and 0 otherwise:
   - $\mathsf{BARG.Verify}(\mathsf{crs}_{\mathsf{BARG}}, C_{i^*,1}, (1, \ldots, 2n-1), \pi) = 1$.
   - $\mathsf{HE.Dec}(\mathsf{sk}_{\mathsf{main}}, \mathbf{ct}^{\mathsf{main}}_{\mathsf{root}}) = \mathbf{0}$.

   Here, the circuit $C_{i^*,1}$ computes the relation from Fig. 1:

$$\mathcal{R}\left[\mathsf{crs}_{\mathsf{Com}}, \left\{\mathsf{pk}_b, \mathsf{com}^{(b)}_{\mathsf{hk}}, \mathsf{com}_b, \mathbf{ct}^{(b)}_{\mathsf{zero}}, \mathbf{ct}^{(b)}_{\mathsf{root}}\right\}_{b\in\{\mathsf{main},\mathsf{shadow}\}}, i^*, 1\right].$$

- Hyb_1: Same as Hyb_0, except the challenger replaces the encryption of $\mathbf{v}_{i^*}$ in the shadow branch with an encryption of $\mathbf{0}$. Specifically, during setup, the challenger instead samples $\mathsf{ct}^{\mathsf{shadow}}_{i^*} \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_{\mathsf{shadow}}, \mathbf{0})$.
- Hyb_2: Same as Hyb_1 except the challenger implements extraction by decrypting on the shadow branch instead of the main branch. Specifically, the output of this experiment is 1 if all of the following conditions hold:
  - $\mathsf{BARG.Verify}(\mathsf{crs}_{\mathsf{BARG}}, C_{i^*,1}, (1, \ldots, 2n-1), \pi) = 1$.
  - $\mathsf{HE.Dec}(\mathsf{sk}_{\mathsf{shadow}}, \mathbf{ct}^{\mathsf{shadow}}_{\mathsf{root}}) = \mathbf{v}_{i^*}$.
- Hyb_3: Same as Hyb_2, except the challenger switches the encryption of $\mathbf{v}_{i^*}$ in the main branch to an encryption of $\mathbf{0}$. Specifically, during setup, the challenger samples $\mathsf{ct}^{\mathsf{main}}_{i^*} \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_{\mathsf{main}}, \mathbf{0})$.

Lemma 4.15. *If Π_{HE} is CPA-secure, then $\left|\Pr[\mathsf{Hyb}_0(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_1(\mathcal{A}) = 1]\right| = \mathsf{negl}(\lambda)$.*

Proof. Suppose $\left|\Pr[\mathsf{Hyb}_0(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_1(\mathcal{A}) = 1]\right| = \varepsilon$. We use $\mathcal{A}$ to construct an efficient attacker $\mathcal{B}$ for the CPA security game as follows:

1. On input 1^λ, algorithm $\mathcal{B}$ runs $\mathcal{A}$ to obtain the input length 1^n, a set $S \subseteq [n]$, and an index i^*.
2. The challenger sends the public key $\mathsf{pk}_{\mathsf{shadow}}$ to $\mathcal{B}$.
3. Algorithm $\mathcal{B}$ samples a random $\mathbf{v}_i \xleftarrow{R} \{0,1\}^\lambda \setminus \{\mathbf{0}\}$ for each $i \in [n]$.
4. Algorithm $\mathcal{B}$ sends the challenge $(\mathbf{0}, \mathbf{v}_{i^*})$ to the challenger and gets a ciphertext $\mathbf{ct}^*$.
5. Algorithm $\mathcal{B}$ samples the following:
   - $(\mathsf{sk}_{\mathsf{main}}, \mathsf{pk}_{\mathsf{main}}) \leftarrow \mathsf{HE.Gen}(1^\lambda)$.
   - $(\mathsf{crs}_{\mathsf{BARG}}, \mathsf{vk}_{\mathsf{BARG}}) \leftarrow \mathsf{Gen}(1^\lambda, 1^{2n}, 1^s, 1^3)$.
   - $\mathsf{crs}_{\mathsf{Com}} \leftarrow \mathsf{Com.Setup}(1^\lambda, 1^{\lambda\cdot\ell_{\mathsf{ct}}(\lambda)}, 2n-1)$.
   - $\mathbf{ct}^{(b)}_{\mathsf{zero}} \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_b, \mathbf{0})$ for all $b \in \{\mathsf{main}, \mathsf{shadow}\}$.
   - For all $i \in [n], b \in \{\mathsf{main}, \mathsf{shadow}\}$, if $i \in S$ then sample $\mathbf{ct}^{(b)}_i \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_b, \mathbf{v}_i)$. Otherwise, sample $\mathbf{ct}^{(b)}_i \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_b, \mathbf{0})$.
   - Set $\mathsf{ct}^{\mathsf{shadow}}_{i^*} = \mathsf{ct}^*$.
   - $\left(\mathsf{com}^{(b)}_{\mathsf{hk}}, \sigma^{(b)}_{\mathsf{hk},1}, \ldots, \sigma^{(b)}_{\mathsf{hk},n}\right) \leftarrow \mathsf{Com.Commit}\left(\mathsf{crs}_{\mathsf{Com}}, (\mathbf{ct}^{(b)}_1, \ldots, \mathbf{ct}^{(b)}_n)\right)$ for all $b \in \{\mathsf{main}, \mathsf{shadow}\}$.
6. Algorithm $\mathcal{B}$ constructs the hash key hk and the verification vk as defined in Eqs. (4.1) and (4.2) and runs $\mathcal{A}$ on $(\mathsf{hk}, \mathsf{vk})$ to get (dig, π).
7. Algorithm $\mathcal{B}$ parses $\mathsf{dig} = (\mathbf{ct}^{\mathsf{main}}_{\mathsf{root}}, \mathbf{ct}^{\mathsf{shadow}}_{\mathsf{root}}, \mathsf{com}_{\mathsf{main}}, \mathsf{com}_{\mathsf{shadow}}, \pi_{\mathsf{dig}})$, and outputs 1 if all of the following conditions hold, and 0 otherwise:
   (a) $\mathsf{BARG.Verify}(\mathsf{crs}_{\mathsf{BARG}}, C_{i^*,1}, (1, \ldots, 2n-1), \pi) = 1$.

(b) $\mathsf{HE.Dec}(\mathsf{sk}_{\mathsf{main}}, \mathbf{ct}^{\mathsf{main}}_{\mathsf{root}}) = \mathbf{0}$.

By construction, if ct^* is an encryption of $\mathbf{v}_{i^*}$ then algorithm $\mathcal{B}$ simulates Hyb_0 with attacker $\mathcal{A}$ and if ct^* is an encryption of $\mathbf{0}$ then attacker $\mathcal{B}$ simulates Hyb_1 with attacker $\mathcal{A}$. Furthermore, attacker $\mathcal{B}$ outputs the guess 1 if and only if $\mathcal{A}$ wins the simulated game, therefore the advantage of $\mathcal{B}$ is exactly $\left|\Pr[\mathsf{Hyb}_0(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_1(\mathcal{A}) = 1]\right|$. In addition, if $\mathcal{A}$ is efficient then so is $\mathcal{B}$, therefore by the security of Π_{HE}, we conclude that ε is negligible and the claim follows. □

Lemma 4.16. *If Π_{Com} is binding and Π_{BARG} satisfies set hiding with extraction, set hiding and is somewhere extractable, then $\left|\Pr[\mathsf{Hyb}_1(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_2(\mathcal{A}) = 1]\right| = \mathsf{negl}(\lambda)$.*

Proof. We will leverage Theorem 4.12. To do so, we start by defining a mapping DeriveChal as follows:

$$\mathsf{DeriveChal}(S, i^*) \coloneqq (S, i^*) \mapsto (S, S \setminus \{i^*\}, (i^*, 1)).$$

Secondly, we define the additive invariant P. Recall the tree-indexing definition from Definition 4.1. For any $j \in [2n-1]$, we define the set T_j to be the set of nodes in the sub-tree of node j. We start by defining a predicate $P(\mathbf{ct}_0, \mathbf{ct}_1, \mathsf{sk}_0, \mathsf{sk}_1, j, (\mathbf{v}_1, \ldots, \mathbf{v}_n, \mathsf{idx}))$ as follows:

- On input ciphertexts $\mathbf{ct}_0, \mathbf{ct}_1$, decryption keys $\mathsf{sk}_0, \mathsf{sk}_1$, a sub-tree index j, vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$ of the same length λ, and an index $\mathsf{idx} = (i^*, y)$ where $i^* \in [n]$, compute the difference vector
$$\mathbf{d} = \mathsf{HE.Dec}(\mathsf{sk}_0, \mathbf{ct}_0) - \mathsf{HE.Dec}(\mathsf{sk}_1, \mathbf{ct}_1).$$
- Compute the target vector $\mathbf{t} = \mathbf{v}_{i^*}$ if $i^* \in T_j$ and $\mathbf{t} = \mathbf{0}$ otherwise.
- If $\mathbf{t} = \mathbf{d}$ then output 1. Otherwise, output 0.

In words, the predicate requires the following:

- If i^* is in the sub-tree of node j, then the difference between the encrypted vectors is $\mathbf{v}_{i^*}$.
- If i^* is not in the sub-tree of node j, then the ciphertexts should encrypt identical vectors.

For convenience, we write $P(\mathbf{ct}_0, \mathbf{ct}_1, \mathsf{sk}_0, \mathsf{sk}_1, j, (\mathbf{v}_{i^*}, i^*)) \coloneqq P(\mathbf{ct}_0, \mathbf{ct}_1, \mathsf{sk}_0, \mathsf{sk}_1, j, (\mathbf{v}_1, \ldots, \mathbf{v}_n, \mathsf{idx}))$ since P does not depended on $\mathbf{v}_i$ for all $i \neq i^*$. Let $\mathsf{Expt} \coloneqq \mathsf{Expt}[P, \mathsf{DeriveChal}]$ be the predicate propagation experiment from Definition 4.10. First, we claim that the difference between $\mathcal{A}$ winning Hyb_1 and Hyb_2 is bounded by the probability that $\mathcal{A}$ wins Expt.

Claim 4.17. $\left|\Pr[\mathsf{Hyb}_1(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_2(\mathcal{A}) = 1]\right| \leq \Pr[\mathsf{Expt}(\mathcal{A}) = 1]$.

Proof. Define the event E in Hyb_2 to be:

$$\mathsf{BARG.Verify}(\mathsf{crs}_{\mathsf{BARG}}, C_{i^*,1}, (1, \ldots, 2n-1), \pi) = 1 \text{ and } \mathsf{HE.Dec}(\mathsf{sk}_{\mathsf{main}}, \mathbf{ct}^{\mathsf{main}}_{\mathsf{root}}) = \mathbf{0}.$$

Observe that the view of $\mathcal{A}$ in Hyb_1 is identical to its view in Hyb_2. Furthermore, event E is exactly the condition in which Hyb_1 outputs 1, therefore $\Pr[\mathsf{Hyb}_1(\mathcal{A}) = 1] = \Pr[E]$. Then, we have

$$\begin{aligned}
\left|\Pr[\mathsf{Hyb}_1(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_2(\mathcal{A}) = 1]\right| &= \left|\Pr[E] - \Pr[\mathsf{Hyb}_2(\mathcal{A}) = 1]\right| \\
&= \left|\Pr[E \wedge \mathsf{Hyb}_2(\mathcal{A}) = 0] - \Pr[\neg E \wedge \mathsf{Hyb}_2(\mathcal{A}) = 1]\right| \\
&\leq \max\left\{\Pr[E \wedge \mathsf{Hyb}_2(\mathcal{A}) = 0], \Pr[\neg E \wedge \mathsf{Hyb}_2(\mathcal{A}) = 1]\right\}
\end{aligned}$$

where the second equality follows from the law of total probability and the last inequality follows from the fact that probabilities are non-negative. Now observe that the view of $\mathcal{A}$ in Expt is also identical to its view in Hyb_1 and Hyb_2 by the choice of DeriveChal. Moreover, note that the event E implies $\mathsf{HE.Dec}(\mathsf{sk}_{\mathsf{main}}, \mathbf{ct}^{\mathsf{main}}_{\mathsf{root}}) = \mathbf{0}$ and the event $\mathsf{Hyb}_2(\mathcal{A}) = 1$ implies $\mathsf{HE.Dec}(\mathsf{sk}_{\mathsf{shadow}}, \mathbf{ct}^{\mathsf{shadow}}_{\mathsf{root}}) = \mathbf{v}_{i^*}$. If *exactly one* of those events hold, then

$$\mathsf{HE.Dec}(\mathsf{sk}_{\mathsf{main}}, \mathbf{ct}^{\mathsf{main}}_{\mathsf{root}}) - \mathsf{HE.Dec}(\mathsf{sk}_{\mathsf{shadow}}, \mathbf{ct}^{\mathsf{shadow}}_{\mathsf{root}}) \neq \mathbf{v}_{i^*}.$$

Further, both events E and $\mathsf{Hyb}_2(\mathcal{A}) = 1$ imply $\mathsf{BARG.Verify}(\mathsf{crs}_{\mathsf{BARG}}, C_{i^*,1}, (1, \ldots, 2n-1), \pi) = 1$. In total, both events $E \wedge \mathsf{Hyb}_2(\mathcal{A}) = 0$ and $\neg E \wedge \mathsf{Hyb}_2(\mathcal{A}) = 1$ imply

- $\mathsf{BARG.Verify}(\mathsf{crs}_{\mathsf{BARG}}, C_{i^*,1}, (1, \ldots, 2n-1), \pi) = 1$.
- $\mathsf{HE.Dec}(\mathsf{sk}_{\mathsf{main}}, \mathbf{ct}^{\mathsf{main}}_{\mathsf{root}}) - \mathsf{HE.Dec}(\mathsf{sk}_{\mathsf{shadow}}, \mathbf{ct}^{\mathsf{shadow}}_{\mathsf{root}}) \neq \mathbf{v}_{i^*}$.

In this case, $\mathsf{Expt}(\mathcal{A}) = 1$ by definition of the additive invariant P and the fact that i^* is always in the "sub-tree" of root. Therefore, we conclude that

$$\max\left\{\Pr[E \wedge \mathsf{Hyb}_2(\mathcal{A}) = 0], \Pr[\neg E \wedge \mathsf{Hyb}_2(\mathcal{A}) = 1]\right\} \leq \Pr[\mathsf{Expt}(\mathcal{A}) = 1]$$

and the claim follows. □

Next, we show that $\Pr[\mathsf{Expt}(\mathcal{A}) = 1] = \mathsf{negl}(\lambda)$. The strategy is to use Theorem 4.12. We start by proving that P is a tree-based additive invariant.

Claim 4.18. *If Π_{HE} satisfies evaluation correctness, then the predicate P is a tree-based additive invariant.*

Proof. Let $n \in \mathbb{N}$ be a power of 2 and $\lambda \in \mathbb{N}$. Fix the following quantities:

- any key pairs $(\mathsf{sk}_0, \mathsf{pk}_0), (\mathsf{sk}_1, \mathsf{pk}_1)$ in the support of $\mathsf{HE.Gen}(1^\lambda)$;
- any triple of indices $j, j_{\mathrm{L}}, j_{\mathrm{R}} \in [2n-1]$ where $j_{\mathrm{L}}, j_{\mathrm{R}}$ are the children of j according to Definition 4.1;
- and set of ciphertext vectors $(\mathbf{ct}_{\mathrm{L}}^{(0)}, \mathbf{ct}_{\mathrm{L}}^{(1)}), (\mathbf{ct}_{\mathrm{R}}^{(0)}, \mathbf{ct}_{\mathrm{R}}^{(1)})$ each of length λ;
- any index $i^* \in [n]$;
- any vector $\mathbf{v} := \mathbf{v}_{i^*} \in \{0,1\}^\lambda$.

Let $\mathbf{ct}^{(0)} = \mathsf{HE.Add}(\mathsf{pk}_0, \mathbf{ct}_{\mathrm{L}}^{(0)}, \mathbf{ct}_{\mathrm{R}}^{(0)})$ and $\mathbf{ct}^{(1)} = \mathsf{HE.Add}(\mathsf{pk}_1, \mathbf{ct}_{\mathrm{L}}^{(1)}, \mathbf{ct}_{\mathrm{R}}^{(1)})$, and suppose

$$P(\mathbf{ct}_{\mathrm{L}}^{(0)}, \mathbf{ct}_{\mathrm{L}}^{(1)}, \mathsf{sk}_0, \mathsf{sk}_1, j_{\mathrm{L}}, (\mathbf{v}, i^*)) = 1 \text{ and } P(\mathbf{ct}_{\mathrm{R}}^{(0)}, \mathbf{ct}_{\mathrm{R}}^{(1)}, \mathsf{sk}_0, \mathsf{sk}_1, j_{\mathrm{R}}, (\mathbf{v}, i^*)) = 1.$$

We consider the following cases:

- If $i^* \notin T_j$ then $i^* \notin T_{j_{\mathrm{L}}}$ and $i^* \notin T_{j_{\mathrm{R}}}$. By definition of the predicate P, it holds that

  $$\mathsf{HE.Dec}(\mathsf{sk}_0, \mathbf{ct}_{\mathrm{L}}^{(0)}) - \mathsf{HE.Dec}(\mathsf{sk}_1, \mathbf{ct}_{\mathrm{L}}^{(1)}) = \mathbf{0} \quad \text{and} \quad \mathsf{HE.Dec}(\mathsf{sk}_0, \mathbf{ct}_{\mathrm{R}}^{(0)}) - \mathsf{HE.Dec}(\mathsf{sk}_1, \mathbf{ct}_{\mathrm{R}}^{(1)}) = \mathbf{0}.$$

  By the correctness of Π_{HE}, it holds that $\mathsf{HE.Dec}(\mathsf{sk}_0, \mathbf{ct}^{(0)}) - \mathsf{HE.Dec}(\mathsf{sk}_1, \mathbf{ct}^{(1)}) = \mathbf{0}$ and therefore by definition of P, it holds that $P(\mathbf{ct}^{(0)}, \mathbf{ct}^{(1)}, \mathsf{sk}_0, \mathsf{sk}_1, j, \mathbf{v}, i^*) = 1$.

- Suppose $i^* \in T_j$. Without loss of generality, suppose $i^* \in T_{j_{\mathrm{L}}}$ and $i^* \notin T_{j_{\mathrm{R}}}$; the other case is analogous. Then, by definition of P, it holds that

  $$\mathsf{HE.Dec}(\mathsf{sk}_0, \mathbf{ct}_{\mathrm{L}}^{(0)}) - \mathsf{HE.Dec}(\mathsf{sk}_1, \mathbf{ct}_{\mathrm{L}}^{(1)}) = \mathbf{v} \quad \text{and} \quad \mathsf{HE.Dec}(\mathsf{sk}_0, \mathbf{ct}_{\mathrm{R}}^{(0)}) - \mathsf{HE.Dec}(\mathsf{sk}_1, \mathbf{ct}_{\mathrm{R}}^{(1)}) = \mathbf{0}.$$

  By the correctness of Π_{HE}, it holds that $\mathsf{HE.Dec}(\mathsf{sk}_0, \mathbf{ct}^{(0)}) - \mathsf{HE.Dec}(\mathsf{sk}_1, \mathbf{ct}^{(1)}) = \mathbf{v}$ and therefore by definition of P, it holds that $P(\mathbf{ct}^{(0)}, \mathbf{ct}^{(1)}, \mathsf{sk}_0, \mathsf{sk}_1, j, \mathbf{v}, i^*) = 1$.

In any case, $P(\mathbf{ct}^{(0)}, \mathbf{ct}^{(1)}, \mathsf{sk}_0, \mathsf{sk}_1, j, \mathbf{v}, i^*) = 1$ and therefore P is a tree-based additive invariant by definition. □

For each $j \in [n]$, let $\mathsf{Expt}_j := \mathsf{Expt}_j[P, \mathsf{DeriveChal}]$ be the predicate propagation hybrid experiment from Definition 4.11. The final ingredient needed to invoke Theorem 4.12 is to show that $\mathcal{A}$ wins each of the experiments Expt_j with negligible probability.

Lemma 4.19. *If Π_{Com} is binding against efficient non-uniform adversaries, then for any $j \in [n]$, it holds that*

$$\Pr[\mathsf{Expt}_j(\mathcal{A}) = 1] = \mathsf{negl}(\lambda).$$

Proof. Suppose $\Pr[\mathsf{Expt}_j(\mathcal{A}) = 1] = \varepsilon$. We use $\mathcal{A}$ to construct an efficient adversary $\mathcal{B}$ for the binding security game of Π_{Com} as follows:

1. On input 1^λ, algorithm $\mathcal{B}$ runs $\mathcal{A}$ to obtain the input length 1^n, a set $S \subseteq [n]$ and an index $i^* \in [n]$.
2. Algorithm $\mathcal{B}$ sends 1^{2n-1} to the challenger and gets a CRS $\mathsf{crs}_{\mathsf{Com}}$.
3. Algorithm $\mathcal{B}$ samples the following:
   - $(\mathsf{sk}_{\mathsf{main}}, \mathsf{pk}_{\mathsf{main}}) \leftarrow \mathsf{HE.Gen}(1^\lambda, 1^n)$ and $(\mathsf{sk}_{\mathsf{shadow}}, \mathsf{pk}_{\mathsf{shadow}}) \leftarrow \mathsf{HE.Gen}(1^\lambda, 1^n)$
   - $(\mathsf{crs}_{\mathsf{BARG}}, \mathsf{vk}_{\mathsf{BARG}}, \mathsf{td}_{\mathsf{BARG}}) \leftarrow \mathsf{Gen}(1^\lambda, 1^{2n}, 1^s, 1^3)$.
   - $\mathbf{ct}_{\mathsf{zero}}^{(b)} \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_b, \mathbf{0})$ for all $b \in \{\mathsf{main}, \mathsf{shadow}\}$.
   - For all $i \in [n]$, sample a random $\mathbf{v}_i \xleftarrow{\mathrm{R}} \{0,1\}^\lambda \setminus \{\mathbf{0}\}$.
   - For all $i \in [n], b \in \{\mathsf{main}, \mathsf{shadow}\}$, if $i \in S$ then sample $\mathbf{ct}_i^{(b)} \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_b, \mathbf{v}_i)$, otherwise sample $\mathbf{ct}_i^{(b)} \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_b, \mathbf{0})$.
   - Overwrite $\mathbf{ct}_{i^*}^{\mathsf{shadow}} \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_{\mathsf{shadow}}, \mathbf{0})$.
   - $\left(\mathsf{com}_{\mathsf{hk}}^{(b)}, \sigma_{\mathsf{hk},1}^{(b)}, \ldots, \sigma_{\mathsf{hk},n}^{(b)}\right) \leftarrow \mathsf{Com.Commit}\left(\mathsf{crs}_{\mathsf{Com}}, (\mathbf{ct}_1^{(b)}, \ldots, \mathbf{ct}_n^{(b)})\right)$ for all $b \in \{\mathsf{main}, \mathsf{shadow}\}$.
4. Algorithm $\mathcal{B}$ computes hk and vk as defined in Eqs. (4.1) and (4.2), and passes $(\mathsf{hk}, \mathsf{vk})$ to get (dig, π).
5. Algorithm $\mathcal{B}$ parses $\mathsf{dig} = (\mathbf{ct}_{\mathsf{root}}^{\mathsf{main}}, \mathbf{ct}_{\mathsf{root}}^{\mathsf{shadow}}, \mathsf{com}_{\mathsf{main}}, \mathsf{com}_{\mathsf{shadow}}, \pi_{\mathsf{dig}})$.
6. Algorithm $\mathcal{B}$ extracts $(\hat{\mathbf{ct}}^{\mathsf{main}}, \hat{\mathbf{ct}}^{\mathsf{shadow}}, \sigma^{\mathsf{main}}, \sigma^{\mathsf{shadow}}, \tilde{w}) \leftarrow \mathsf{BARG.Extract}(\mathsf{td}_{\mathsf{BARG}}, \pi, j)$.
7. If any of the following conditions do not hold, algorithm $\mathcal{B}$ aborts:
   (a) $\mathsf{BARG.Verify}(\mathsf{crs}_{\mathsf{BARG}}, C_{i^*,1}, (1, \ldots, 2n-1), \pi) = 1$.
   (b) $C_{i^*,1}(j, (\hat{\mathbf{ct}}^{\mathsf{main}}, \hat{\mathbf{ct}}^{\mathsf{shadow}}, \sigma^{\mathsf{main}}, \sigma^{\mathsf{shadow}}, \tilde{w})) = 1$.
   (c) $P(\hat{\mathbf{ct}}^{\mathsf{main}}, \hat{\mathbf{ct}}^{\mathsf{shadow}}, \mathsf{sk}_{\mathsf{main}}, \mathsf{sk}_{\mathsf{shadow}}, j, \mathbf{v}_{i^*}, i^*) = 0$.
8. Algorithm $\mathcal{B}$ parses $\tilde{w} = (\tilde{\mathbf{ct}}^{\mathsf{main}}, \tilde{\mathbf{ct}}^{\mathsf{shadow}}, \sigma_{\mathsf{hk}}^{\mathsf{main}}, \sigma_{\mathsf{hk}}^{\mathsf{shadow}})$.
9. If $\mathsf{HE.Dec}(\mathsf{sk}_{\mathsf{main}}, \mathbf{ct}_{\mathsf{hk},j}^{\mathsf{main}}) \neq \mathsf{HE.Dec}(\mathsf{sk}_{\mathsf{main}}, \tilde{\mathbf{ct}}^{\mathsf{main}})$ then algorithm $\mathcal{B}$ sets $b \leftarrow \mathsf{main}$, otherwise $b \leftarrow \mathsf{shadow}$.
10. Algorithm $\mathcal{B}$ outputs the commitment $\mathsf{com}_{\mathsf{hk}}^{(b)}$, the index j and the openings $(\mathbf{ct}_j^{(b)}, \sigma_{\mathsf{hk},j}^{(b)})$, $(\tilde{\mathbf{ct}}^{(b)}, \sigma_{\mathsf{hk}}^{(b)})$.

Let $\mathbf{d} = \mathbf{v}_{i^*}$ if $j = i^*$ and $\mathbf{d} = \mathbf{0}$ otherwise. At a high level, algorithm $\mathcal{B}$ simulates Expt_j with algorithm $\mathcal{A}$, hoping that $\mathcal{A}$ wins the experiment. By construction of hk, vk, the difference between the main and shadow vectors encrypted in position j should match $\mathbf{d}$, whereas by definition of the invariant P, the extracted encryptions do not satisfy the condition (since P outputs 0). Therefore, it must be the cast that $\mathcal{A}$ produced a different encryption for position j for either the main or shadow copy. Moreover, algorithm $\mathcal{A}$ produced a valid opening for that value. Together with the valid opening produced by $\mathcal{B}$, this contradicts the binding property of commitment. We now give the formal argument:

By construction, algorithm $\mathcal{B}$ perfectly simulates Expt_j with attacker $\mathcal{A}$ and aborts if and only if $\mathcal{A}$ loses the simulated game. Assume $\mathcal{A}$ wins the simulated game. This implies all of the following:

- Since $C_{i^*,1}(j, (\hat{\mathbf{ct}}^{\mathsf{main}}, \hat{\mathbf{ct}}^{\mathsf{shadow}}, \sigma^{\mathsf{main}}, \sigma^{\mathsf{shadow}}, \tilde{w})) = 1$, then $\mathsf{Com.Verify}(\mathsf{crs}_{\mathsf{Com}}, \mathsf{com}_{\mathsf{hk}}^{(b)}, j, \hat{\mathbf{ct}}^{(b)}, \sigma^{(b)}) = 1$ for each $b \in \{\mathsf{main}, \mathsf{shadow}\}$.
- Since $P(\hat{\mathbf{ct}}^{\mathsf{main}}, \hat{\mathbf{ct}}^{\mathsf{shadow}}, \mathsf{sk}_{\mathsf{main}}, \mathsf{sk}_{\mathsf{shadow}}, j, \mathbf{v}_{i^*}, i^*) = 0$, then by definition

$$\mathsf{HE.Dec}(\mathsf{sk}_{\mathsf{main}}, \hat{\mathbf{ct}}^{\mathsf{main}}) - \mathsf{HE.Dec}(\mathsf{sk}_{\mathsf{shadow}}, \hat{\mathbf{ct}}^{\mathsf{shadow}}) \neq \mathbf{d}.$$

However, by the construction of hk, vk, it holds that:

- $\mathsf{Com.Verify}(\mathsf{crs}_{\mathsf{Com}}, \mathsf{com}_{\mathsf{hk}}^{(b)}, j, \mathbf{ct}_j^{(b)}, \sigma^{(b)}) = 1$ for each $b \in \{\mathsf{main}, \mathsf{shadow}\}$.
- $\mathsf{HE.Dec}(\mathsf{sk}_{\mathsf{main}}, \mathbf{ct}_j^{\mathsf{main}}) - \mathsf{HE.Dec}(\mathsf{sk}_{\mathsf{shadow}}, \mathbf{ct}_j^{\mathsf{shadow}}) = \mathbf{d}$.

Since

$$\mathsf{HE.Dec}(\mathsf{sk}_{\mathsf{main}}, \hat{\mathbf{ct}}^{\mathsf{main}}) - \mathsf{HE.Dec}(\mathsf{sk}_{\mathsf{shadow}}, \hat{\mathbf{ct}}^{\mathsf{shadow}}) \neq \mathsf{HE.Dec}(\mathsf{sk}_{\mathsf{main}}, \mathbf{ct}_j^{\mathsf{main}}) - \mathsf{HE.Dec}(\mathsf{sk}_{\mathsf{shadow}}, \mathbf{ct}_j^{\mathsf{shadow}}),$$

there exists $b \in \{\mathsf{main}, \mathsf{shadow}\}$ such that $\hat{\mathbf{ct}}^{(b)} \neq \mathbf{ct}_j^{(b)}$ by correctness of Π_{HE}. Furthermore, algorithm $\mathcal{B}$ finds that b and outputs $\mathsf{com}_{\mathsf{hk}}^{(b)}$, the index j and the openings $(\mathbf{ct}_j^{(b)}, \sigma_{\mathsf{hk},j}^{(b)})$, $(\tilde{\mathbf{ct}}^{(b)}, \sigma_{\mathsf{hk}}^{(b)})$. Thus, algorithm $\mathcal{B}$ wins the binding game with the same probability ε, so ε is negligible by the binding property of Π_{Com}. The lemma follows. □

By Lemma 4.19 and Claim 4.18, we can apply Theorem 4.12 and conclude that $\Pr[\mathsf{Expt}(\mathcal{A}) = 1] = \mathsf{negl}(\lambda)$. By Claim 4.17, we conclude that $\left|\Pr[\mathsf{Hyb}_1(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_2(\mathcal{A}) = 1]\right| \leq \mathsf{negl}(\lambda)$ and Lemma 4.16 follows. □

Lemma 4.20. *If Π_{HE} is CPA-secure, then $\left|\Pr[\mathsf{Hyb}_2(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_3(\mathcal{A}) = 1]\right| = \mathsf{negl}(\lambda)$.*

Proof. Follow by the analogous argument as in the proof of Lemma 4.15. □

Claim 4.21. $\Pr[\mathsf{Hyb}_3(\mathcal{A}) = 1] = \mathsf{negl}(\lambda)$.

Proof. The view of $\mathcal{A}$ in Hyb_3 is entirely *independent* of $\mathbf{v}_{i^*}$. Thus, in Hyb_3, the challenger can defer the sampling $\mathbf{v}_{i^*}$ to *after* the adversary outputs $\mathsf{ct}_{\mathsf{root}}^{\mathsf{shadow}}$. In order for $\mathcal{A}$ to win the game, it needs to output $\mathbf{ct}_{\mathsf{root}}^{\mathsf{shadow}}$ such that $\mathsf{HE.Dec}(\mathsf{sk}_{\mathsf{shadow}}, \mathbf{ct}_{\mathsf{root}}^{\mathsf{shadow}}) = \mathbf{v}_{i^*}$. Since $\mathbf{v}_{i^*} \xleftarrow{R} \{0,1\}^\lambda \setminus \{\mathbf{0}\}$, this holds with probability $1/(2^\lambda - 1) = \mathsf{negl}(\lambda)$. □

Theorem 4.14 now follows from Lemmas 4.15, 4.16 and 4.20 and Claim 4.21 and a standard hybrid argument. □

4.4 Extractor Validity

In this section, we show that Construction 4.2 satisfies extractor validity. In the extractor validity game, the hash key is sampled to be zero-fixing on the empty set $\emptyset$, and the goal of the adversary is to produce a valid, but *non-matching* digest. In this setting, the ciphertexts in the hash key are all encryptions of $\mathbf{0}$. In order to break the extractor validity property, the adversary needs to produce a root ciphertext that encrypts a non-zero value, and yet, still argue that the root ciphertext was derived by summing a collection of ciphertexts that each encrypt $\mathbf{0}$. The latter is ensured by security of the BARG, and specifically the predicate propagation theorem (Theorem 4.12). We give the formal theorem statement and proof below:

Theorem 4.22. *If Π_{Com} is binding and Π_{BARG} satisfies set hiding, set hiding with extraction and is somewhere extractable, then Π_{H} satisfies the extractor validity.*

Proof. Let $\mathcal{A}$ be an efficient adversary for the extractor validity. For any $\lambda \in \mathbb{N}$, denote $1^n \leftarrow \mathcal{A}_1(1^\lambda)$. We start by defining the mapping $\mathsf{DeriveChal}_\emptyset$ as follows:

$$\mathsf{DeriveChal}(S, i^*) := (S, i^*) \mapsto (\emptyset, \emptyset, \bot).$$

Secondly, we define the predicate $P_{\mathsf{main}}^{\mathsf{Matching}}$ as follows:

$$P_{\mathsf{main}}^{\mathsf{Matching}}(\mathbf{ct}_{\mathsf{main}}, \mathbf{ct}_{\mathsf{shadow}}, \mathsf{sk}_{\mathsf{main}}, \mathsf{sk}_{\mathsf{shadow}}, j, z) = \begin{cases} 1 & \mathsf{HE.Dec}(\mathsf{sk}_{\mathsf{main}}, \mathbf{c}_{\mathsf{main}}) = \mathbf{0} \\ 0 & \mathsf{HE.Dec}(\mathsf{sk}_{\mathsf{main}}, \mathbf{ct}_{\mathsf{main}}) \neq \mathbf{0} \end{cases}$$

Since $P_{\mathsf{main}}^{\mathsf{Matching}}$ does not depend on $\mathbf{ct}_{\mathsf{shadow}}, \mathsf{sk}_{\mathsf{shadow}}, j$ and z, we omit these quantities in the following exposition (i.e., implicitly set them to $\bot$). We start by showing that $P_{\mathsf{main}}^{\mathsf{Matching}}$ is a tree-based additive invariant.

Claim 4.23. *If Π_{HE} satisfies correctness, then the predicate $P^{\mathsf{Matching}}_{\mathsf{main}}$ is a tree-based additive invariant.*

Proof. Let $n \in \mathbb{N}$ be a power of 2 and $\lambda \in \mathbb{N}$. Fix the following quantities:

- a key pair $(\mathsf{sk}, \mathsf{pk})$ in the support of $\mathsf{HE.Gen}(1^\lambda)$;
- a set of ciphertext vectors $\mathbf{ct}_{\mathrm{L}}$ and $\mathbf{ct}_{\mathrm{R}}$ each of length λ;
- $\mathbf{ct} = \mathsf{HE.Add}(\mathsf{pk}, \mathbf{ct}_{\mathrm{L}}, \mathbf{ct}_{\mathrm{R}})$.

Suppose $P^{\mathsf{Matching}}_{\mathsf{main}}(\mathbf{ct}_{\mathrm{L}}, \mathsf{sk}) = 1$ and $P^{\mathsf{Matching}}_{\mathsf{main}}(\mathbf{ct}_{\mathrm{R}}, \mathsf{sk}) = 1$. This implies that $\mathsf{Dec}(\mathsf{sk}, \mathbf{ct}_{\mathrm{L}}) = \mathsf{Dec}(\mathsf{sk}, \mathbf{ct}_{\mathrm{L}}) = \mathbf{0}$ by definition of $P^{\mathsf{Matching}}_{\mathsf{main}}$. By the correctness of Π_{HE}, we have $\mathsf{Dec}(\mathsf{sk}, \mathbf{ct}_{\mathrm{L}}) = \mathbf{0}$, and again by definition of $P^{\mathsf{Matching}}_{\mathsf{main}}$, we get $P^{\mathsf{Matching}}_{\mathsf{main}}(\mathbf{ct}, \mathsf{sk}) = 1$ and the claim follows. □

Let $\mathsf{Expt} := \mathsf{Expt}[P^{\mathsf{Matching}}_{\mathsf{main}}, \mathsf{DeriveChal}_{\varnothing}]$ be the predicate propagation experiment from Definition 4.10. We first claim that we can use $\mathcal{A}$ to construct an adversary $\mathcal{A}'$ such that

$$\Pr[\mathsf{ExptEV}_{\mathcal{A}}(\lambda) = 1] \leq \Pr[\mathsf{Expt}(\mathcal{A}') = 1]. \tag{4.4}$$

Algorithm $\mathcal{A}'$ works as follows:

1. On input the security parameter 1^λ, algorithm $\mathcal{A}'$ runs $\mathcal{A}$ on the same security parameter. Algorithm $\mathcal{A}$ outputs an input length 1^n. Algorithm $\mathcal{A}'$ outputs the input length 1^n, the set $S = \varnothing$, and the index $i^* = \bot$.
2. The challenger replies with $(\mathsf{hk}, \mathsf{vk})$ which $\mathcal{A}'$ forwards to $\mathcal{A}$.
3. Algorithm $\mathcal{A}$ outputs a digest $\mathsf{dig} = \left(\mathbf{ct}^{\mathsf{main}}_{\mathsf{root}}, \mathbf{ct}^{\mathsf{shadow}}_{\mathsf{root}}, \mathsf{com}_{\mathsf{main}}, \mathsf{com}_{\mathsf{shadow}}, \pi_{\mathsf{dig}}\right)$. Algorithm $\mathcal{A}'$ outputs the same digest dig and $\pi = \pi_{\mathsf{dig}}$.

We now show that Eq. (4.4) holds. By construction, the pair $(\mathsf{hk}, \mathsf{vk})$ sampled by the challenger are distributed according to the real setup algorithm. Thus, algorithm $\mathcal{A}$ perfectly simulates an execution of $\mathsf{ExptEV}_{\mathcal{A}}$ for adversary $\mathcal{A}$. Thus, with probability $\Pr[\mathsf{ExptEV}_{\mathcal{A}}(\lambda) = 1]$, algorithm $\mathcal{A}$ outputs a digest dig where $\mathsf{Extract}(\mathsf{td}, \mathsf{dig}) = \mathsf{NotMatching}$ and $\mathsf{ValidateDigest}(\mathsf{hk}, \mathsf{dig}) = 1$. This means the following:

- By construction, $\mathsf{Extract}(\mathsf{td}, \mathsf{dig})$ outputs NotMatching if $\mathsf{HE.Dec}(\mathsf{sk}_{\mathsf{main}}, \mathsf{ct}^{\mathsf{main}}_{\mathsf{root}}) \neq 0$. By construction of $P^{\mathsf{Matching}}_{\mathsf{main}}$, this means $P^{\mathsf{Matching}}_{\mathsf{main}}(\mathbf{ct}^{\mathsf{main}}_{\mathsf{root}}, \mathsf{sk}_{\mathsf{main}}) = 0$.
- Next, ValidateDigest outputs 1 if $\mathsf{BARG.Verify}(\mathsf{vk}_{\mathsf{BARG}}, C_{\bot}, 2n-1, \pi_{\mathsf{dig}}) = 1$. By construction of DeriveChal, we have that $\mathsf{idx} = \bot$ in the execution of $\mathsf{Expt}(\mathcal{A})$, so this means that $\mathsf{BARG.Verify}(\mathsf{vk}_{\mathsf{BARG}}, C_{\mathsf{idx}}, 2n-1, \pi_{\mathsf{dig}}) = 1$.

Since $P^{\mathsf{Matching}}_{\mathsf{main}}(\mathbf{ct}^{\mathsf{main}}_{\mathsf{root}}, \mathsf{sk}_{\mathsf{main}}) = 0$ and $\mathsf{BARG.Verify}(\mathsf{vk}_{\mathsf{BARG}}, C_{\mathsf{idx}}, 2n-1, \pi_{\mathsf{dig}}) = 1$, the predicate propagation experiment $\mathsf{Expt}(\mathcal{A}')$ also outputs 1. Hence, we conclude that $\Pr[\mathsf{Expt}(\mathcal{A}') = 1] \geq \Pr[\mathsf{ExptEV}_{\mathcal{A}}(\lambda) = 1]$. To complete the proof, we now show using Theorem 4.12 that $\Pr[\mathsf{Expt}(\mathcal{A}') = 1] \leq \mathsf{negl}(\lambda)$. To leverage Theorem 4.12, we analyze the predicate propagation hybrid experiment $\mathsf{Expt}_j := \mathsf{Expt}_j[P^{\mathsf{Matching}}_{\mathsf{main}}, \mathsf{DeriveChal}]$ from Definition 4.11.

Claim 4.24. *If Π_{Com} satisfies binding against efficient non-uniform adversaries then for any $j \in [n]$, it holds that*

$$\Pr[\mathsf{Expt}_j(\mathcal{A}') = 1] = \mathsf{negl}(\lambda).$$

Proof. Suppose $\Pr[\mathsf{Expt}_j(\mathcal{A}') = 1] = \varepsilon$. We use $\mathcal{A}'$ to construct an efficient adversary $\mathcal{B}$ for the binding security game of Π_{Com} as follows:

1. On input 1^λ, algorithm $\mathcal{B}$ runs $\mathcal{A}'$ to obtain 1^n, the set $S = \varnothing$ and the index $i^* = \bot$.
2. Algorithm $\mathcal{B}$ outputs the block length $\lambda \cdot \ell_{\mathsf{ct}}(\lambda)$ and the vector length $2n-1$ to the challenger. The challenger responds with $\mathsf{crs}_{\mathsf{Com}}$.

3. Algorithm $\mathcal{B}$ samples the following quantities as Setup:
   - $(\mathsf{sk}_{\mathsf{main}}, \mathsf{pk}_{\mathsf{main}}) \leftarrow \mathsf{HE.Gen}(1^\lambda, 1^n)$ and $(\mathsf{sk}_{\mathsf{shadow}}, \mathsf{pk}_{\mathsf{shadow}}) \leftarrow \mathsf{HE.Gen}(1^\lambda, 1^n)$.
   - $(\mathsf{crs}_{\mathsf{BARG}}, \mathsf{vk}_{\mathsf{BARG}}, \mathsf{td}_{\mathsf{BARG}}) \leftarrow \mathsf{BARG.TrapGen}(1^\lambda, 1^{2n}, 1^s, 1^3, \{j\})$.
   - $\mathbf{ct}_{\mathsf{zero}}^{(b)} \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_b, \mathbf{0})$ for all $b \in \{\mathsf{main}, \mathsf{shadow}\}$.
   - For all $i \in [n], b \in \{\mathsf{main}, \mathsf{shadow}\}$: $\mathbf{ct}_i^{(b)} \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_b, \mathbf{0})$.
   - For each $b \in \{\mathsf{main}, \mathsf{shadow}\}$, let $\left(\mathsf{com}_{\mathsf{hk}}^{(b)}, \sigma_{\mathsf{hk},1}^{(b)}, \ldots, \sigma_{\mathsf{hk},n}^{(b)}\right) \leftarrow \mathsf{Com.Commit}\left(\mathsf{crs}_{\mathsf{Com}}, (\mathbf{ct}_1^{(b)}, \ldots, \mathbf{ct}_n^{(b)})\right)$.
4. Algorithm $\mathcal{B}$ computes hk and vk as defined in Eqs. (4.1) and (4.2), and runs $\mathcal{A}'$ on (hk, vk) to obtain (dig, π).
5. Algorithm $\mathcal{B}$ parses $\mathsf{dig} = (\mathbf{ct}_{\mathsf{root}}^{\mathsf{main}}, \mathbf{ct}_{\mathsf{root}}^{\mathsf{shadow}}, \mathsf{com}_{\mathsf{main}}, \mathsf{com}_{\mathsf{shadow}}, \pi_{\mathsf{dig}})$.
6. Algorithm $\mathcal{B}$ extracts $\hat{w} = (\hat{\mathbf{ct}}^{\mathsf{main}}, \hat{\mathbf{ct}}^{(\mathsf{shadow})}, \sigma^{\mathsf{main}}, \sigma^{\mathsf{shadow}}, \tilde{w}) \leftarrow \mathsf{BARG.Extract}(\mathsf{td}_{\mathsf{BARG}}, \pi, j)$ and parses $\tilde{w} = (\tilde{\mathbf{ct}}^{\mathsf{main}}, \tilde{\mathbf{ct}}^{\mathsf{shadow}}, \sigma_{\mathsf{hk}}^{\mathsf{main}}, \sigma_{\mathsf{hk}}^{\mathsf{shadow}})$.
7. Algorithm $\mathcal{B}$ outputs the commitment $\mathsf{com}_{\mathsf{hk}}^{\mathsf{main}}$, the index j and the openings $(\mathbf{ct}_j^{\mathsf{main}}, \sigma_{\mathsf{hk},j}^{\mathsf{main}})$ and $(\tilde{\mathbf{ct}}^{\mathsf{main}}, \sigma_{\mathsf{hk}}^{\mathsf{main}})$.

By construction, the challenger samples $\mathsf{crs}_{\mathsf{Com}} \leftarrow \mathsf{Com.Setup}(1^\lambda, 1^{\lambda \cdot \ell_{\mathsf{ct}}(\lambda,n)}, 2n-1)$, which matches the specification in Expt_j. Thus, algorithm $\mathcal{B}$ perfectly simulates an execution of Expt_j for $\mathcal{A}'$. By assumption, with probability ε, algorithm $\mathcal{A}'$ outputs dig and π such that the experiment outputs 1. This means the following conditions hold:

$$C_\perp\left(j, \left(\hat{\mathbf{ct}}^{\mathsf{main}}, \hat{\mathbf{ct}}^{(\mathsf{shadow})}, \sigma^{\mathsf{main}}, \sigma^{\mathsf{shadow}}, \tilde{w}\right)\right) = 1 \quad \text{and} \quad P_{\mathsf{main}}^{\mathsf{Matching}}(\hat{\mathbf{ct}}^{\mathsf{main}}, \mathsf{sk}_{\mathsf{main}}) = 0.$$

By definition of $C_\perp$ and using the fact that $j \in [n]$, this means

$$\mathsf{Com.Verify}\left(\mathsf{crs}_{\mathsf{com}}, \mathsf{com}_{\mathsf{hk}}^{\mathsf{main}}, j, \tilde{\mathbf{ct}}^{\mathsf{main}}, \sigma_{\mathsf{hk}}^{\mathsf{main}}\right) = 1 \quad \text{and} \quad \hat{\mathbf{ct}}^{\mathsf{main}} \in \left\{\mathbf{ct}_{\mathsf{zero}}^{\mathsf{main}}, \tilde{\mathbf{ct}}^{\mathsf{main}}\right\}.$$

Next, by correctness of Π_{Com},

$$\mathsf{Com.Verify}\left(\mathsf{crs}_{\mathsf{com}}, \mathsf{com}_{\mathsf{hk}}^{(0)}, j, \mathbf{ct}_j^{\mathsf{main}}, \sigma_{\mathsf{hk},j}^{\mathsf{main}}\right) = 1.$$

Therefore, it suffices to argue that $\mathbf{ct}_j^{\mathsf{main}} \neq \tilde{\mathbf{ct}}^{\mathsf{main}}$. Since $P_{\mathsf{main}}^{\mathsf{Matching}}(\hat{\mathbf{ct}}^{\mathsf{main}}, \mathsf{sk}_{\mathsf{main}}) = 0$, this means

$$\mathsf{HE.Dec}(\mathsf{sk}_{\mathsf{main}}, \hat{\mathbf{ct}}^{\mathsf{main}}) \neq 0.$$

Since $\mathsf{ct}_{\mathsf{zero}}^{\mathsf{main}}$ is an encryption of 0, we can appeal to perfect correctness of Π_{HE} to conclude that $\hat{\mathbf{ct}}^{\mathsf{main}} \neq \mathsf{ct}_{\mathsf{zero}}^{\mathsf{main}}$. Therefore it must be that $\hat{\mathbf{ct}}^{\mathsf{main}} = \tilde{\mathbf{ct}}^{\mathsf{main}}$. Moreover, $\mathbf{ct}_j^{\mathsf{main}}$ is also an encryption of 0, so again by perfect correctness of the encryption scheme, we can conclude that $\mathbf{ct}_j^{\mathsf{main}} \neq \hat{\mathbf{ct}}^{\mathsf{main}} = \tilde{\mathbf{ct}}^{\mathsf{main}}$. In this case, algorithm $\mathcal{B}$ successfully opens $\mathsf{com}_{\mathsf{hk}}^{\mathsf{main}}$ to two distinct values $\mathbf{ct}_j^{\mathsf{main}} \neq \tilde{\mathbf{ct}}^{\mathsf{main}}$. Thus algorithm $\mathcal{B}$ breaks binding with the same advantage ε. □

By Claims 4.23 and 4.24, we can invoke Theorem 4.12 to conclude that $\Pr[\mathsf{Expt}(\mathcal{A}') = 1] \leq \mathsf{negl}(\lambda)$. Extractor-validity security now follows via Eq. (4.4). □

4.4.1 Index Hiding with Extracted Guess

In this section, we show that Construction 4.2 satisfies the index hiding with extracted guess property. The challenge in this reduction is we need to switch from an encryption of 0 to an encryption of 1 (in the hash key) while retaining the ability to decide whether the digest is "Matching" or not (which in the real scheme, requires knowledge of the secret key for the underlying encryption scheme). Similar to the proof of Theorem 4.14 and as described in Section 2, we leverage a Naor-Yung proof strategy for the analysis here.

Theorem 4.25. *If Π_{HE} satisfies perfect correctness, evaluation correctness, and CPA-security, Π_{Com} is computationally binding and Π_{BARG} satisfies set hiding with extraction, set hiding, and is somewhere extractable, then Construction 4.2 satisfies index hiding with extracted guess.*

Proof. Let $\mathcal{A}$ be an efficient adversary for the index hiding with extracted guess security game. We define a sequence of hybrid experiments:

- Hyb_0: This is $\mathsf{ExptIHE}_{\mathcal{A}}(\lambda, 0)$. Specifically, the game proceeds as follows:
  1. On input the security parameter 1^λ, algorithm $\mathcal{A}$ outputs the input length 1^n, a set $S \subseteq [n]$, and an index $i^* \in S$.
  2. The challenger now samples the following quantities as in Setup:
     - Sample $(\mathsf{sk}_{\mathsf{main}}, \mathsf{pk}_{\mathsf{main}}) \leftarrow \mathsf{HE.Gen}(1^\lambda, 1^n)$ and $(\mathsf{sk}_{\mathsf{shadow}}, \mathsf{pk}_{\mathsf{shadow}}) \leftarrow \mathsf{HE.Gen}(1^\lambda, 1^n)$.
     - Sample $\mathsf{crs}_{\mathsf{Com}} \leftarrow \mathsf{Com.Setup}(1^\lambda, 1^{\lambda \cdot \ell_{\mathsf{ct}}(\lambda, n)}, 2n-1)$.
     - Sample $(\mathsf{crs}_{\mathsf{BARG}}, \mathsf{vk}_{\mathsf{BARG}}) \leftarrow \mathsf{BARG.Gen}(1^\lambda, 1^{2n-1}, 1^s, 1^3)$, where s is a bound on the size of the circuit computing the index relation from Fig. 1.
     - For each $b \in \{\mathsf{main}, \mathsf{shadow}\}$, sample $\mathsf{ct}_{\mathsf{zero}}^{(b)} \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_b, \mathbf{0})$.
     - For all $i \in [n]$, sample a random $\mathbf{v}_i \xleftarrow{\mathrm{R}} \{0,1\}^\lambda \setminus \{\mathbf{0}\}$.
     - For each $i \in [n]$ and $b \in \{\mathsf{main}, \mathsf{shadow}\}$, if $i \in S \setminus \{i^*\}$, sample $\mathbf{ct}_i^{(b)} \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_b, \mathbf{v}_i)$; otherwise sample $\mathbf{ct}_i^{(b)} \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_b, \mathbf{0})$.
     - For each $b \in \{\mathsf{main}, \mathsf{shadow}\}$, let $\left(\mathsf{com}_{\mathsf{hk}}^{(b)}, \sigma_{\mathsf{hk},1}^{(b)}, \ldots, \sigma_{\mathsf{hk},n}^{(b)}\right) \leftarrow \mathsf{Com.Commit}\left(\mathsf{crs}_{\mathsf{Com}}, (\mathbf{ct}_1^{(b)}, \ldots, \mathbf{ct}_n^{(b)})\right)$.
  3. The challenger constructs hk and vk according to Eqs. (4.1) and (4.2):
  $$\mathsf{hk} = \left(\mathsf{crs}_{\mathsf{Com}}, \mathsf{crs}_{\mathsf{BARG}}, \left\{\mathsf{pk}_b, \mathbf{ct}_{\mathsf{zero}}^{(b)}, \mathbf{ct}_1^{(b)}, \ldots, \mathbf{ct}_n^{(b)}, \sigma_{\mathsf{hk},1}^{(b)}, \ldots, \sigma_{\mathsf{hk},n}^{(b)}\right\}_{b \in \{\mathsf{main},\mathsf{shadow}\}}\right)$$
  $$\mathsf{vk} = \left(\mathsf{crs}_{\mathsf{Com}}, \mathsf{vk}_{\mathsf{BARG}}, \left\{\mathsf{pk}_b, \mathbf{ct}_{\mathsf{zero}}^{(b)}, \mathsf{com}_{\mathsf{hk}}^{(b)}\right\}_{b \in \{\mathsf{main},\mathsf{shadow}\}}\right)$$
  The challenger gives $(\mathsf{hk}, \mathsf{vk})$ to $\mathcal{A}$.
  4. Algorithm $\mathcal{A}$ outputs a digest $\mathsf{dig} = \left(\mathbf{ct}_{\mathsf{root}}^{\mathsf{main}}, \mathbf{ct}_{\mathsf{root}}^{\mathsf{shadow}}, \mathsf{com}_{\mathsf{main}}, \mathsf{com}_{\mathsf{shadow}}, \pi_{\mathsf{dig}}\right)$ and an opening π.
  5. The output of the experiment is 1 if
  $$\mathsf{BARG.Verify}(\mathsf{vk}_{\mathsf{BARG}}, C_{i^*,0}, 2n-1, \pi) = 1 \quad \text{and} \quad \mathsf{HE.Dec}\left(\mathsf{sk}_{\mathsf{main}}, \mathbf{ct}_{\mathsf{root}}^{\mathsf{main}}\right) = \mathbf{0}.$$
  Otherwise, the output is 0.
- Hyb_1: Same as Hyb_0, except the challenger samples $\mathbf{ct}_{i^*}^{\mathsf{shadow}} \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_{\mathsf{shadow}}, \mathbf{v}_{i^*})$.
- Hyb_2: Same as Hyb_1, except the output of the experiment is 1 if
  $$\mathsf{BARG.Verify}(\mathsf{vk}_{\mathsf{BARG}}, C_{i^*,0}, 2n-1, \pi) = 1 \quad \text{and} \quad \mathsf{HE.Dec}\left(\mathsf{sk}_{\mathsf{shadow}}, \mathbf{ct}_{\mathsf{root}}^{\mathsf{shadow}}\right) = \mathbf{0}.$$
  Notably, the challenger's behavior in this experiment does *not* depend on $\mathsf{sk}_{\mathsf{main}}$.
- Hyb_3: Same as Hyb_2, except the challenger samples $\mathbf{ct}_{i^*}^{\mathsf{main}} \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_{\mathsf{main}}, \mathbf{v}_{i^*})$.
- Hyb_4: Same as Hyb_3, except the output of the experiment is 1 if
  $$\mathsf{BARG.Verify}(\mathsf{vk}_{\mathsf{BARG}}, C_{i^*,0}, 2n-1, \pi) = 1 \quad \text{and} \quad \mathsf{HE.Dec}\left(\mathsf{sk}_{\mathsf{main}}, \mathbf{ct}_{\mathsf{root}}^{\mathsf{main}}\right) = \mathbf{0}.$$
  This is experiment $\mathsf{ExptIHE}_{\mathcal{A}}(\lambda, 1)$.

We write $\mathsf{Hyb}_i(\mathcal{A})$ to denote the output of experiment of Hyb_i with adversary $\mathcal{A}$. We now analyze each pair of hybrid experiments.

Claim 4.26. *If Π_{HE} is CPA-secure, then there exists a negligible function $\mathsf{negl}(\cdot)$ such that*

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

Proof. Suppose $\left|\Pr[\mathsf{Hyb}_0(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_1(\mathcal{A}) = 1]\right| = \varepsilon$. We use $\mathcal{A}$ to construct an efficient attacker $\mathcal{B}$ for the CPA security game as follows:

1. On input 1^λ, algorithm $\mathcal{B}$ runs $\mathcal{A}$ to obtain the input length 1^n, a set $S \subseteq [n]$, and an index $i^* \in S$.
2. The challenger sends the public key $\mathsf{pk}_{\mathsf{shadow}}$ to $\mathcal{B}$.
3. Algorithm $\mathcal{B}$ samples a random $\mathbf{v}_i \xleftarrow{\mathrm{R}} \{0,1\}^\lambda \setminus \{\mathbf{0}\}$ for each $i \in [n]$.
4. Algorithm $\mathcal{B}$ sends the challenge $(\mathbf{0}, \mathbf{v}_{i^*})$ to the challenger and gets an encryption $\mathbf{ct}^*$.
5. Algorithm $\mathcal{B}$ samples the following:
   - $(\mathsf{sk}_{\mathsf{main}}, \mathsf{pk}_{\mathsf{main}}) \leftarrow \mathsf{HE.Gen}(1^\lambda)$.
   - $(\mathsf{crs}_{\mathsf{BARG}}, \mathsf{vk}_{\mathsf{BARG}}) \leftarrow \mathsf{Gen}(1^\lambda, 1^{2n}, 1^s, 1^3)$.
   - $\mathsf{crs}_{\mathsf{Com}} \leftarrow \mathsf{Com.Setup}(1^\lambda, 1^{\lambda \cdot \ell_{\mathsf{ct}}(\lambda)}, 2n-1)$.
   - $\mathbf{ct}_{\mathsf{zero}}^{(b)} \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_b, \mathbf{0})$ for all $b \in \{\mathsf{main}, \mathsf{shadow}\}$.
   - For all $i \in [n], b \in \{\mathsf{main}, \mathsf{shadow}\}$, if $i \in S \setminus \{i^*\}$ then sample $\mathbf{ct}_i^{(b)} \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_b, \mathbf{v}_i)$. Otherwise sample $\mathbf{ct}_i^{(b)} \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_b, \mathbf{0})$.
   - Let $\mathsf{ct}_{i^*}^{\mathsf{shadow}} \leftarrow \mathsf{ct}^*$.
   - $\left(\mathsf{com}_{\mathsf{hk}}^{(b)}, \sigma_{\mathsf{hk},1}^{(b)}, \ldots, \sigma_{\mathsf{hk},n}^{(b)}\right) \leftarrow \mathsf{Com.Commit}\left(\mathsf{crs}_{\mathsf{Com}}, (\mathbf{ct}_1^{(b)}, \ldots, \mathbf{ct}_n^{(b)})\right)$ for all $b \in \{\mathsf{main}, \mathsf{shadow}\}$.
6. The challenger constructs hk and vk according to Eqs. (4.1) and (4.2):

$$\mathsf{hk} = \left(\mathsf{crs}_{\mathsf{Com}}, \mathsf{crs}_{\mathsf{BARG}}, \left\{\mathsf{pk}_b, \mathbf{ct}_{\mathsf{zero}}^{(b)}, \mathbf{ct}_1^{(b)}, \ldots, \mathbf{ct}_n^{(b)}, \sigma_{\mathsf{hk},1}^{(b)}, \ldots, \sigma_{\mathsf{hk},n}^{(b)}\right\}_{b \in \{\mathsf{main},\mathsf{shadow}\}}\right)$$
$$\mathsf{vk} = \left(\mathsf{crs}_{\mathsf{Com}}, \mathsf{vk}_{\mathsf{BARG}}, \left\{\mathsf{pk}_b, \mathbf{ct}_{\mathsf{zero}}^{(b)}, \mathsf{com}_{\mathsf{hk}}^{(b)}\right\}_{b \in \{\mathsf{main},\mathsf{shadow}\}}\right)$$

   The challenger gives $(\mathsf{hk}, \mathsf{vk})$ to $\mathcal{A}$.
7. Algorithm $\mathcal{A}$ outputs a digest $\mathsf{dig} = \left(\mathbf{ct}_{\mathsf{root}}^{\mathsf{main}}, \mathbf{ct}_{\mathsf{root}}^{\mathsf{shadow}}, \mathsf{com}_{\mathsf{main}}, \mathsf{com}_{\mathsf{shadow}}, \pi_{\mathsf{dig}}\right)$ and an opening π.
8. Algorithm $\mathcal{B}$ parses outputs 1 if all of the following conditions hold, and 0 otherwise:
   (a) $\mathsf{BARG.Verify}(\mathsf{crs}_{\mathsf{BARG}}, C_{i^*,1}, (1, \ldots, 2n-1), \pi) = 1$.
   (b) $\mathsf{HE.Dec}(\mathsf{sk}_{\mathsf{main}}, \mathbf{ct}_{\mathsf{root}}^{\mathsf{main}}) = \mathbf{0}$.

By construction, if ct^* is an encryption of $\mathbf{0}$ then algorithm $\mathcal{B}$ simulates Hyb_0 with attacker $\mathcal{A}$ and if ct^* is an encryption of $\mathbf{v}_{i^*}$ then attacker $\mathcal{B}$ simulates Hyb_1 with attacker $\mathcal{A}$. Furthermore, attacker $\mathcal{B}$ outputs the guess 1 if and only if $\mathcal{A}$ wins the simulated game, therefore the advantage of $\mathcal{B}$ is exactly $\left|\Pr[\mathsf{Hyb}_0(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_1(\mathcal{A}) = 1]\right|$. In addition, if $\mathcal{A}$ is efficient then so is $\mathcal{B}$, therefore by the security of Π_{HE}, we conclude that ε is negligible and the claim follows. □

Claim 4.27. *If Π_{HE} is perfectly correct and satisfies evaluation correctness, Π_{Com} is computationally binding, Π_{BARG} satisfies set hiding with extraction, set hiding, and is somewhere extractable, then there exists a negligible function $\mathsf{negl}(\cdot)$ such that $|\Pr[\mathsf{Hyb}_2(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_1(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda)$.*

Proof. By construction, the only difference between the execution of Hyb_1 and Hyb_2 is the output condition. Let E be the following event in an execution of Hyb_1 and Hyb_2:

$$\mathsf{BARG.Verify}(\mathsf{vk}_{\mathsf{BARG}}, C_{i^*,0}, 2n-1, \pi) = 1 \quad \text{and} \quad \mathsf{HE.Dec}\big(\mathsf{sk}_{\mathsf{main}}, \mathbf{ct}_{\mathsf{root}}^{\mathsf{main}}\big) \neq \mathsf{HE.Dec}\big(\mathsf{sk}_{\mathsf{shadow}}, \mathbf{ct}_{\mathsf{root}}^{\mathsf{shadow}}\big). \tag{4.5}$$

Observe that if E does not occur, then the output of Hyb_1 and Hyb_2 is identical. This means that

$$|\Pr[\mathsf{Hyb}_2(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_1(\mathcal{A}) = 1]| \leq \Pr[E].$$

We now leverage Theorem 4.12 to argue that $\Pr[E] = \mathsf{negl}(\lambda)$. To do so, we start by defining the mapping DeriveChal as follows:

$$\mathsf{DeriveChal}(S, i^*) := (S, i^*) \to (S, S \setminus \{i^*\}, (i^*, 0)).$$

Next, we define the validity predicate $P_{\mathsf{Valid}} \colon \{0,1\}^* \to \{0,1\}$ as follows:

$$P_{\mathsf{Valid}}(\mathsf{ct}_{\mathsf{main}}, \mathsf{ct}_{\mathsf{shadow}}, \mathsf{sk}_{\mathsf{main}}, \mathsf{sk}_{\mathsf{shadow}}, j, z) = \begin{cases} 1 & \mathsf{HE.Dec}(\mathsf{sk}_{\mathsf{main}}, \mathsf{ct}_{\mathsf{main}}) = \mathsf{HE.Dec}(\mathsf{sk}_{\mathsf{shadow}}, \mathsf{ct}_{\mathsf{shadow}}) \\ 0 & \mathsf{HE.Dec}(\mathsf{sk}_{\mathsf{main}}, \mathsf{ct}_{\mathsf{main}}) \neq \mathsf{HE.Dec}(\mathsf{sk}_{\mathsf{shadow}}, \mathsf{ct}_{\mathsf{shadow}}) \end{cases}$$

Since P_{Valid} does not use the index j and the auxiliary input z, we omit them in the following exposition. We now show that P_{Valid} is a tree-based additive invariant.

Lemma 4.28. *If Π_{HE} satisfies evaluation correctness, then P_{Valid} is a tree-based additive invariant.*

Proof. Let $\lambda \in \mathbb{N}$. Fix the following quantities:

- any two key pairs $(\mathsf{sk}_{\mathsf{main}}, \mathsf{pk}_{\mathsf{main}})$, $(\mathsf{sk}_{\mathsf{shadow}}, \mathsf{pk}_{\mathsf{shadow}})$ in the support of $\mathsf{HE.Gen}(1^\lambda, 1^n)$;
- any tuple of ciphertext vectors $\big(\mathbf{ct}_{\mathsf{L}}^{\mathsf{main}}, \mathbf{ct}_{\mathsf{L}}^{\mathsf{shadow}}\big)$, $\big(\mathbf{ct}_{\mathsf{R}}^{\mathsf{main}}, \mathbf{ct}_{\mathsf{R}}^{\mathsf{shadow}}\big)$, where each vector has length λ;
- for each $b \in \{\mathsf{main}, \mathsf{shadow}\}$, let $\mathbf{ct}_{\mathsf{sum}}^{(b)} = \mathsf{HE.Add}\big(\mathsf{pk}_b, \mathbf{ct}_{\mathsf{L}}^{(b)}, \mathbf{ct}_{\mathsf{R}}^{(b)}\big)$.

Suppose

$$P_{\mathsf{Valid}}(\mathbf{ct}_{\mathsf{L}}^{\mathsf{main}}, \mathbf{ct}_{\mathsf{L}}^{\mathsf{shadow}}, \mathsf{sk}_{\mathsf{main}}, \mathsf{sk}_{\mathsf{shadow}}) = P_{\mathsf{Valid}}(\mathbf{ct}_{\mathsf{R}}^{\mathsf{main}}, \mathbf{ct}_{\mathsf{R}}^{\mathsf{shadow}}, \mathsf{sk}_{\mathsf{main}}, \mathsf{sk}_{\mathsf{shadow}}) = 1.$$

This implies

$$\begin{aligned} \mathsf{HE.Dec}(\mathsf{sk}_{\mathsf{main}}, \mathbf{ct}_{\mathsf{L}}^{\mathsf{main}}) &= \mathsf{HE.Dec}(\mathsf{sk}_{\mathsf{shadow}}, \mathbf{ct}_{\mathsf{L}}^{\mathsf{shadow}}) \\ \mathsf{HE.Dec}(\mathsf{sk}_{\mathsf{main}}, \mathbf{ct}_{\mathsf{R}}^{\mathsf{main}}) &= \mathsf{HE.Dec}(\mathsf{sk}_{\mathsf{shadow}}, \mathbf{ct}_{\mathsf{R}}^{\mathsf{shadow}}). \end{aligned}$$

By the evaluation correctness of Π_{HE}, we conclude that

$$\begin{aligned} \mathsf{HE.Dec}(\mathsf{sk}_{\mathsf{main}}, \mathbf{ct}_{\mathsf{sum}}^{\mathsf{main}}) &= \mathsf{HE.Dec}(\mathsf{sk}_{\mathsf{main}}, \mathbf{ct}_{\mathsf{L}}^{\mathsf{main}}) + \mathsf{HE.Dec}(\mathsf{sk}_{\mathsf{main}}, \mathbf{ct}_{\mathsf{R}}^{\mathsf{main}}) \\ &= \mathsf{HE.Dec}(\mathsf{sk}_{\mathsf{shadow}}, \mathbf{ct}_{\mathsf{L}}^{\mathsf{shadow}}) + \mathsf{HE.Dec}(\mathsf{sk}_{\mathsf{shadow}}, \mathbf{ct}_{\mathsf{R}}^{\mathsf{shadow}}) \\ &= \mathsf{HE.Dec}(\mathsf{sk}_{\mathsf{shadow}}, \mathbf{ct}_{\mathsf{sum}}^{\mathsf{shadow}}) \end{aligned}$$

Therefore we conclude that $P_{\mathsf{Valid}}(\mathbf{ct}_{\mathsf{sum}}^{\mathsf{main}}, \mathbf{ct}_{\mathsf{sum}}^{\mathsf{shadow}}, \mathsf{sk}_{\mathsf{main}}, \mathsf{sk}_{\mathsf{shadow}}) = 1$ and the claim follows. □

Let $\mathsf{Expt} := \mathsf{Expt}[P_{\mathsf{Valid}}, \mathsf{DeriveChal}]$ be the predicate propagation experiment from Definition 4.10. We argue that

$$\Pr[E] \leq \Pr[\mathsf{Expt}(\mathcal{A}) = 1], \tag{4.6}$$

where E is the event from Eq. (4.5). By construction, the adversary's view in Hyb_1 and Expt is identical. Suppose E occurs in an execution of Hyb_1. Then the following hold:

- $\mathsf{BARG.Verify}(\mathsf{vk}_{\mathsf{BARG}}, C_{i^*,0}, 2n-1, \pi) = 1$. By construction of DeriveChal, we have that $\mathsf{idx} = (i^*, 0)$ in the execution of $\mathsf{Expt}(\mathcal{A})$. Hence, this means that $\mathsf{BARG.Verify}(\mathsf{vk}_{\mathsf{BARG}}, C_{\mathsf{idx}}, 2n-1, \pi) = 1$.

- $\mathsf{HE.Dec}\big(\mathsf{sk}_{\mathsf{main}}, \mathbf{ct}^{\mathsf{main}}_{\mathsf{root}}\big) \neq \mathsf{HE.Dec}\big(\mathsf{sk}_{\mathsf{shadow}}, \mathbf{ct}^{\mathsf{shadow}}_{\mathsf{root}}\big)$. This means $P_{\mathsf{Valid}}\big(\mathbf{ct}^{\mathsf{main}}_{\mathsf{root}}, \mathbf{ct}^{\mathsf{shadow}}_{\mathsf{root}}, \mathsf{sk}_{\mathsf{main}}, \mathsf{sk}_{\mathsf{shadow}}\big) = 0$.

Correspondingly, the output in Expt is also 1 in this case. Hence, we conclude that $\Pr[\mathsf{Expt}(\mathcal{A}) = 1] \geq \Pr[E]$. To complete the proof, we analyze the predicate propagation hybrid experiment $\mathsf{Expt}_j := \mathsf{Expt}_j[P_{\mathsf{Valid}}, \mathsf{DeriveChal}]$.

Lemma 4.29. *If Π_{HE} is perfectly correct and Π_{Com} satisfies computational binding, then there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $j \in [n]$, it holds that $\Pr[\mathsf{Expt}_j(\mathcal{A}) = 1] = \mathsf{negl}(\lambda)$.*

Proof. Suppose there exists some $j \in [n]$ where $\Pr[\mathsf{Expt}_j(\mathcal{A}) = 1] \geq \varepsilon(\lambda)$ for some non-negligible ε. We use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ that breaks computational binding of Π_{Com}.

1. On input the security parameter 1^λ, algorithm $\mathcal{B}$ runs algorithm $\mathcal{A}$ to obtain the input length 1^n, a set $S \subseteq [n]$, and an index $i^* \in S$.
2. Algorithm $\mathcal{B}$ outputs the block length $1^{\lambda \cdot \ell_{\mathsf{ct}}(\lambda, n)}$ and the vector length $2n - 1$ to the challenger. The challenger responds with $\mathsf{crs}_{\mathsf{Com}}$.
3. Algorithm $\mathcal{B}$ computes $(S \setminus \{i^*\}, S, (i^*, 0)) \leftarrow \mathsf{DeriveChal}(S, i^*)$. It then samples the following components:
   - $(\mathsf{sk}_{\mathsf{main}}, \mathsf{pk}_{\mathsf{main}}) \leftarrow \mathsf{HE.Gen}(1^\lambda, 1^n)$, $(\mathsf{sk}_{\mathsf{shadow}}, \mathsf{pk}_{\mathsf{shadow}}) \leftarrow \mathsf{HE.Gen}(1^\lambda, 1^n)$.
   - $(\mathsf{crs}_{\mathsf{BARG}}, \mathsf{vk}_{\mathsf{BARG}}, \mathsf{td}_{\mathsf{BARG}}) \leftarrow \mathsf{BARG.TrapGen}(1^\lambda, 1^{2n-1}, 1^s, 1^3, \{j\})$.
   - Sample a random $\mathbf{v}_i \xleftarrow{R} \{0,1\}^\lambda \setminus \{\mathbf{0}\}$ for each $i \in [n]$.
   - For each $b \in \{\mathsf{main}, \mathsf{shadow}\}$, sample $\mathbf{ct}^{(b)}_{\mathsf{zero}} \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_b, \mathbf{0})$.
   - For each $i \in [n] \setminus \{i^*\}$ and $b \in \{\mathsf{main}, \mathsf{shadow}\}$, if $i \in S$, sample $\mathbf{ct}^{(b)}_i \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_b, \mathbf{v}_i)$. If $i \notin S$, sample $\mathbf{ct}^{(b)}_i \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_b, \mathbf{0})$.
   - Sample $\mathbf{ct}^{\mathsf{main}}_{i^*} \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_{\mathsf{main}}, \mathbf{0})$ and $\mathbf{ct}^{\mathsf{shadow}}_{i^*} \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_{\mathsf{shadow}}, \mathbf{v}_{i^*})$.
   - For each $b \in \{\mathsf{main}, \mathsf{shadow}\}$, let $\big(\mathsf{com}^{(b)}_{\mathsf{hk}}, \sigma^{(b)}_{\mathsf{hk},1}, \ldots, \sigma^{(b)}_{\mathsf{hk},n}\big) \leftarrow \mathsf{Com.Commit}\big(\mathsf{crs}_{\mathsf{Com}}, (\mathbf{ct}^{(b)}_1, \ldots, \mathbf{ct}^{(b)}_n)\big)$.
4. Algorithm $\mathcal{B}$ constructs hk and vk according to Eqs. (4.1) and (4.2):

$$\mathsf{hk} = \Big(\mathsf{crs}_{\mathsf{Com}}, \mathsf{crs}_{\mathsf{BARG}}, \big\{\mathsf{pk}_b, \mathbf{ct}^{(b)}_{\mathsf{zero}}, \mathbf{ct}^{(b)}_1, \ldots, \mathbf{ct}^{(b)}_n, \sigma^{(b)}_{\mathsf{hk},1}, \ldots, \sigma^{(b)}_{\mathsf{hk},n}\big\}_{b \in \{\mathsf{main},\mathsf{shadow}\}}\Big)$$
$$\mathsf{vk} = \Big(\mathsf{crs}_{\mathsf{Com}}, \mathsf{vk}_{\mathsf{BARG}}, \big\{\mathsf{pk}_b, \mathbf{ct}^{(b)}_{\mathsf{zero}}, \mathsf{com}^{(b)}_{\mathsf{hk}}\big\}_{b \in \{\mathsf{main},\mathsf{shadow}\}}\Big)$$

   Algorithm $\mathcal{B}$ gives $(\mathsf{hk}, \mathsf{vk})$ to $\mathcal{A}$.
5. Algorithm $\mathcal{A}$ outputs a digest $\mathsf{dig} = \big(\mathbf{ct}^{\mathsf{main}}_{\mathsf{root}}, \mathbf{ct}^{\mathsf{shadow}}_{\mathsf{root}}, \mathsf{com}_{\mathsf{main}}, \mathsf{com}_{\mathsf{shadow}}, \pi_{\mathsf{dig}}\big)$ and a proof π.
6. Algorithm $\mathcal{B}$ extracts $\hat{w} = (\hat{\mathbf{ct}}^{\mathsf{main}}, \hat{\mathbf{ct}}^{(\mathsf{shadow})}, \sigma^{\mathsf{main}}, \sigma^{\mathsf{shadow}}, \tilde{w}) \leftarrow \mathsf{BARG.Extract}(\mathsf{td}_{\mathsf{BARG}}, \pi, j)$ and parses $\tilde{w} = (\tilde{\mathbf{ct}}^{\mathsf{main}}, \tilde{\mathbf{ct}}^{\mathsf{shadow}}, \sigma^{\mathsf{main}}_{\mathsf{hk}}, \sigma^{\mathsf{shadow}}_{\mathsf{hk}})$.
7. Algorithm $\mathcal{B}$ checks if there exists $b \in \{\mathsf{main}, \mathsf{shadow}\}$ where $\mathsf{Com.Verify}\big(\mathsf{crs}_{\mathsf{Com}}, \mathsf{com}^{(b)}_{\mathsf{hk}}, j, \tilde{\mathbf{ct}}^{(b)}, \sigma^{(b)}_{\mathsf{hk}}\big) = 1$ and $\tilde{\mathbf{ct}}^{(b)} \neq \mathbf{ct}^{(b)}_j$. If so, it outputs the commitment $\mathsf{com}^{(b)}_{\mathsf{hk}}$, the index j, and the value-opening pairs $\big(\mathbf{ct}^{(b)}_j, \sigma^{(b)}_{\mathsf{hk},j}\big)$ and $\big(\tilde{\mathbf{ct}}^{(b)}, \sigma^{(b)}_{\mathsf{hk}}\big)$.

By construction, the challenger samples $\mathsf{crs}_{\mathsf{Com}} \leftarrow \mathsf{Com.Setup}(1^\lambda, 1^{\ell_{\mathsf{ct}}(\lambda,n)}, 2n-1)$, which matches the specification in Expt_j. This, algorithm $\mathcal{B}$ perfectly simulates an execution of Expt_j for $\mathcal{A}$. By assumption, with probability ε, algorithm $\mathcal{A}$ outputs dig and π such that the experiment outputs 1. This means the following conditions hold:

$$C_{i^*,0}\big(j, (\hat{\mathbf{ct}}^{\mathsf{main}}, \hat{\mathbf{ct}}^{(\mathsf{shadow})}, \sigma^{\mathsf{main}}, \sigma^{\mathsf{shadow}}, \tilde{w})\big) = 1 \quad \text{and} \quad P_{\mathsf{Valid}}\big(\hat{\mathbf{ct}}^{\mathsf{main}}, \hat{\mathbf{ct}}^{\mathsf{shadow}}, \mathsf{sk}_{\mathsf{main}}, \mathsf{sk}_{\mathsf{shadow}},\big) = 0.$$

We consider two possibilities:

- Suppose $j = i^*$. By construction of $C_{i^*,0}$ (see Fig. 1), this means $\hat{\mathbf{ct}}^{(b)} = \mathbf{ct}_{\mathsf{zero}}^{(b)}$ for $b \in \{0,1\}$. By construction, $\mathbf{ct}_{\mathsf{zero}}^{(b)}$ is an encryption of $\mathbf{0}$ under pk_b. In this case, $P_{\mathsf{Valid}}\big(\hat{\mathbf{ct}}^{\mathsf{main}}, \hat{\mathbf{ct}}^{\mathsf{shadow}}, \mathsf{sk}_{\mathsf{main}}, \mathsf{sk}_{\mathsf{shadow}},\big) = 1$, which contradicts the premise.
- Suppose $j \neq i^*$. By construction of $C_{i^*,0}$, there are now two more possibilities:
  - Suppose for $b \in \{\mathsf{main}, \mathsf{shadow}\}$, $\hat{\mathbf{ct}}^{(b)} = \mathbf{ct}_{\mathsf{zero}}^{(b)}$. As in the first case, this means $\hat{\mathbf{ct}}^{\mathsf{main}}$ and $\hat{\mathbf{ct}}^{\mathsf{shadow}}$ both decrypt to $\mathbf{0}$ under $\mathsf{sk}_{\mathsf{main}}$ and $\mathsf{sk}_{\mathsf{shadow}}$, respectively. In this case $P_{\mathsf{Valid}}\big(\hat{\mathbf{ct}}^{\mathsf{main}}, \hat{\mathbf{ct}}^{\mathsf{shadow}}, \mathsf{sk}_{\mathsf{main}}, \mathsf{sk}_{\mathsf{shadow}},\big) = 1$, which again contradicts the premise.
  - Suppose for $b \in \{\mathsf{main}, \mathsf{shadow}\}$, $\hat{\mathbf{ct}}^{(b)} = \tilde{\mathbf{ct}}^{(b)}$. In this case, we also have
    - $\mathsf{Com.Verify}\big(\mathsf{crs}_{\mathsf{Com}}, \mathsf{com}_{\mathsf{hk}}^{\mathsf{main}}, j, \tilde{\mathbf{ct}}^{\mathsf{main}}, \sigma_{\mathsf{hk}}^{\mathsf{main}}\big) = 1$; and
    - $\mathsf{Com.Verify}\big(\mathsf{crs}_{\mathsf{Com}}, \mathsf{com}_{\mathsf{hk}}^{\mathsf{shadow}}, j, \tilde{\mathbf{ct}}^{\mathsf{shadow}}, \sigma_{\mathsf{hk}}^{\mathsf{shadow}}\big) = 1$.

    Suppose $\tilde{\mathbf{ct}}^{(b)} = \mathbf{ct}_j^{(b)}$ for all $b \in \{0,1\}$. In this case, since $j \neq i^*$, the ciphertexts $\mathbf{ct}_j^{\mathsf{main}}, \mathbf{ct}_j^{\mathsf{shadow}}$ are either both encryptions of $\mathbf{0}$ (if $j \notin S$) or both encryptions of $\mathbf{v}_j$ (if $j \in S$). This again contradicts the premise. Thus, if P_{Valid} is not satisfied, we conclude that there exists some $b \in \{0,1\}$ such that $\tilde{\mathbf{ct}}^{(b)} \neq \mathbf{ct}_j^{(b)}$.

Thus, there exists some $b \in \{0,1\}$ such that the following holds:

$$\tilde{\mathbf{ct}}^{(b)} \neq \mathbf{ct}_j^{(b)} \quad \text{and} \quad \mathsf{Com.Verify}\big(\mathsf{crs}_{\mathsf{Com}}, \mathsf{com}_{\mathsf{hk}}^{(b)}, j, \tilde{\mathbf{ct}}^{(b)}, \sigma_{\mathsf{hk}}^{(b)}\big) = 1.$$

Moreover, by correctness of Π_{Com}, we have that

$$\mathsf{Com.Verify}\big(\mathsf{crs}_{\mathsf{Com}}, \mathsf{com}_{\mathsf{hk}}^{(b)}, j, \mathbf{ct}_j^{(b)}, \sigma_{\mathsf{hk},j}^{(b)}\big) = 1.$$

In this case, algorithm $\mathcal{B}$ successfully breaks the binding property of the commitment scheme. □

Since for all $j \in [n]$, it holds that $\Pr[\mathsf{Expt}_j(\mathcal{A}) = 1] = \mathsf{negl}(\lambda)$, we can invoke Theorem 4.12 to conclude that $\Pr[\mathsf{Expt}(\mathcal{A}) = 1] = \mathsf{negl}(\lambda)$. Claim 4.27 now follows via Eqs. (4.5) and (4.6). □

Claim 4.30. *If Π_{HE} is CPA-secure, then there exists a negligible function $\mathsf{negl}(\cdot)$ such that*

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

Proof. This follows by an analogous argument as the proof of Claim 4.26. In particular, the reduction obtains $\mathsf{pk}_{\mathsf{main}}$ and $\mathbf{ct}_{i^*}^{\mathsf{main}}$ from the challenger. It samples $(\mathsf{pk}_{\mathsf{shadow}}, \mathsf{sk}_{\mathsf{shadow}})$ itself which it can use to compute the output (according to the specification in Hyb_2 and Hyb_3). □

Claim 4.31. *If Π_{HE} is perfectly correct and satisfies evaluation correctness, Π_{Com} is computationally binding, and Π_{BARG} satisfies set hiding with extraction, set hiding, and is somewhere extractable, then there exists a negligible function $\mathsf{negl}(\cdot)$ such that $|\Pr[\mathsf{Hyb}_4(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_3(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda)$.*

Proof. This follows by an analogous argument as the proof of Claim 4.27. The only difference is that we take the mapping DeriveChal to be

$$\mathsf{DeriveChal}(S, i) := (S, i) \mapsto (S, S, (i, 0)).$$

The rest of the analysis proceeds exactly as before. □

Theorem 4.25 now follows by combining Claims 4.26, 4.27, 4.30 and 4.31. □

Acknowledgments

We would like to thank the PKC 2025 reviewers for their useful comments. Brent Waters is supported by NSF CNS-1908611, CNS-2318701, and a Simons Investigator award. David J. Wu is supported by NSF CNS-2140975, CNS-2318701, a Microsoft Research Faculty Fellowship, and a Google Research Scholar award.

References

[ADM+24] Gennaro Avitabile, Nico Döttling, Bernardo Magri, Christos Sakkas, and Stella Wohnig. Signature-based witness encryption with compact ciphertext. In *ASIACRYPT*, pages 3–31, 2024.

[BBK+23] Zvika Brakerski, Maya Farber Brodsky, Yael Tauman Kalai, Alex Lombardi, and Omer Paneth. SNARGs for monotone policy batch NP. In *CRYPTO*, pages 252–283, 2023.

[BCJP24] Maya Farber Brodsky, Arka Rai Choudhuri, Abhishek Jain, and Omer Paneth. Monotone-policy aggregate signatures. In *EUROCRYPT*, pages 168–195, 2024.

[Ben94] Josh Benaloh. Dense probabilistic encryption. In *SAC*, pages 120–128, 1994.

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

[BKP+24] Nir Bitansky, Chethan Kamath, Omer Paneth, Ron D. Rothblum, and Prashant Nalini Vasudevan. Batch proofs are statistically hiding. In *STOC*, pages 435–443, 2024.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*, pages 97–106, 2011.

[BWW24] Eli Bradley, Brent Waters, and David J. Wu. Batch arguments to nizks from one-way functions. In *TCC*, 2024.

[CGJ+23] Arka Rai Choudhuri, Sanjam Garg, Abhishek Jain, Zhengzhong Jin, and Jiaheng Zhang. Correlation intractability and snargs from sub-exponential DDH. In *CRYPTO*, pages 635–668, 2023.

[CJJ21a] Arka Rai Choudhuri, Abhishek Jain, and Zhengzhong Jin. Non-interactive batch arguments for NP from standard assumptions. In *CRYPTO*, pages 394–423, 2021.

[CJJ21b] Arka Rai Choudhuri, Abhishek Jain, and Zhengzhong Jin. SNARGs for P from LWE. In *FOCS*, pages 68–79, 2021.

[CLTV15] Ran Canetti, Huijia Lin, Stefano Tessaro, and Vinod Vaikuntanathan. Obfuscation of probabilistic circuits and applications. In *TCC*, pages 468–497, 2015.

[CW23] Jeffrey Champion and David J. Wu. Non-interactive zero-knowledge from non-interactive batch arguments. In *CRYPTO*, pages 38–71, 2023.

[DDO+01] Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In *CRYPTO*, pages 566–598, 2001.

[DGKV22] Lalita Devadas, Rishab Goyal, Yael Kalai, and Vinod Vaikuntanathan. Rate-1 non-interactive arguments for batch-NP and applications. In *FOCS*, pages 1057–1068, 2022.

[FWW23] Cody Freitag, Brent Waters, and David J. Wu. How to use (plain) witness encryption: Registered abe, flexible broadcast, and more. In *CRYPTO*, pages 498–531, 2023.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.

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

[GM82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *STOC*, pages 365–377, 1982.

[GO92] Shafi Goldwasser and Rafail Ostrovsky. Invariant signatures and non-interactive zero-knowledge proofs are equivalent (extended abstract). In *CRYPTO*, pages 228–245, 1992.

[GVW19] Rishab Goyal, Satyanarayana Vusirikala, and Brent Waters. Collusion resistant broadcast and trace from positional witness encryption. In *PKC*, pages 3–33, 2019.

[HLR07] Chun-Yuan Hsiao, Chi-Jen Lu, and Leonid Reyzin. Conditional computational entropy, or toward separating pseudoentropy from compressibility. In *EUROCRYPT*, pages 169–186, 2007.

[IKO05] Yuval Ishai, Eyal Kushilevitz, and Rafail Ostrovsky. Sufficient conditions for collision-resistant hashing. In *TCC*, pages 445–456, 2005.

[KLVW23] Yael Kalai, Alex Lombardi, Vinod Vaikuntanathan, and Daniel Wichs. Boosting batch arguments and RAM delegation. In *STOC*, pages 1545–1552, 2023.

[KVZ21] Yael Tauman Kalai, Vinod Vaikuntanathan, and Rachel Yun Zhang. Somewhere statistical soundness, post-quantum security, and SNARGs. In *TCC*, pages 330–368, 2021.

[Mer87] Ralph C. Merkle. A digital signature based on a conventional encryption function. In *CRYPTO*, pages 369–378, 1987.

[NWW24] Shafik Nassar, Brent Waters, and David J. Wu. Monotone policy BARGs from BARGs and additively homomorphic encryption. In *TCC*, 2024.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *STOC*, pages 427–437, 1990.

[Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *FOCS*, pages 543–553, 1999.

[WW22] Brent Waters and David J. Wu. Batch arguments for NP and more from standard bilinear group assumptions. In *CRYPTO*, pages 433–463, 2022.

A Proof of Theorem 4.12 (Predicate Propagation)

Our proof follows a very similar structure as the corresponding proof from [NWW24, Theorem 5.9]. As noted in Remark 4.13, we cannot use the proof from [NWW24] as a black box. For this reason, we reproduce the analysis here. Some parts of the description are taken verbatim from [NWW24, Theorem 5.9]. To simplify notation, we write $\mathsf{Expt} := \mathsf{Expt}[P, \mathsf{DeriveChal}]$ and $\mathsf{Expt}_j := \mathsf{Expt}_j[P, \mathsf{DeriveChal}]$ in the following proof. Fix an adversary $\mathcal{A}$ and let n be the input length chosen by $\mathcal{A}$. We proceed by induction on the index $j \in [2n-1]$. In the following, we will view the index j as an index of a node in a (complete) binary tree with n leaves (indexed according to Definition 4.1). As such, we can refer to the "height" of an index j. Then, we show the following lemma:

Lemma A.1. *Suppose the conditions of Theorem 4.12 hold. Take any index $j \in [2n-1]$ and let h be the height of node j (where the leaf nodes have height 0). Then, there exists a negligible function $\varepsilon_j(\lambda)$ such that*

$$\Pr[\mathsf{Expt}_j(\mathcal{A}) = 1] = 2^h \cdot \varepsilon_j(\lambda).$$

Proof. Suppose the conditions of Theorem 4.12 hold. We prove the lemma by induction on the height h of the index $j \in [2n-1]$.

Base case. For the indices $j \in [n]$ of height 0 (i.e., the leaves of the tree), the lemma follows by assumption.

Inductive step. Suppose the inductive hypothesis holds for every index $j' \in [2n-1]$ of height h. Let $j \in [2n-1]$ be an index with height $h+1$. Let $j_{\mathrm{L}}, j_{\mathrm{R}} \in [2n-1]$ be the indices of the left and right child of node j (as defined in Definition 4.1). By construction, j_{L} and j_{R} have height h. The inductive hypothesis now asserts that for $j^* \in \{j_{\mathrm{L}}, j_{\mathrm{R}}\}$,

$$\Pr\left[\mathsf{Expt}_{j^*}(\mathcal{A}) = 1\right] = 2^h \cdot \varepsilon_{j^*}(\lambda), \tag{A.1}$$

for some negligible function $\varepsilon_{j^*}(\lambda)$. We now define an intermediate experiment Expt'_j for each node j of height $h > 0$:

1. On input the security parameter 1^λ, algorithm $\mathcal{A}$ outputs the input length 1^n, a set $S \subseteq [n]$, and an index $i^* \in S$ (or a special symbol $\perp$).
2. The challenger computes $(S_{\mathsf{main}}, S_{\mathsf{shadow}}, \mathsf{idx}) \leftarrow \mathsf{DeriveChal}(S, i^*)$.
3. The challenger samples the following quantities as in Setup:
   - $(\mathsf{sk}_{\mathsf{main}}, \mathsf{pk}_{\mathsf{main}}) \leftarrow \mathsf{HE.Gen}(1^\lambda, 1^n)$ and $(\mathsf{sk}_{\mathsf{shadow}}, \mathsf{pk}_{\mathsf{shadow}}) \leftarrow \mathsf{HE.Gen}(1^\lambda, 1^n)$
   - $(\mathsf{crs}_{\mathsf{BARG}}, \mathsf{vk}_{\mathsf{BARG}}, \mathsf{td}_{\mathsf{BARG}}) \leftarrow \mathsf{TrapGen}(1^\lambda, 1^{2n}, 1^s, 1^3, \{j, j_{\mathrm{L}}, j_{\mathrm{R}}\})$.
   - $\mathsf{crs}_{\mathsf{Com}} \leftarrow \mathsf{Com.Setup}(1^\lambda, 1^{\lambda \cdot \ell_{\mathsf{ct}}(\lambda)}, 2n-1)$.
   - $\mathbf{ct}_{\mathsf{zero}}^{(b)} \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_b, \mathbf{0})$ for all $b \in \{\mathsf{main}, \mathsf{shadow}\}$.
   - For all $i \in [n]$, sample a random $\mathbf{v}_i \xleftarrow{\mathrm{R}} \{0,1\}^\lambda \setminus \{\mathbf{0}\}$.
   - For all $i \in [n], b \in \{\mathsf{main}, \mathsf{shadow}\}$, if $i \in S_b$ then sample $\mathbf{ct}_i^{(b)} \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_b, \mathbf{v}_i)$, otherwise sample $\mathbf{ct}_i^{(b)} \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_b, \mathbf{0})$.
   - $\left(\mathsf{com}_{\mathsf{hk}}^{(b)}, \sigma_{\mathsf{hk},1}^{(b)}, \ldots, \sigma_{\mathsf{hk},n}^{(b)}\right) \leftarrow \mathsf{Com.Commit}\left(\mathsf{crs}_{\mathsf{Com}}, (\mathbf{ct}_1^{(b)}, \ldots, \mathbf{ct}_n^{(b)})\right)$ for all $b \in \{\mathsf{main}, \mathsf{shadow}\}$.
4. The challenger constructs hk and vk as defined in Eqs. (4.1) and (4.2):

$$\mathsf{hk} = \left(\mathsf{crs}_{\mathsf{Com}}, \mathsf{crs}_{\mathsf{BARG}}, \left\{\mathsf{pk}_b, \mathbf{ct}_{\mathsf{zero}}^{(b)}, \mathbf{ct}_1^{(b)}, \ldots, \mathbf{ct}_n^{(b)}, \sigma_{\mathsf{hk},1}^{(b)}, \ldots, \sigma_{\mathsf{hk},n}^{(b)}\right\}_{b \in \{\mathsf{main},\mathsf{shadow}\}}\right)$$
$$\mathsf{vk} = \left(\mathsf{crs}_{\mathsf{Com}}, \mathsf{vk}_{\mathsf{BARG}}, \left\{\mathsf{pk}_b, \mathbf{ct}_{\mathsf{zero}}^{(b)}, \mathsf{com}_{\mathsf{hk}}^{(b)}\right\}_{b \in \{\mathsf{main},\mathsf{shadow}\}}\right)$$

   The challenger gives $(\mathsf{hk}, \mathsf{vk})$ to $\mathcal{A}$.
5. Algorithm $\mathcal{A}$ outputs a digest $\mathsf{dig} = (\mathbf{ct}_{\mathsf{root}}^{\mathsf{main}}, \mathbf{ct}_{\mathsf{root}}^{\mathsf{shadow}}, \mathsf{com}_{\mathsf{main}}, \mathsf{com}_{\mathsf{shadow}}, \pi_{\mathsf{dig}})$ and a proof π.
6. The challenger computes $(\hat{\mathbf{ct}}_j^{\mathsf{main}}, \hat{\mathbf{ct}}_j^{\mathsf{shadow}}, \sigma_j^{\mathsf{main}}, \sigma_j^{\mathsf{shadow}}, \tilde{w}_j) \leftarrow \mathsf{BARG.Extract}(\mathsf{td}_{\mathsf{BARG}}, \pi, j)$.
7. The output of the experiment is 1 if all of the following conditions hold, and 0 otherwise:
   (a) $\mathsf{BARG.Verify}(\mathsf{crs}_{\mathsf{BARG}}, C_{\mathsf{idx}}, (1, \ldots, 2n-1), \pi) = 1$.
   (b) $C_{\mathsf{idx}}(j, (\hat{\mathbf{ct}}_j^{\mathsf{main}}, \hat{\mathbf{ct}}_j^{\mathsf{shadow}}, \sigma_j^{\mathsf{main}}, \sigma_j^{\mathsf{shadow}}, \tilde{w}_j)) = 1$.
   (c) $P(\hat{\mathbf{ct}}_j^{\mathsf{main}}, \hat{\mathbf{ct}}_j^{\mathsf{shadow}}, \mathsf{sk}_{\mathsf{main}}, \mathsf{sk}_{\mathsf{shadow}}, j, (\mathbf{v}_1, \ldots, \mathbf{v}_n, \mathsf{idx})) = 0$.

In our analysis below, we define an additional set of events in an execution of Expt'_j with $\mathcal{A}$. First, define the following two quantities:

- $(\hat{\mathbf{ct}}_{j_{\mathrm{L}}}^{\mathsf{main}}, \hat{\mathbf{ct}}_{j_{\mathrm{L}}}^{\mathsf{shadow}}, \sigma_{j_{\mathrm{L}}}^{\mathsf{main}}, \sigma_{j_{\mathrm{L}}}^{\mathsf{shadow}}, \tilde{w}_{j_{\mathrm{L}}}) \leftarrow \mathsf{BARG.Extract}(\mathsf{td}_{\mathsf{BARG}}, \pi, j_{\mathrm{L}})$.
- $(\hat{\mathbf{ct}}_{j_{\mathrm{R}}}^{\mathsf{main}}, \hat{\mathbf{ct}}_{j_{\mathrm{R}}}^{\mathsf{shadow}}, \sigma_{j_{\mathrm{R}}}^{\mathsf{main}}, \sigma_{j_{\mathrm{R}}}^{\mathsf{shadow}}, \tilde{w}_{j_{\mathrm{R}}}) \leftarrow \mathsf{BARG.Extract}(\mathsf{td}_{\mathsf{BARG}}, \pi, j_{\mathrm{R}})$.

Now, define the following events:

- $E_{\mathsf{Verify}}^{(j)}$: This is the event that $\mathsf{BARG.Verify}(\mathsf{vk}_{\mathsf{BARG}}, C_{\mathsf{idx}}, 2n-1, \pi) = 1$.

- $E^{(j)}_{P,j^*}$ for each $j^* \in \{j, j_{\mathsf{L}}, j_{\mathsf{R}}\}$: This is the event where $P\big(\hat{\mathbf{ct}}^{\mathsf{main}}_{j^*}, \hat{\mathbf{ct}}^{\mathsf{shadow}}_{j^*}, \mathsf{sk}_{\mathsf{main}}, \mathsf{sk}_{\mathsf{shadow}}, j^*, (\mathbf{v}_1, \ldots, \mathbf{v}_n, \mathsf{idx})\big) = 1$.
- $E^{(j)}_{\mathsf{ValidCom},j^*}$ for each $j^* \in \{j_{\mathsf{L}}, j_{\mathsf{R}}\}$: This is the event

$$\mathsf{Com.Verify}\big(\mathsf{crs}_{\mathsf{Com}}, \mathsf{com}_{\mathsf{main}}, j^*, \hat{\mathbf{ct}}^{\mathsf{main}}_{j^*}, \sigma^{\mathsf{main}}_{j^*}\big) = 1 = \mathsf{Com.Verify}\big(\mathsf{crs}_{\mathsf{Com}}, \mathsf{com}_{\mathsf{shadow}}, j^*, \hat{\mathbf{ct}}^{\mathsf{shadow}}_{j^*}, \sigma^{\mathsf{shadow}}_{j^*}\big).$$

- $E^{(j)}_{\mathsf{CorrectWit},j^*}$ for each $j^* \in \{j, j_{\mathsf{L}}, j_{\mathsf{R}}\}$: This is the event $C_{\mathsf{idx}}\big(j^*, (\hat{\mathbf{ct}}^{\mathsf{main}}_{j^*}, \hat{\mathbf{ct}}^{\mathsf{shadow}}_{j^*}, \sigma^{\mathsf{main}}_{j^*}, \sigma^{\mathsf{shadow}}_{j^*}, \tilde{w}_{j^*})\big) = 1$.

We now relate the probability that $\mathsf{Expt}_j(\mathcal{A})$ outputs 1 to the probability that $\mathsf{Expt}_{j_{\mathsf{L}}}(\mathcal{A})$ and $\mathsf{Expt}_{j_{\mathsf{R}}}(\mathcal{A})$ outputs 1. To do so, we first program the BARG to be extracting on the set $\{j, j_{\mathsf{L}}, j_{\mathsf{R}}\}$. We then argue via somewhere extractability of the BARG and computational binding of the commitment scheme that if the values associated with the nodes j_{L} and j_{R} satisfy the predicate P and the proof verifies, then the value associated with j must also satisfy the predicate P. In this case, the output of $\mathsf{Expt}_j(\mathcal{A})$ is guaranteed to be 0.

Claim A.2. *If Π_{BARG} satisfies set hiding with extraction, then there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $j^* \in \{j, j_{\mathsf{L}}, j_{\mathsf{R}}\}$, it holds that*

$$\left|\Pr[\mathsf{Expt}_{j^*}(\mathcal{A}) = 1] - \Pr\left[E^{(j)}_{\mathsf{Verify}} \wedge E^{(j)}_{\mathsf{CorrectWit},j^*} \wedge \neg E^{(j)}_{P,j^*}\right]\right| = \mathsf{negl}(\lambda).$$

Proof. Take any $j^* \in \{j, j_{\mathsf{L}}, j_{\mathsf{R}}\}$ and suppose

$$\left|\Pr[\mathsf{Expt}_{j^*}(\mathcal{A}) = 1] - \Pr\left[E^{(j)}_{\mathsf{Verify}} \wedge E^{(j)}_{\mathsf{CorrectWit},j^*} \wedge \neg E^{(j)}_{P,j^*}\right]\right| = \varepsilon$$

for some non-negligible ε. Importantly, note that the events $E^{(j)}_{\mathsf{Verify}}$, $E^{(j)}_{\mathsf{CorrectWit},j^*}$, and $E^{(j)}_{P,j^*}$ are defined for Expt'_j and *not* Expt_{j^*}. We use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ for the set hiding with extraction game of Π_{BARG}:

1. On input the security parameter 1^λ, algorithm $\mathcal{B}$ runs algorithm $\mathcal{A}$ to obtain the input length 1^n, the set $S \subseteq [n]$, and an index $i^* \in S$.
2. Algorithm $\mathcal{B}$ outputs 1^{2n-1}, 1^s, 1^3, the challenge set $J = \{j, j_{\mathsf{L}}, j_{\mathsf{R}}\}$, and the challenge index $j^* \in J$ to the challenger, where s is the bound on the size of the circuit in Fig. 1. The challenger responds with $(\mathsf{crs}_{\mathsf{BARG}}, \mathsf{vk}_{\mathsf{BARG}})$.
3. Algorithm $\mathcal{B}$ computes $(S_0, S_1, \mathsf{idx}) \leftarrow \mathsf{DeriveChal}(S, i^*)$. It then samples the following components:
   - $(\mathsf{sk}_{\mathsf{main}}, \mathsf{pk}_{\mathsf{main}}) \leftarrow \mathsf{HE.Gen}(1^\lambda, 1^n)$, $(\mathsf{sk}_{\mathsf{shadow}}, \mathsf{pk}_{\mathsf{shadow}}) \leftarrow \mathsf{HE.Gen}(1^\lambda, 1^n)$.
   - $\mathsf{crs}_{\mathsf{Com}} \leftarrow \mathsf{Com.Setup}(1^\lambda, 1^{\lambda \cdot \ell_{\mathsf{ct}}(\lambda,n)}, 2n-1)$.
   - For all $i \in [n]$, sample a random $\mathbf{v}_i \xleftarrow{\mathrm{R}} \{0,1\}^\lambda \setminus \{\mathbf{0}\}$.
   - For each $b \in \{\mathsf{main}, \mathsf{shadow}\}$, sample $\mathbf{ct}^{(b)}_{\mathsf{zero}} \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_b, \mathbf{0})$. Then, for each $i \in [n]$ and $b \in \{\mathsf{main}, \mathsf{shadow}\}$, if $i \in S_b$, sample $\mathbf{ct}^{(b)}_i \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_b, \mathbf{v}_i)$; otherwise, if $i \notin S_b$, sample $\mathsf{ct}^{(b)}_i \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_b, \mathbf{0})$.
   - For each $b \in \{\mathsf{main}, \mathsf{shadow}\}$, let $\big(\mathsf{com}^{(b)}_{\mathsf{hk}}, \sigma^{(b)}_{\mathsf{hk},1}, \ldots, \sigma^{(b)}_{\mathsf{hk},n}\big) \leftarrow \mathsf{Com.Commit}\big(\mathsf{crs}_{\mathsf{Com}}, (\mathbf{ct}^{(b)}_1, \ldots, \mathbf{ct}^{(b)}_n)\big)$.
4. Algorithm $\mathcal{B}$ constructs hk and vk according to Eqs. (4.1) and (4.2):

$$\mathsf{hk} = \left(\mathsf{crs}_{\mathsf{Com}}, \mathsf{crs}_{\mathsf{BARG}}, \left\{\mathsf{pk}_b, \mathbf{ct}^{(b)}_{\mathsf{zero}}, \mathbf{ct}^{(b)}_1, \ldots, \mathbf{ct}^{(b)}_n, \sigma^{(b)}_{\mathsf{hk},1}, \ldots, \sigma^{(b)}_{\mathsf{hk},n}\right\}_{b \in \{\mathsf{main},\mathsf{shadow}\}}\right)$$
$$\mathsf{vk} = \left(\mathsf{crs}_{\mathsf{Com}}, \mathsf{vk}_{\mathsf{BARG}}, \left\{\mathsf{pk}_b, \mathbf{ct}^{(b)}_{\mathsf{zero}}, \mathsf{com}^{(b)}_{\mathsf{hk}}\right\}_{b \in \{\mathsf{main},\mathsf{shadow}\}}\right)$$

   Algorithm $\mathcal{B}$ gives $(\mathsf{hk}, \mathsf{vk})$ to $\mathcal{A}$.
5. Algorithm $\mathcal{A}$ outputs a digest $\mathsf{dig} = \big(\mathbf{ct}^{\mathsf{main}}_{\mathsf{root}}, \mathbf{ct}^{\mathsf{shadow}}_{\mathsf{root}}, \mathsf{com}_{\mathsf{main}}, \mathsf{com}_{\mathsf{shadow}}, \pi_{\mathsf{dig}}\big)$ and a proof π.

6. Let C_{idx} be the circuit as defined in Definition 4.10. Algorithm $\mathcal{B}$ first checks

$$\mathsf{BARG.Verify}(\mathsf{vk}_{\mathsf{BARG}}, C_{\mathsf{idx}}, 2n-1, \pi) = 1.$$

   If the check fails, algorithm $\mathcal{B}$ aborts with output $\perp$. Otherwise, algorithm $\mathcal{B}$ sends the circuit C_{idx}, the instance number $2n-1$, and the proof π to the challenger. The challenger replies with a string which $\mathcal{B}$ parses as $(\hat{\mathbf{ct}}^{\mathsf{main}}_{j^*}, \hat{\mathbf{ct}}^{\mathsf{shadow}}_{j^*}, \sigma^{\mathsf{main}}_{j^*}, \sigma^{\mathsf{shadow}}_{j^*}, \tilde{w}_{j^*})$.

7. Algorithm $\mathcal{B}$ outputs 1 all of the following conditions hold:

   - $C_{\mathsf{idx}}\big(j^*, (\hat{\mathbf{ct}}^{\mathsf{main}}_{j^*}, \hat{\mathbf{ct}}^{\mathsf{shadow}}_{j^*}, \sigma^{\mathsf{main}}_{j^*}, \sigma^{\mathsf{shadow}}_{j^*}, \tilde{w}_{j^*})\big) = 1$.
   - $P\big(\hat{\mathbf{ct}}^{\mathsf{main}}_{j^*}, \hat{\mathbf{ct}}^{\mathsf{shadow}}_{j^*}, \mathsf{sk}_{\mathsf{main}}, \mathsf{sk}_{\mathsf{shadow}}, j^*, (\mathbf{v}_1, \ldots, \mathbf{v}_n, \mathsf{idx})\big) = 0$.

   Otherwise, algorithm $\mathcal{B}$ outputs 0.

Let $(\mathsf{crs}_{\mathsf{BARG}}, \mathsf{vk}_{\mathsf{BARG}}, \mathsf{td}_{\mathsf{BARG}})$ be the parameters sampled by the challenger in the set hiding with extraction game. In the game, after $\mathcal{B}$ outputs $(C_{\mathsf{idx}}, 2n-1, \pi)$, the challenger checks $\mathsf{BARG.Verify}(\mathsf{vk}_{\mathsf{BARG}}, C_{\mathsf{idx}}, 2n-1, \pi) = 1$. If the check passes, it replies with $(\hat{\mathbf{ct}}^{\mathsf{main}}_{j^*}, \hat{\mathbf{ct}}^{\mathsf{shadow}}_{j^*}, \sigma^{\mathsf{main}}_{j^*}, \sigma^{\mathsf{shadow}}_{j^*}, \tilde{w}_{j^*})$. We now consider the two possibilities:

- Suppose the challenger responds according to the specification of $\mathsf{ExptIHE}_{\mathcal{A}}(\lambda, 0)$. In this case, the challenger samples $(\mathsf{crs}_{\mathsf{BARG}}, \mathsf{vk}_{\mathsf{BARG}}, \mathsf{td}_{\mathsf{BARG}}) \leftarrow \mathsf{BARG.TrapGen}(1^\lambda, 1^{2n-1}, 1^s, 1^3, \{j, j_{\mathrm{L}}, j_{\mathrm{R}}\})$. Thus, algorithm $\mathcal{B}$ perfectly simulates for $\mathcal{A}$ an execution of Expt'_j. We claim that algorithm $\mathcal{B}$ outputs 1 if and only if the event $E^{(j)}_{\mathsf{Verify}} \wedge E^{(j)}_{\mathsf{CorrectWit},j^*} \wedge \neg E^{(j)}_{P,j^*}$ occurs. This event corresponds to the conjunction of the following set of conditions:
  - $\mathsf{BARG.Verify}(\mathsf{vk}_{\mathsf{BARG}}, C_{\mathsf{idx}}, 2n-1, \pi) = 1$.
  - $\mathsf{BARG.Verify}(\mathsf{vk}_{\mathsf{BARG}}, C_{\mathsf{idx}}, 2n-1, \pi) = 1$ and $C_{\mathsf{idx}}\big(j^*, (\hat{\mathbf{ct}}^{\mathsf{main}}_{j^*}, \hat{\mathbf{ct}}^{\mathsf{shadow}}_{j^*}, \sigma^{\mathsf{main}}_{j^*}, \sigma^{\mathsf{shadow}}_{j^*}, \tilde{w}_{j^*})\big) = 1$.
  - $P\big(\hat{\mathbf{ct}}^{\mathsf{main}}_{j^*}, \hat{\mathbf{ct}}^{\mathsf{shadow}}_{j^*}, \mathsf{sk}_{\mathsf{main}}, \mathsf{sk}_{\mathsf{shadow}}, j^*, (\mathbf{v}_1, \ldots, \mathbf{v}_n, \mathsf{idx})\big) = 0$.

  where $(\hat{\mathbf{ct}}^{\mathsf{main}}_{j^*}, \hat{\mathbf{ct}}^{\mathsf{shadow}}_{j^*}, \sigma^{\mathsf{main}}_{j^*}, \sigma^{\mathsf{shadow}}_{j^*}, \tilde{w}_{j^*}) \leftarrow \mathsf{BARG.Extract}(\mathsf{td}_{\mathsf{BARG}}, \pi, j^*)$. This is the same set of conditions that algorithm $\mathcal{B}$ checks, so algorithm $\mathcal{B}$ outputs 1 with probability $\Pr\left[E^{(j)}_{\mathsf{Verify}} \wedge E^{(j)}_{\mathsf{CorrectWit},j^*} \wedge \neg E^{(j)}_{P,j^*}\right]$ in this case.
- Suppose the challenger responds according to the specification of $\mathsf{ExptIHE}_{\mathcal{A}}(\lambda, 1)$. In this case, the challenger samples $(\mathsf{crs}_{\mathsf{BARG}}, \mathsf{vk}_{\mathsf{BARG}}, \mathsf{td}_{\mathsf{BARG}}) \leftarrow \mathsf{BARG.TrapGen}(1^\lambda, 1^{2n-1}, 1^s, 1^3, \{j^*\})$. Thus, algorithm $\mathcal{B}$ simulates for $\mathcal{A}$ an execution of Expt_{j^*}. We claim that algorithm $\mathcal{B}$ outputs 1 if and only if $\mathsf{Expt}_{j^*}(\mathcal{A})$ outputs 1. The latter corresponds to the conjunction of the following set of conditions:
  - $\mathsf{BARG.Verify}(\mathsf{vk}_{\mathsf{BARG}}, C_{\mathsf{idx}}, 2n-1, \pi) = 1$.
  - $\mathsf{BARG.Verify}(\mathsf{vk}_{\mathsf{BARG}}, C_{\mathsf{idx}}, 2n-1, \pi) = 1$ and $C_{\mathsf{idx}}\big(j^*, (\hat{\mathbf{ct}}^{\mathsf{main}}_{j^*}, \hat{\mathbf{ct}}^{\mathsf{shadow}}_{j^*}, \sigma^{\mathsf{main}}_{j^*}, \sigma^{\mathsf{shadow}}_{j^*}, \tilde{w}_{j^*})\big) = 1$.
  - $P\big(\hat{\mathbf{ct}}^{\mathsf{main}}_{j^*}, \hat{\mathbf{ct}}^{\mathsf{shadow}}_{j^*}, \mathsf{sk}_{\mathsf{main}}, \mathsf{sk}_{\mathsf{shadow}}, j^*, (\mathbf{v}_1, \ldots, \mathbf{v}_n, \mathsf{idx})\big) = 0$.

  where $(\hat{\mathbf{ct}}^{\mathsf{main}}_{j^*}, \hat{\mathbf{ct}}^{\mathsf{shadow}}_{j^*}, \sigma^{\mathsf{main}}_{j^*}, \sigma^{\mathsf{shadow}}_{j^*}, \tilde{w}_{j^*}) \leftarrow \mathsf{BARG.Extract}(\mathsf{td}_{\mathsf{BARG}}, \pi, j^*)$. Once again, this is the same set of conditions that $\mathcal{B}$ checks. Thus, in this case algorithm $\mathcal{B}$ outputs 1 with probability $\Pr[\mathsf{Expt}_{j^*}(\mathcal{A}) = 1]$.

We conclude that the distinguishing advantage of $\mathcal{B}$ is precisely

$$\left|\Pr[\mathsf{Expt}_{j^*}(\mathcal{A}) = 1] - \Pr\left[E^{(j)}_{\mathsf{Verify}} \wedge E^{(j)}_{\mathsf{CorrectWit},j^*} \wedge \neg E^{(j)}_{P,j^*}\right]\right| = \varepsilon,$$

which completes the proof. □

Claim A.3. *If Π_{BARG} is somewhere extractable then there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $j^* \in \{j, j_L, j_R\}$, it holds that $\Pr\left[E^{(j)}_{\mathsf{Verify}} \wedge \neg E^{(j)}_{\mathsf{CorrectWit},j^*}\right] = \mathsf{negl}(\lambda)$.*

Proof. Take any $j^* \in \{j, j_L, j_R\}$ and suppose $\Pr\left[E^{(j)}_{\mathsf{Verify}} \wedge \neg E^{(j)}_{\mathsf{CorrectWit},j^*}\right] \geq \varepsilon$. We use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ for the somewhere extractability game of Π_{BARG}:

1. On input the security parameter 1^λ, algorithm $\mathcal{B}$ runs algorithm $\mathcal{A}$ to obtain the input length 1^n, the set $S \subseteq [n]$, and an index $i^* \in S$.
2. Algorithm $\mathcal{B}$ outputs $1^{2n-1}, 1^s, 1^3$, the challenge set $J = \{j, j_L, j_R\}$, and the challenge index $j^* \in J$ to the challenger, where s is the bound on the size of the circuit in Fig. 1. The challenger responds with $(\mathsf{crs}_{\mathsf{BARG}}, \mathsf{vk}_{\mathsf{BARG}})$.
3. Algorithm $\mathcal{B}$ computes $(S_0, S_1, \mathsf{idx}) \leftarrow \mathsf{DeriveChal}(S, i^*)$. It then samples the following components:
   - $(\mathsf{sk}_{\mathsf{main}}, \mathsf{pk}_{\mathsf{main}}) \leftarrow \mathsf{HE.Gen}(1^\lambda, 1^n)$, $(\mathsf{sk}_{\mathsf{shadow}}, \mathsf{pk}_{\mathsf{shadow}}) \leftarrow \mathsf{HE.Gen}(1^\lambda, 1^n)$.
   - $\mathsf{crs}_{\mathsf{Com}} \leftarrow \mathsf{Com.Setup}(1^\lambda, 1^{\lambda \cdot \ell_{\mathsf{ct}}(\lambda,n)}, 2n-1)$.
   - For all $i \in [n]$, sample a random $\mathbf{v}_i \xleftarrow{R} \{0,1\}^\lambda \setminus \{\mathbf{0}\}$.
   - For each $b \in \{\mathsf{main}, \mathsf{shadow}\}$, sample $\mathbf{ct}^{(b)}_{\mathsf{zero}} \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_b, \mathbf{0})$. Then, for each $i \in [n]$ and $b \in \{\mathsf{main}, \mathsf{shadow}\}$, if $i \in S_b$, sample $\mathbf{ct}^{(b)}_i \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_b, \mathbf{v}_i)$; otherwise, if $i \notin S_b$, sample $\mathbf{ct}^{(b)}_i \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_b, \mathbf{0})$.
   - For each $b \in \{\mathsf{main}, \mathsf{shadow}\}$, let $\left(\mathsf{com}^{(b)}_{\mathsf{hk}}, \sigma^{(b)}_{\mathsf{hk},1}, \ldots, \sigma^{(b)}_{\mathsf{hk},n}\right) \leftarrow \mathsf{Com.Commit}\left(\mathsf{crs}_{\mathsf{Com}}, (\mathbf{ct}^{(b)}_1, \ldots, \mathbf{ct}^{(b)}_n)\right)$.
4. Algorithm $\mathcal{B}$ constructs hk and vk according to Eqs. (4.1) and (4.2):

$$\mathsf{hk} = \left(\mathsf{crs}_{\mathsf{Com}}, \mathsf{crs}_{\mathsf{BARG}}, \left\{\mathsf{pk}_b, \mathbf{ct}^{(b)}_{\mathsf{zero}}, \mathbf{ct}^{(b)}_1, \ldots, \mathbf{ct}^{(b)}_n, \sigma^{(b)}_{\mathsf{hk},1}, \ldots, \sigma^{(b)}_{\mathsf{hk},n}\right\}_{b \in \{\mathsf{main},\mathsf{shadow}\}}\right)$$
$$\mathsf{vk} = \left(\mathsf{crs}_{\mathsf{Com}}, \mathsf{vk}_{\mathsf{BARG}}, \left\{\mathsf{pk}_b, \mathbf{ct}^{(b)}_{\mathsf{zero}}, \mathsf{com}^{(b)}_{\mathsf{hk}}\right\}_{b \in \{\mathsf{main},\mathsf{shadow}\}}\right)$$

   Algorithm $\mathcal{B}$ gives $(\mathsf{hk}, \mathsf{vk})$ to $\mathcal{A}$.
5. Algorithm $\mathcal{A}$ outputs a digest $\mathsf{dig} = \left(\mathbf{ct}^{\mathsf{main}}_{\mathsf{root}}, \mathbf{ct}^{\mathsf{shadow}}_{\mathsf{root}}, \mathsf{com}_{\mathsf{main}}, \mathsf{com}_{\mathsf{shadow}}, \pi_{\mathsf{dig}}\right)$ and a proof π.
6. Let C_{idx} be the circuit as defined in Definition 4.10. Algorithm $\mathcal{B}$ outputs the circuit C_{idx}, the instance number $2n-1$, and the proof π.

By construction, algorithm $\mathcal{B}$ perfectly simulates an execution of Expt_j. Thus, with probability at least ε, the digest dig and proof π output by $\mathcal{A}$ satisfies $E^{(j)}_{\mathsf{Verify}}$ but not $E^{(j)}_{\mathsf{CorrectWit},j^*}$. This means

$$\mathsf{BARG.Verify}(\mathsf{vk}_{\mathsf{BARG}}, C_{\mathsf{idx}}, 2n-1, \pi) = 1 \quad \text{and} \quad C_{\mathsf{idx}}\left(j^*, (\hat{\mathbf{ct}}^{\mathsf{main}}_{j^*}, \hat{\mathbf{ct}}^{\mathsf{shadow}}_{j^*}, \sigma^{\mathsf{main}}_{j^*}, \sigma^{\mathsf{shadow}}_{j^*}, \tilde{w}_{j^*})\right) = 0.$$

This means algorithm $\mathcal{B}$ successfully wins the somewhere extractability game of Π_{BARG} with probability at least ε and the claim follows. □

Claim A.4. *Suppose the conditions in Claims A.2 and A.3 hold. Then, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that*

$$\Pr\left[\mathsf{Expt}'_j(\mathcal{A}) = 1 \wedge \left(\neg E^{(j)}_{\mathsf{ValidCom},j_L} \vee \neg E^{(j)}_{P,j_L} \vee \neg E^{(j)}_{\mathsf{ValidCom},j_R} \vee \neg E^{(j)}_{P,j_R}\right)\right] \leq 2^{h+1} \cdot \varepsilon_j(\lambda) + \mathsf{negl}(\lambda),$$

where $\varepsilon_j(\lambda) = \max(\varepsilon_{j_L}(\lambda), \varepsilon_{j_R}(\lambda))$.

Proof. By Claim A.2 there exists a negligible function $\mathsf{negl}_1(\cdot)$ such that for all $j^* \in \{j_L, j_R\}$, it holds that:

$$\left|\Pr[\mathsf{Expt}_{j^*}(\mathcal{A}) = 1] - \Pr\left[E^{(j)}_{\mathsf{Verify}} \wedge E^{(j)}_{\mathsf{CorrectWit},j^*} \wedge \neg E^{(j)}_{P,j^*}\right]\right| \leq \mathsf{negl}_1(\lambda). \tag{A.2}$$

By Claim A.3 there exists a negligible function $\mathsf{negl}_2(\cdot)$ such that for all $j^* \in \{j_L, j_R\}$ it holds that

$$\Pr\left[E^{(j)}_{\mathsf{Verify}} \wedge \neg E^{(j)}_{\mathsf{CorrectWit},j^*}\right] \leq \mathsf{negl}_2(\lambda). \tag{A.3}$$

By definition, if $\mathsf{Expt}'_j(\mathcal{A}) = 1$, then event $E^{(j)}_{\mathsf{Verify}}$ also occurs. Thus, for all events E, it holds that

$$\Pr[\mathsf{Expt}'_j(\mathcal{A}) = 1 \wedge E] \leq \Pr\left[E^{(j)}_{\mathsf{Verify}} \wedge E\right]. \tag{A.4}$$

Similarly, by construction of the circuit C_{idx}, the event $\neg E^{(j)}_{\mathsf{ValidCom},j^*}$ implies event $\neg E^{(j)}_{\mathsf{CorrectWit},j^*}$. Thus, for any event E, it holds that

$$\Pr\left[\neg E^{(j)}_{\mathsf{ValidCom},j^*} \wedge E\right] \leq \Pr\left[\neg E^{(j)}_{\mathsf{CorrectWit},j^*} \wedge E\right]. \tag{A.5}$$

Take any $j^* \in \{j_{\mathsf{L}}, j_{\mathsf{R}}\}$. Since the height of j^* is h, the inductive hypothesis applies and Eq. (A.1) holds. We first show that

$$\Pr\left[\mathsf{Expt}'_j(\mathcal{A}) = 1 \wedge \neg E^{(j)}_{P,j^*}\right] \leq 2^h \cdot \varepsilon_{j^*}(\lambda) + \mathsf{negl}_1(\lambda) + \mathsf{negl}_2(\lambda). \tag{A.6}$$

This follows by the following sequence of calculations:

$$\begin{aligned}
\Pr\left[\mathsf{Expt}'_j(\mathcal{A}) = 1 \wedge \neg E^{(j)}_{P,j^*}\right] &\leq \Pr\left[E^{(j)}_{\mathsf{Verify}} \wedge \neg E^{(j)}_{P,j^*}\right] && \text{by Eq. (A.4)} \\
&= \Pr\left[E^{(j)}_{\mathsf{Verify}} \wedge E^{(j)}_{\mathsf{CorrectWit},j^*} \wedge \neg E^{(j)}_{P,j^*}\right] + \Pr\left[E^{(j)}_{\mathsf{Verify}} \wedge \neg E^{(j)}_{\mathsf{CorrectWit},j^*} \wedge \neg E^{(j)}_{P,j^*}\right] \\
&\leq \Pr[E^{(j)}_{\mathsf{Verify}} \wedge E^{(j)}_{\mathsf{CorrectWit},j^*} \wedge \neg E^{(j)}_{P,j^*}] + \mathsf{negl}_2(\lambda) && \text{by Eq. (A.3)} \\
&\leq \Pr[\mathsf{Expt}_{j^*}(\mathcal{A}) = 1] + \mathsf{negl}_1(\lambda) + \mathsf{negl}_2(\lambda) && \text{by Eq. (A.2)} \\
&\leq 2^h \cdot \varepsilon_{j^*}(\lambda) + \mathsf{negl}_1(\lambda) + \mathsf{negl}_2(\lambda) && \text{by Eq. (A.1).}
\end{aligned}$$

Next, we have

$$\begin{aligned}
\Pr\left[\mathsf{Expt}'_j(\mathcal{A}) = 1 \wedge \neg E_{\mathsf{ValidCom},j^*}\right] &\leq \Pr\left[E^{(j)}_{\mathsf{Verify}} \wedge \neg E_{\mathsf{ValidCom},j^*}\right] && \text{by Eq. (A.4)} \\
&\leq \Pr\left[E^{(j)}_{\mathsf{Verify}} \wedge \neg E_{\mathsf{CorrectWit},j^*}\right] && \text{by Eq. (A.5)} \\
&\leq \mathsf{negl}_2(\lambda) && \text{by Eq. (A.3).}
\end{aligned}$$

Combined with Eq. (A.6) and applying a union bound, we have

$$\begin{aligned}
\Pr\left[\mathsf{Expt}'_j(\mathcal{A}) = 1 \wedge \left(\neg E^{(j)}_{\mathsf{ValidCom},j_{\mathsf{L}}} \vee \neg E^{(j)}_{P,j_{\mathsf{L}}} \vee \neg E^{(j)}_{\mathsf{ValidCom},j_{\mathsf{R}}} \vee \neg E^{(j)}_{P,j_{\mathsf{R}}}\right)\right] &\leq 2^h \cdot \left(\varepsilon_{j_{\mathsf{L}}}(\lambda) + \varepsilon_{j_{\mathsf{R}}}(\lambda)\right) + \delta(\lambda) \\
&\leq 2^{h+1} \cdot \varepsilon_j(\lambda) + \delta(\lambda),
\end{aligned}$$

where $\delta(\lambda) = 2\mathsf{negl}_1(\lambda) + 4\mathsf{negl}_2(\lambda) = \mathsf{negl}(\lambda)$ and $\varepsilon_j(\lambda) = \max(\varepsilon_{j_{\mathsf{L}}}(\lambda), \varepsilon_{j_{\mathsf{R}}}(\lambda))$. □

Claim A.5. *If P is a tree-based additive invariant and Π_{Com} is computationally binding, then there exists a negligible function $\mathsf{negl}(\cdot)$ such that*

$$\Pr\left[\mathsf{Expt}'_j(\mathcal{A}) = 1 \wedge E^{(j)}_{\mathsf{ValidCom},j_{\mathsf{L}}} \wedge E^{(j)}_{P,j_{\mathsf{L}}} \wedge E^{(j)}_{\mathsf{ValidCom},j_{\mathsf{R}}} \wedge E^{(j)}_{P,j_{\mathsf{R}}}\right] \leq \mathsf{negl}(\lambda).$$

Proof. Suppose

$$\Pr\left[\mathsf{Expt}'_j(\mathcal{A}) = 1 \wedge E^{(j)}_{\mathsf{ValidCom},j_{\mathsf{L}}} \wedge E^{(j)}_{P,j_{\mathsf{L}}} \wedge E^{(j)}_{\mathsf{ValidCom},j_{\mathsf{R}}} \wedge E^{(j)}_{P,j_{\mathsf{R}}}\right] \geq \varepsilon.$$

We use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ for the binding game for Π_{Com} as follows:

1. On input the security parameter 1^λ, algorithm $\mathcal{B}$ runs algorithm $\mathcal{A}$ to obtain the input length 1^n, the set $S \subseteq [n]$, and an index $i^* \in S$.
2. Algorithm $\mathcal{B}$ outputs the block length $1^{\lambda \cdot \ell_{\mathsf{ct}}(\lambda,n)}$ and the vector length $2n-1$ to the challenger. The challenger responds with $\mathsf{crs}_{\mathsf{Com}}$.
3. Algorithm $\mathcal{B}$ computes $(S_0, S_1, \mathsf{idx}) \leftarrow \mathsf{DeriveChal}(S, i^*)$. It then samples the following components:
   - $(\mathsf{sk}_{\mathsf{main}}, \mathsf{pk}_{\mathsf{main}}) \leftarrow \mathsf{HE.Gen}(1^\lambda, 1^n)$ and $(\mathsf{sk}_{\mathsf{shadow}}, \mathsf{pk}_{\mathsf{shadow}}) \leftarrow \mathsf{HE.Gen}(1^\lambda, 1^n)$.

- $(\mathsf{crs}_{\mathsf{BARG}}, \mathsf{vk}_{\mathsf{BARG}}, \mathsf{td}_{\mathsf{BARG}}) \leftarrow \mathsf{BARG.TrapGen}(1^\lambda, 1^{2n-1}, 1^s, 1^3, \{j, j_{\mathrm{L}}, j_{\mathrm{R}}\})$.
- For all $i \in [n]$, sample a random $\mathbf{v}_i \xleftarrow{\mathrm{R}} \{0,1\}^\lambda \setminus \{\mathbf{0}\}$.
- For each $b \in \{\mathsf{main}, \mathsf{shadow}\}$, sample $\mathbf{ct}_{\mathsf{zero}}^{(b)} \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_b, \mathbf{0})$. Then, for each $i \in [n]$ and $b \in \{\mathsf{main}, \mathsf{shadow}\}$, if $i \in S_b$, sample $\mathbf{ct}_i^{(b)} \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_b, \mathbf{v}_i)$; otherwise, if $i \notin S_b$, sample $\mathbf{ct}_i^{(b)} \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_b, \mathbf{0})$.
- For each $b \in \{\mathsf{main}, \mathsf{shadow}\}$, let $\left(\mathsf{com}_{\mathsf{hk}}^{(b)}, \sigma_{\mathsf{hk},1}^{(b)}, \ldots, \sigma_{\mathsf{hk},n}^{(b)}\right) \leftarrow \mathsf{Com.Commit}\left(\mathsf{crs}_{\mathsf{Com}}, (\mathbf{ct}_1^{(b)}, \ldots, \mathbf{ct}_n^{(b)})\right)$.

4. Algorithm $\mathcal{B}$ constructs hk and vk according to Eqs. (4.1) and (4.2):

$$\mathsf{hk} = \left(\mathsf{crs}_{\mathsf{Com}}, \mathsf{crs}_{\mathsf{BARG}}, \left\{\mathsf{pk}_b, \mathbf{ct}_{\mathsf{zero}}^{(b)}, \mathbf{ct}_1^{(b)}, \ldots, \mathbf{ct}_n^{(b)}, \sigma_{\mathsf{hk},1}^{(b)}, \ldots, \sigma_{\mathsf{hk},n}^{(b)}\right\}_{b \in \{\mathsf{main},\mathsf{shadow}\}}\right)$$
$$\mathsf{vk} = \left(\mathsf{crs}_{\mathsf{Com}}, \mathsf{vk}_{\mathsf{BARG}}, \left\{\mathsf{pk}_b, \mathbf{ct}_{\mathsf{zero}}^{(b)}, \mathsf{com}_{\mathsf{hk}}^{(b)}\right\}_{b \in \{\mathsf{main},\mathsf{shadow}\}}\right)$$

   Algorithm $\mathcal{B}$ gives $(\mathsf{hk}, \mathsf{vk})$ to $\mathcal{A}$.

5. Algorithm $\mathcal{A}$ outputs a digest $\mathsf{dig} = \left(\mathbf{ct}_{\mathsf{root}}^{\mathsf{main}}, \mathbf{ct}_{\mathsf{root}}^{\mathsf{shadow}}, \mathsf{com}_{\mathsf{main}}, \mathsf{com}_{\mathsf{shadow}}, \pi_{\mathsf{dig}}\right)$ and a proof π.

6. Algorithm $\mathcal{B}$ computes the following:

   - $(\hat{\mathbf{ct}}_j^{\mathsf{main}}, \hat{\mathbf{ct}}_j^{\mathsf{shadow}}, \sigma_j^{\mathsf{main}}, \sigma_j^{\mathsf{shadow}}, \tilde{w}_j) \leftarrow \mathsf{BARG.Extract}(\mathsf{td}_{\mathsf{BARG}}, \pi, j)$.
   - $(\hat{\mathbf{ct}}_{\mathrm{L}}^{\mathsf{main}}, \hat{\mathbf{ct}}_{\mathrm{L}}^{\mathsf{shadow}}, \sigma_{\mathrm{L}}^{\mathsf{main}}, \sigma_{\mathrm{L}}^{\mathsf{shadow}}, \tilde{w}_{\mathrm{L}}) \leftarrow \mathsf{BARG.Extract}(\mathsf{td}_{\mathsf{BARG}}, \pi, j_{\mathrm{L}})$.
   - $(\hat{\mathbf{ct}}_{\mathrm{R}}^{\mathsf{main}}, \hat{\mathbf{ct}}_{\mathrm{R}}^{\mathsf{shadow}}, \sigma_{\mathrm{R}}^{\mathsf{main}}, \sigma_{\mathrm{R}}^{\mathsf{shadow}}, \tilde{w}_{\mathrm{R}}) \leftarrow \mathsf{BARG.Extract}(\mathsf{td}_{\mathsf{BARG}}, \pi, j_{\mathrm{R}})$.

   In addition, it parses $\tilde{w}_j = (\tilde{w}_{j,\mathrm{L}}, \tilde{w}_{j,\mathrm{R}})$ and the internal witnesses $\tilde{w}_{j,\mathrm{L}} = \left(\hat{\mathbf{ct}}_{j,\mathrm{L}}^{\mathsf{main}}, \hat{\mathbf{ct}}_{j,\mathrm{L}}^{\mathsf{shadow}}, \sigma_{j,\mathrm{L}}^{\mathsf{main}}, \sigma_{j,\mathrm{L}}^{\mathsf{shadow}}\right)$ and $\tilde{w}_{j,\mathrm{R}} = \left(\hat{\mathbf{ct}}_{j,\mathrm{R}}^{\mathsf{main}}, \hat{\mathbf{ct}}_{j,\mathrm{R}}^{\mathsf{shadow}}, \sigma_{j,\mathrm{R}}^{\mathsf{main}}, \sigma_{j,\mathrm{R}}^{\mathsf{shadow}}\right)$.

7. Algorithm $\mathcal{B}$ checks if there exists $b \in \{\mathsf{main}, \mathsf{shadow}\}$ and $d \in \{\mathrm{L}, \mathrm{R}\}$ such that $\hat{\mathbf{ct}}_d^{(b)} \neq \hat{\mathbf{ct}}_{j,d}^{(b)}$ and

$$\mathsf{Com.Verify}\left(\mathsf{crs}_{\mathsf{Com}}, \mathsf{com}_b, j_d, \hat{\mathbf{ct}}_{j,d}^{(b)}, \sigma_{j,d}^{(b)}\right) = 1 \quad \text{and} \quad \mathsf{Com.Verify}\left(\mathsf{crs}_{\mathsf{Com}}, \mathsf{com}_b, j_d, \hat{\mathbf{ct}}_d^{(b)}, \sigma_d^{(b)}\right) = 1.$$

   If so, it outputs the commitment com_b, the index $j_d \in [2n-1]$, and the value-opening pairs $\left(\hat{\mathbf{ct}}_{j,d}^{(b)}, \sigma_{j,d}^{(b)}\right)$ and $\left(\hat{\mathbf{ct}}_d^{(b)}, \sigma_d^{(b)}\right)$. Otherwise, algorithm $\mathcal{B}$ aborts with output $\perp$.

By construction, algorithm $\mathcal{B}$ perfectly simulates an execution of Expt'_j for adversary $\mathcal{A}$. By assumption, with probability at least ε, algorithm $\mathcal{A}$ will output a digest dig and a proof π such that the following conditions hold:

- $\mathsf{Expt}'_j(\mathcal{A}) = 1$: This means $\mathsf{BARG.Verify}(\mathsf{vk}_{\mathsf{BARG}}, C_{\mathsf{idx}}, 2n{-}1, \pi) = 1, C_{\mathsf{idx}}\left(j, (\hat{\mathbf{ct}}_j^{\mathsf{main}}, \hat{\mathbf{ct}}_j^{\mathsf{shadow}}, \sigma_j^{\mathsf{main}}, \sigma_j^{\mathsf{shadow}}, \tilde{w}_j)\right) = 1$, and $P\left(\hat{\mathbf{ct}}_j^{\mathsf{main}}, \hat{\mathbf{ct}}_j^{\mathsf{shadow}}, \mathsf{sk}_{\mathsf{main}}, \mathsf{sk}_{\mathsf{shadow}}, j, (\mathbf{v}_1, \ldots, \mathbf{v}_n, \mathsf{idx})\right) = 0$.
- $E_{\mathsf{ValidCom},j_d}^{(j)}$ for $d \in \{\mathrm{L}, \mathrm{R}\}$: This means

$$\mathsf{Com.Verify}\left(\mathsf{crs}_{\mathsf{Com}}, \mathsf{com}_{\mathsf{main}}, j_d, \hat{\mathbf{ct}}_d^{\mathsf{main}}, \sigma_d^{\mathsf{main}}\right) = 1 = \mathsf{Com.Verify}\left(\mathsf{crs}_{\mathsf{Com}}, \mathsf{com}_{\mathsf{shadow}}, j_d, \hat{\mathbf{ct}}_d^{\mathsf{shadow}}, \sigma_d^{\mathsf{shadow}}\right).$$

- $E_{P,j_d}^{(j)}$ for $d \in \{\mathrm{L}, \mathrm{R}\}$: This means $P\left(\hat{\mathbf{ct}}_d^{\mathsf{main}}, \hat{\mathbf{ct}}_d^{\mathsf{shadow}}, \mathsf{sk}_{\mathsf{main}}, \mathsf{sk}_{\mathsf{shadow}}, j_d, (\mathbf{v}_1, \ldots, \mathbf{v}_n, \mathsf{idx})\right) = 1$.

We consider two possibilities:

- Suppose for all $b \in \{\mathsf{main}, \mathsf{shadow}\}$, we have $\hat{\mathbf{ct}}_{\mathrm{L}}^{(b)} = \hat{\mathbf{ct}}_{j,\mathrm{L}}^{(b)}$ and $\hat{\mathbf{ct}}_{\mathrm{R}}^{(b)} = \hat{\mathbf{ct}}_{j,\mathrm{R}}^{(b)}$. By the third condition, we get

$$P\big(\hat{\mathbf{ct}}_{j,\mathrm{L}}^{\mathsf{main}}, \hat{\mathbf{ct}}_{j,\mathrm{L}}^{\mathsf{shadow}}, \mathsf{sk}_{\mathsf{main}}, \mathsf{sk}_{\mathsf{shadow}}, j_{\mathrm{L}}, (\mathbf{v}_1, \ldots, \mathbf{v}_n, \mathsf{idx})\big) = 1$$
$$P\big(\hat{\mathbf{ct}}_{j,\mathrm{R}}^{\mathsf{main}}, \hat{\mathbf{ct}}_{j,\mathrm{R}}^{\mathsf{shadow}}, \mathsf{sk}_{\mathsf{main}}, \mathsf{sk}_{\mathsf{shadow}}, j_{\mathrm{R}}, (\mathbf{v}_1, \ldots, \mathbf{v}_n, \mathsf{idx})\big) = 1.$$

  By the first condition, we also have $C_{i^*,y}\big(j, (\hat{\mathbf{ct}}_j^{\mathsf{main}}, \hat{\mathbf{ct}}_j^{\mathsf{shadow}}, \sigma_j^{\mathsf{main}}, \sigma_j^{\mathsf{shadow}}, \tilde{w}_j)\big) = 1$, this means that $\hat{\mathbf{ct}}_j^{(b)} = \mathsf{HE.Add}\big(\mathsf{pk}_b, \hat{\mathbf{ct}}_{j,\mathrm{L}}^{(b)}, \hat{\mathbf{ct}}_{j,\mathrm{R}}^{(b)}\big)$ for all $b \in \{\mathsf{main}, \mathsf{shadow}\}$. Since P is a tree-based additive invariant, we get that

$$P\big(\hat{\mathbf{ct}}_j^{\mathsf{main}}, \hat{\mathbf{ct}}_j^{\mathsf{shadow}}, \mathsf{sk}_{\mathsf{main}}, \mathsf{sk}_{\mathsf{shadow}}, j, (\mathbf{v}_1, \ldots, \mathbf{v}_n, \mathsf{idx})\big) = 1.$$

  However, this contradicts the condition that $P\big(\hat{\mathbf{ct}}_j^{\mathsf{main}}, \hat{\mathbf{ct}}_j^{\mathsf{shadow}}, \mathsf{sk}_{\mathsf{main}}, \mathsf{sk}_{\mathsf{shadow}}, j, (\mathbf{v}_1, \ldots, \mathbf{v}_n, \mathsf{idx})\big) = 0$, so this case does not occur.

- Suppose there exists $b \in \{\mathsf{main}, \mathsf{shadow}\}$ and $d \in \{\mathrm{L}, \mathrm{R}\}$ where $\hat{\mathbf{ct}}_d^{(b)} \neq \hat{\mathbf{ct}}_{j,d}^{(b)}$. By the first condition, we have $C_{i^*,y}\big(j, (\hat{\mathbf{ct}}_j^{\mathsf{main}}, \hat{\mathbf{ct}}_j^{\mathsf{shadow}}, \sigma_j^{\mathsf{main}}, \sigma_j^{\mathsf{shadow}}, \tilde{w}_j)\big) = 1$, this means that $\mathsf{Com.Verify}\big(\mathsf{crs}_{\mathsf{Com}}, \mathsf{com}_b, j_d, \hat{\mathbf{ct}}_{j,d}^{(b)}, \sigma_{j,d}^{(b)}\big) = 1$. By the second condition, we also have

$$\mathsf{Com.Verify}\big(\mathsf{crs}_{\mathsf{Com}}, \mathsf{com}_b, j_d, \hat{\mathbf{ct}}_d^{(b)}, \sigma_d^{(b)}\big) = 1.$$

  In this case, algorithm $\mathcal{B}$ outputs the commitment com_b, the index j_d, and the value-opening pairs $\big(\hat{\mathbf{ct}}_{j,d}^{(b)}, \sigma_{j,d}^{(b)}\big)$ and $\big(\hat{\mathbf{ct}}_d^{(b)}, \sigma_d^{(b)}\big)$. This is a pair of valid openings for com_b so algorithm $\mathcal{B}$ wins the binding game.

We conclude that algorithm $\mathcal{B}$ succeeds with the same advantage ε and the claim follows. □

Claim A.6. *Suppose the conditions of Claims A.4 and A.5 hold. Then there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that*

$$\Pr[\mathsf{Expt}'_j(\mathcal{A}) = 1] \leq 2^{h+1} \cdot \varepsilon_j(\lambda) + \mathsf{negl}(\lambda),$$

where $\varepsilon_j(\lambda) = \max(\varepsilon_{j_{\mathrm{L}}}(\lambda), \varepsilon_{j_{\mathrm{R}}}(\lambda))$.

Proof. By the law of total probability, we have

$$\Pr[\mathsf{Expt}'_j(\mathcal{A}) = 1] \leq \Pr\left[\mathsf{Expt}'_j(\mathcal{A}) = 1 \wedge E^{(j)}_{\mathsf{ValidCom},j_{\mathrm{L}}} \wedge E^{(j)}_{P,j_{\mathrm{L}}} \wedge E^{(j)}_{\mathsf{ValidCom},j_{\mathrm{R}}} \wedge E^{(j)}_{P,j_{\mathrm{R}}}\right] +$$
$$\Pr\left[\mathsf{Expt}'_j(\mathcal{A}) = 1 \wedge \left(\neg E^{(j)}_{\mathsf{ValidCom},j_{\mathrm{L}}} \vee \neg E^{(j)}_{P,j_{\mathrm{L}}} \vee \neg E^{(j)}_{\mathsf{ValidCom},j_{\mathrm{R}}} \vee \neg E^{(j)}_{P,j_{\mathrm{R}}}\right)\right].$$

By Claims A.4 and A.5, there exist negligible functions $\mathsf{negl}_1(\cdot)$ and $\mathsf{negl}_2(\cdot)$ such that:

$$\Pr\left[\mathsf{Expt}'_j(\mathcal{A}) = 1 \wedge \left(\neg E^{(j)}_{\mathsf{ValidCom},j_{\mathrm{L}}} \vee \neg E^{(j)}_{P,j_{\mathrm{L}}} \vee \neg E^{(j)}_{\mathsf{ValidCom},j_{\mathrm{R}}} \vee \neg E^{(j)}_{P,j_{\mathrm{R}}}\right)\right] \leq 2^{h+1} \cdot \varepsilon_j(\lambda) + \mathsf{negl}_1(\lambda)$$
$$\Pr\left[\mathsf{Expt}'_j(\mathcal{A}) = 1 \wedge E^{(j)}_{\mathsf{ValidCom},j_{\mathrm{L}}} \wedge E^{(j)}_{P,j_{\mathrm{L}}} \wedge E^{(j)}_{\mathsf{ValidCom},j_{\mathrm{R}}} \wedge E^{(j)}_{P,j_{\mathrm{R}}}\right] \leq \mathsf{negl}_2(\lambda).$$

where $\varepsilon_j(\lambda) = \max(\varepsilon_{j_{\mathrm{L}}}(\lambda), \varepsilon_{j_{\mathrm{R}}}(\lambda))$. The claim follows. □

Completing the proof of Lemma A.1. To complete the proof of the inductive step (for Lemma A.1), we first appeal to Claim A.6 to conclude that there exists negligible function $\mathsf{negl}_1(\cdot)$ such that

$$\Pr[\mathsf{Expt}'_j(\mathcal{A}) = 1] \leq 2^{h+1} \cdot \varepsilon_j(\lambda) + \mathsf{negl}_1(\lambda),$$

where $\varepsilon_j(\lambda) = \max(\varepsilon_{j_{\mathrm{L}}}(\lambda), \varepsilon_{j_{\mathrm{R}}}(\lambda))$. From the inductive hypothesis, $\varepsilon_{j_{\mathrm{L}}}(\lambda)$ and $\varepsilon_{j_{\mathrm{R}}}(\lambda)$ are both negligible functions. By definition of Expt'_j, we have that

$$\Pr[\mathsf{Expt}'_j(\mathcal{A}) = 1] = \Pr\left[E^{(j)}_{\mathsf{Verify}} \wedge E^{(j)}_{\mathsf{CorrectWit},j} \wedge \neg E^{(j)}_{P,j}\right].$$

By Claim A.2, there exists a negligible function $\mathsf{negl}_2(\cdot)$ such that

$$\left|\Pr[\mathsf{Expt}_j(\mathcal{A}) = 1] - \Pr[\mathsf{Expt}'_j(\mathcal{A}) = 1]\right| \leq \mathsf{negl}_2(\lambda).$$

We conclude that

$$\Pr[\mathsf{Expt}_j(\mathcal{A}) = 1] \leq 2^{h+1} \cdot \varepsilon_j(\lambda) + \mathsf{negl}_1(\lambda) + \mathsf{negl}_2(\lambda).$$

Setting $\varepsilon'_j(\lambda) = \max\left(\varepsilon_j(\lambda), (\mathsf{negl}_1(\lambda) + \mathsf{negl}_2(\lambda))/2^{h+1}\right)$, we have that $\Pr[\mathsf{Expt}_j(\mathcal{A}) = 1] \leq 2^{h+1} \cdot \varepsilon'_j(\lambda)$, where $\varepsilon'_j(\lambda)$ is a negligible function. Lemma A.1 now follows by induction on the height h. □

Completing the proof of Theorem 4.12. We now use Lemma A.1 to complete the proof of Theorem 4.12. Suppose the conditions of Theorem 4.12 hold. Noting that the index $2n-1$ has height $h = \log n$ in a complete binary tree with n leaves, we appeal to Lemma A.1 and conclude that there exists a negligible function $\mathsf{negl}(\cdot)$ such that

$$\Pr[\mathsf{Expt}_{2n-1}(\mathcal{A}) = 1] \leq n \cdot \mathsf{negl}(\lambda). \tag{A.7}$$

To complete the proof, we define a sequence of hybrid experiments:

- Hyb_0: This is the experiment $\mathsf{Expt}_{2n-1}[P, \mathsf{DeriveChal}]$ with adversary $\mathcal{A}$.
- Hyb_1: Same as Hyb_0, except the output of the experiment is 1 if the following properties hold:
  - $\mathsf{BARG.Verify}(\mathsf{vk}_{\mathsf{BARG}}, C_{\mathsf{idx}}, 2n-1, \pi) = 1$;
  - $C_{\mathsf{idx}}\big(2n-1, (\hat{\mathbf{ct}}^{\mathsf{main}}_{2n-1}, \hat{\mathbf{ct}}^{\mathsf{shadow}}_{2n-1}, \sigma^{\mathsf{main}}_{2n-1}, \sigma^{\mathsf{shadow}}_{2n-1}, \tilde{w}_{2n-1})\big) = 1$; and
  - $P\big(\mathbf{ct}^{\mathsf{main}}_{\mathsf{root}}, \mathbf{ct}^{\mathsf{shadow}}_{\mathsf{root}}, \mathsf{sk}_{\mathsf{main}}, \mathsf{sk}_{\mathsf{shadow}}, 2n-1, (\mathbf{v}_1, \ldots, \mathbf{v}_n, \mathsf{idx})\big) = 1$.
- Hyb_2: Same as Hyb_1, except the output of the experiment is 1 if the following properties hold:
  - $\mathsf{BARG.Verify}(\mathsf{vk}_{\mathsf{BARG}}, C_{\mathsf{idx}}, 2n-1, \pi) = 1$; and
  - $P\big(\mathbf{ct}^{\mathsf{main}}_{\mathsf{root}}, \mathbf{ct}^{\mathsf{shadow}}_{\mathsf{root}}, \mathsf{sk}_{\mathsf{main}}, \mathsf{sk}_{\mathsf{shadow}}, 2n-1, (\mathbf{v}_1, \ldots, \mathbf{v}_n, \mathsf{idx})\big) = 1$.

  In particular, the challenger no longer checks the value of C_{idx}. Note that in this experiment, the challenger's behavior no longer depends on the BARG trapdoor $\mathsf{td}_{\mathsf{BARG}}$.
- Hyb_3: Same as Hyb_2, except when sampling the BARG parameters at the beginning of the experiment, the challenger now samples $(\mathsf{crs}_{\mathsf{BARG}}, \mathsf{vk}_{\mathsf{BARG}}) \leftarrow \mathsf{BARG.Gen}(1^\lambda, 1^{2n-1}, 1^s, 1^3)$. This corresponds to the experiment $\mathsf{Expt}[P, \mathsf{DeriveChal}]$ with adversary $\mathcal{A}$.

For an adversary $\mathcal{A}$, we write $\mathsf{Hyb}_i(\mathcal{A}) = 1$ to denote the output of Hyb_i with adversary $\mathcal{A}$. We now analyze each pair of adjacent experiments.

Claim A.7. *It holds that* $\Pr[\mathsf{Hyb}_1(\mathcal{A}) = 1] = \Pr[\mathsf{Hyb}_0(\mathcal{A}) = 1]$.

Proof. These experiments are identical. Specifically, by definition of C_{idx} (and specifically, the relation in Fig. 1), if $C_{\mathsf{idx}}\big(2n-1, (\hat{\mathbf{ct}}^{\mathsf{main}}_{2n-1}, \hat{\mathbf{ct}}^{\mathsf{shadow}}_{2n-1}, \sigma^{\mathsf{main}}_{2n-1}, \sigma^{\mathsf{shadow}}_{2n-1}, \tilde{w}_{2n-1})\big) = 1$, then $\hat{\mathbf{ct}}^{(b)}_{2n-1} = \mathbf{ct}^{(b)}_{\mathsf{root}}$ for $b \in \{\mathsf{main}, \mathsf{shadow}\}$. This means that

$$P\big(\hat{\mathbf{ct}}^{\mathsf{main}}_{2n-1}, \hat{\mathbf{ct}}^{\mathsf{shadow}}_{2n-1}, \mathsf{sk}_{\mathsf{main}}, \mathsf{sk}_{\mathsf{shadow}}, 2n-1, (\mathbf{v}_1, \ldots, \mathbf{v}_n, \mathsf{idx})\big) = P\big(\mathbf{ct}^{\mathsf{main}}_{\mathsf{root}}, \mathbf{ct}^{\mathsf{shadow}}_{\mathsf{root}}, \mathsf{sk}_{\mathsf{main}}, \mathsf{sk}_{\mathsf{shadow}}, 2n-1, (\mathbf{v}_1, \ldots, \mathbf{v}_n, \mathsf{idx})\big).$$

Thus, the output of $\mathsf{Hyb}_0(\mathcal{A})$ is identical to that of $\mathsf{Hyb}_1(\mathcal{A})$. □

Claim A.8. *If* Π_{BARG} *is somewhere extractable, then there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that*

$$\left|\Pr[\mathsf{Hyb}_2(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_1(\mathcal{A}) = 1]\right| = \mathsf{negl}(\lambda).$$

Proof. Suppose $\Pr[\mathsf{Hyb}_2(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_1(\mathcal{A}) = 1] = \varepsilon$. Since the only difference between Hyb_1 and Hyb_2 is the conditions the challenger checks at the very end of the experiment, this means that with probability at least ε, the adversary in Hyb_1 will output a digest dig and a proof π such that the following conditions hold:

- $\mathsf{BARG.Verify}(\mathsf{vk}_{\mathsf{BARG}}, C_{\mathsf{idx}}, 2n-1, \pi) = 1$.
- $P\big(\mathbf{ct}_{\mathsf{root}}^{\mathsf{main}}, \mathbf{ct}_{\mathsf{root}}^{\mathsf{shadow}}, \mathsf{sk}_{\mathsf{main}}, \mathsf{sk}_{\mathsf{shadow}}, 2n-1, (\mathbf{v}_1, \ldots, \mathbf{v}_n, \mathsf{idx})\big) = 1$.
- $C_{\mathsf{idx}}\big(2n-1, (\hat{\mathbf{ct}}_{2n-1}^{\mathsf{main}}, \hat{\mathbf{ct}}_{2n-1}^{\mathsf{shadow}}, \sigma_{2n-1}^{\mathsf{main}}, \sigma_{2n-1}^{\mathsf{shadow}}, \tilde{w}_{2n-1})\big) = 0$.

In all other settings, the output of the two experiments are identical. We use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ that for the somewhere extractability game of Π_{BARG} (similar to the proof of Claim A.3):

1. On input the security parameter 1^λ, algorithm $\mathcal{B}$ runs algorithm $\mathcal{A}$ to obtain the input length 1^n, the set $S \subseteq [n]$, and an index $i^* \in S$.
2. Let $j = 2n-1$ and $j_{\mathsf{L}}, j_{\mathsf{R}}$ be the indices of the input wires that determine the value of the output wire j. Algorithm $\mathcal{B}$ outputs $1^{2n-1}, 1^s, 1^3$, the challenge set $J = \{j, j_{\mathsf{R}}, j_{\mathsf{L}}\}$, and the challenge index $j = 2n-1$ to the challenger. Here, s is the bound on the size of the circuit in Fig. 1. The challenger responds with $(\mathsf{crs}_{\mathsf{BARG}}, \mathsf{vk}_{\mathsf{BARG}})$.
3. Algorithm $\mathcal{B}$ computes $(S_0, S_1, \mathsf{idx}) \leftarrow \mathsf{DeriveChal}(S, i^*)$. It then samples the following components:
   - Sample $(\mathsf{sk}_{\mathsf{main}}, \mathsf{pk}_{\mathsf{main}}) \leftarrow \mathsf{HE.Gen}(1^\lambda, 1^n)$ and $(\mathsf{sk}_{\mathsf{shadow}}, \mathsf{pk}_{\mathsf{shadow}}) \leftarrow \mathsf{HE.Gen}(1^\lambda, 1^n)$.
   - Sample $\mathsf{crs}_{\mathsf{Com}} \leftarrow \mathsf{Com.Setup}(1^\lambda, 1^{\lambda \cdot \ell_{\mathsf{ct}}(\lambda,n)}, 2n-1)$.
   - For all $i \in [n]$, sample a random $\mathbf{v}_i \xleftarrow{\mathrm{R}} \{0,1\}^\lambda \setminus \{\mathbf{0}\}$.
   - For each $b \in \{\mathsf{main}, \mathsf{shadow}\}$, sample $\mathbf{ct}_{\mathsf{zero}}^{(b)} \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_b, \mathbf{0})$. Then, for each $i \in [n]$ and $b \in \{\mathsf{main}, \mathsf{shadow}\}$, if $i \in S_b$, sample $\mathbf{ct}_i^{(b)} \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_b, \mathbf{v}_i)$; otherwise, if $i \notin S_b$, sample $\mathbf{ct}_i^{(b)} \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_b, \mathbf{0})$.
   - For each $b \in \{\mathsf{main}, \mathsf{shadow}\}$, let $\big(\mathsf{com}_{\mathsf{hk}}^{(b)}, \sigma_{\mathsf{hk},1}^{(b)}, \ldots, \sigma_{\mathsf{hk},n}^{(b)}\big) \leftarrow \mathsf{Com.Commit}\big(\mathsf{crs}_{\mathsf{Com}}, (\mathbf{ct}_1^{(b)}, \ldots, \mathbf{ct}_n^{(b)})\big)$
4. Algorithm $\mathcal{B}$ constructs hk and vk according to Eqs. (4.1) and (4.2):

$$\mathsf{hk} = \Big(\mathsf{crs}_{\mathsf{Com}}, \mathsf{crs}_{\mathsf{BARG}}, \big\{\mathsf{pk}_b, \mathbf{ct}_{\mathsf{zero}}^{(b)}, \mathbf{ct}_1^{(b)}, \ldots, \mathbf{ct}_n^{(b)}, \sigma_{\mathsf{hk},1}^{(b)}, \ldots, \sigma_{\mathsf{hk},n}^{(b)}\big\}_{b \in \{\mathsf{main},\mathsf{shadow}\}}\Big)$$
$$\mathsf{vk} = \Big(\mathsf{crs}_{\mathsf{Com}}, \mathsf{vk}_{\mathsf{BARG}}, \big\{\mathsf{pk}_b, \mathbf{ct}_{\mathsf{zero}}^{(b)}, \mathsf{com}_{\mathsf{hk}}^{(b)}\big\}_{b \in \{\mathsf{main},\mathsf{shadow}\}}\Big)$$

   Algorithm $\mathcal{B}$ gives $(\mathsf{hk}, \mathsf{vk})$ to $\mathcal{A}$.
5. Algorithm $\mathcal{A}$ outputs a digest $\mathsf{dig} = \big(\mathbf{ct}_{\mathsf{root}}^{\mathsf{main}}, \mathbf{ct}_{\mathsf{root}}^{\mathsf{shadow}}, \mathsf{com}_{\mathsf{main}}, \mathsf{com}_{\mathsf{shadow}}, \pi_{\mathsf{dig}}\big)$ and a proof π.
6. Let C_{idx} be the circuit as defined in Definition 4.10. Algorithm $\mathcal{B}$ outputs the circuit C_{idx}, the instance number $2n-1$, and the proof π.

By definition, the challenger samples $(\mathsf{crs}_{\mathsf{BARG}}, \mathsf{vk}_{\mathsf{BARG}}, \mathsf{td}_{\mathsf{BARG}}) \leftarrow \mathsf{BARG.TrapGen}(1^\lambda, 1^{2n-1}, 1^s, 1^3, \{j, j_{\mathsf{L}}, j_{\mathsf{R}}\})$. This means algorithm $\mathcal{B}$ perfectly simulates an execution of Hyb_1. Thus, with probability at least ε, the digest dig and proof π output by $\mathcal{A}$ satisfies

$$\mathsf{BARG.Verify}(\mathsf{vk}_{\mathsf{BARG}}, C_{\mathsf{idx}}, 2n-1, \pi) = 1 \quad \text{and} \quad C_{\mathsf{idx}}\big(2n-1, (\hat{\mathbf{ct}}_{2n-1}^{\mathsf{main}}, \hat{\mathbf{ct}}_{2n-1}^{\mathsf{shadow}}, \sigma_{2n-1}^{\mathsf{main}}, \sigma_{2n-1}^{\mathsf{shadow}}, \tilde{w}_{2n-1})\big) = 0,$$

where $(\hat{\mathbf{ct}}_{2n-1}^{\mathsf{main}}, \hat{\mathbf{ct}}_{2n-1}^{\mathsf{shadow}}, \sigma_{2n-1}^{\mathsf{main}}, \sigma_{2n-1}^{\mathsf{shadow}}, \tilde{w}_{2n-1}) \leftarrow \mathsf{BARG.Extract}(\mathsf{td}_{\mathsf{BARG}}, \pi, 2n-1)$. This means algorithm $\mathcal{B}$ successfully breaks somewhere extractability of Π_{BARG} and the claim holds. □

Claim A.9. *If Π_{BARG} satisfies set hiding then there exists a negligible function $\mathsf{negl}(\cdot)$ such that*

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

Proof. Suppose $\left|\Pr[\mathsf{Hyb}_3(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_2(\mathcal{A}) = 1]\right| \geq \varepsilon(\lambda)$ for some non-negligible ε. We use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ that breaks set hiding of Π_{BARG}:

1. On input the security parameter 1^λ, algorithm $\mathcal{B}$ runs algorithm $\mathcal{A}$ to obtain the input length 1^n, the set $S \subseteq [n]$, and an index $i^* \in S$.

2. Let $j = 2n-1$ and $j_{\mathsf{L}}, j_{\mathsf{R}}$ be the indices of the input wires that determine the value of the output wire j. Algorithm $\mathcal{B}$ outputs 1^{2n-1}, 1^s, 1^3 and the challenge set $J = \{j, j_{\mathsf{L}}, j_{\mathsf{R}}\}$ to the challenger. Here, s is the bound on the size of the circuit in Fig. 1. The challenger responds with $(\mathsf{crs}_{\mathsf{BARG}}, \mathsf{vk}_{\mathsf{BARG}})$.

3. Algorithm $\mathcal{B}$ computes $(S_0, S_1) \leftarrow \mathsf{DeriveChal}(S, \hat{i})$. It then samples the following components:

   - Sample $(\mathsf{sk}_{\mathsf{main}}, \mathsf{pk}_{\mathsf{main}}) \leftarrow \mathsf{HE.Gen}(1^\lambda, 1^n)$ and $(\mathsf{sk}_{\mathsf{shadow}}, \mathsf{pk}_{\mathsf{shadow}}) \leftarrow \mathsf{HE.Gen}(1^\lambda, 1^n)$.
   - Sample $\mathsf{crs}_{\mathsf{Com}} \leftarrow \mathsf{Com.Setup}(1^\lambda, 1^{\lambda \cdot \ell_{\mathsf{ct}}(\lambda,n)}, 2n-1)$.
   - For all $i \in [n]$, sample a random $\mathbf{v}_i \xleftarrow{\mathrm{R}} \{0,1\}^\lambda \setminus \{\mathbf{0}\}$.
   - For each $b \in \{\mathsf{main}, \mathsf{shadow}\}$, sample $\mathbf{ct}_{\mathsf{zero}}^{(b)} \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_b, \mathbf{0})$. Then, for each $i \in [n]$ and $b \in \{\mathsf{main}, \mathsf{shadow}\}$, if $i \in S_b$, sample $\mathbf{ct}_i^{(b)} \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_b, \mathbf{v}_i)$; otherwise, if $i \notin S_b$, sample $\mathbf{ct}_i^{(b)} \leftarrow \mathsf{HE.Enc}(\mathsf{pk}_b, \mathbf{0})$.
   - For each $b \in \{\mathsf{main}, \mathsf{shadow}\}$, let $\left(\mathsf{com}_{\mathsf{hk}}^{(b)}, \sigma_{\mathsf{hk},1}^{(b)}, \ldots, \sigma_{\mathsf{hk},n}^{(b)}\right) \leftarrow \mathsf{Com.Commit}\left(\mathsf{crs}_{\mathsf{Com}}, (\mathbf{ct}_1^{(b)}, \ldots, \mathbf{ct}_n^{(b)})\right)$

4. Algorithm $\mathcal{B}$ constructs hk and vk according to Eqs. (4.1) and (4.2):

$$\mathsf{hk} = \left(\mathsf{crs}_{\mathsf{Com}}, \mathsf{crs}_{\mathsf{BARG}}, \left\{\mathsf{pk}_b, \mathbf{ct}_{\mathsf{zero}}^{(b)}, \mathbf{ct}_1^{(b)}, \ldots, \mathbf{ct}_n^{(b)}, \sigma_{\mathsf{hk},1}^{(b)}, \ldots, \sigma_{\mathsf{hk},n}^{(b)}\right\}_{b \in \{\mathsf{main},\mathsf{shadow}\}}\right)$$
$$\mathsf{vk} = \left(\mathsf{crs}_{\mathsf{Com}}, \mathsf{vk}_{\mathsf{BARG}}, \left\{\mathsf{pk}_b, \mathbf{ct}_{\mathsf{zero}}^{(b)}, \mathsf{com}_{\mathsf{hk}}^{(b)}\right\}_{b \in \{\mathsf{main},\mathsf{shadow}\}}\right)$$

   Algorithm $\mathcal{B}$ gives $(\mathsf{hk}, \mathsf{vk})$ to $\mathcal{A}$.

5. Algorithm $\mathcal{A}$ outputs a digest $\mathsf{dig} = \left(\mathbf{ct}_{\mathsf{root}}^{\mathsf{main}}, \mathbf{ct}_{\mathsf{root}}^{\mathsf{shadow}}, \mathsf{com}_{\mathsf{main}}, \mathsf{com}_{\mathsf{shadow}}, \pi_{\mathsf{dig}}\right)$ and a proof π.

6. Let $C_{i^*, x_{i^*}}$ be the circuit as defined in Definition 4.10. Algorithm $\mathcal{B}$ outputs 1 if

$$\mathsf{BARG.Verify}(\mathsf{vk}_{\mathsf{BARG}}, C_{\mathsf{idx}}, 2n-1, \pi) = 1 \quad \text{and} \quad P\left(\mathbf{ct}_{\mathsf{root}}^{\mathsf{main}}, \mathbf{ct}_{\mathsf{root}}^{\mathsf{shadow}}, \mathsf{sk}_{\mathsf{main}}, \mathsf{sk}_{\mathsf{shadow}}, 2n-1, (\mathbf{v}_1, \ldots, \mathbf{v}_n, \mathsf{idx})\right) = 1$$

   Otherwise, algorithm $\mathcal{B}$ outputs 0.

We now consider the two possibilities:

- Suppose the challenger responds according to the specification of $\mathsf{ExptSH}_{\mathcal{A}}(\lambda, 0)$. In this case, the challenger samples $(\mathsf{crs}_{\mathsf{BARG}}, \mathsf{vk}_{\mathsf{BARG}}) \leftarrow \mathsf{BARG.Gen}(1^\lambda, 1^{2n-1}, 1^s, 1^3)$. In this case, algorithm $\mathcal{B}$ perfectly simulates an execution of Hyb_3 for $\mathcal{A}$. Moreover, algorithm $\mathcal{B}$ computes the outputs according to the same specification of Hyb_3, so we conclude that algorithm $\mathcal{B}$ outputs 1 with $\Pr[\mathsf{Hyb}_3(\mathcal{A}) = 1]$.

- Suppose the challenger responds according to the specification of $\mathsf{ExptSH}_{\mathcal{A}}(\lambda, 1)$. In this case, the challenger samples $(\mathsf{crs}_{\mathsf{BARG}}, \mathsf{vk}_{\mathsf{BARG}}, \mathsf{td}_{\mathsf{BARG}}) \leftarrow \mathsf{BARG.TrapGen}(1^\lambda, 1^{2n-1}, 1^s, 1^3, \{j, j_{\mathsf{L}}, j_{\mathsf{R}}\})$. In this case, algorithm $\mathcal{B}$ perfectly simulates an execution of Hyb_2 for $\mathcal{A}$, and correspondingly, algorithm $\mathcal{B}$ outputs 1 with probability $\Pr[\mathsf{Hyb}_2(\mathcal{A}) = 1]$.

We conclude that the distinguishing advantage of $\mathcal{B}$ is exactly ε, which concludes the proof. □

Combining Claims A.7 to A.9, we conclude that $\left|\Pr[\mathsf{Hyb}_0(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_3(\mathcal{A}) = 1]\right| = \mathsf{negl}(\lambda)$. By construction, $\mathsf{Hyb}_0(\mathcal{A}) \equiv \mathsf{Expt}_{2n-1}(\mathcal{A})$ and $\mathsf{Hyb}_3(\mathcal{A}) \equiv \mathsf{Expt}(\mathcal{A})$. From Eq. (A.7), we have that $\Pr[\mathsf{Expt}_{2n-1}(\mathcal{A}) = 1] = \mathsf{negl}(\lambda)$ and Theorem 4.12 follows. □

B Puncturable Signatures from Unique Signatures

In this section, we show how to construct puncturable signatures from unique signatures. As shown in [NWW24] (Corollary 1.2), puncturable signatures can be combined with (non-adaptively-sound) monotone-policy BARGs to obtain statically-secure monotone-policy aggregate signatures. The work of [ADM+24] show how to construct puncturable signatures from any simulation-sound non-interactive zero-knowledge proof for NP. This is known from most standard number-theoretic assumptions, including QR [BFM88, Sah99, DDO+01]. Here, we describe another simple approach to constructing puncturable signatures based on a unique signature (or more generally, an invariant signature; see Remark B.10). The construction is a standard application of hard-core predicates.

B.1 Preliminaries Signatures

We first recall the definition of a unique signature.

Definition B.1 (Unique Digital Signatures)**.** A unique digital signature scheme with message space $\mathcal{M}$ is a tuple of efficient algorithms $\Pi_{\mathsf{Sig}} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ with the following syntax:

- $\mathsf{Gen}(1^\lambda) \to (\mathsf{vk}, \mathsf{sk})$: On input the security parameter λ, the key-generation algorithm outputs a key pair $(\mathsf{vk}, \mathsf{sk})$.
- $\mathsf{Sign}(\mathsf{sk}, m) \to \sigma$: On input a signing key sk and a message $m \in \mathcal{M}$, the signing algorithm outputs a signature σ.
- $\mathsf{Verify}(\mathsf{vk}, m, \sigma) \to b$: On input a verification key vk, a message $m \in \mathcal{M}$, and a signature σ, the verification algorithm outputs a bit $b \in \{0, 1\}$.

Moreover, the signature scheme should satisfy the following properties:

- **Correctness:** For all $\lambda \in \mathbb{N}$ and all $m \in \mathcal{M}$, it holds that
$$\Pr\left[\ \mathsf{Verify}(\mathsf{vk}, m, \sigma) = 1 \ \ :\ \ \begin{matrix}(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda) \\ \sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)\end{matrix}\ \right] = 1.$$
- **Unforgeability:** For all efficient and admissible adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that
$$\Pr\left[\mathsf{Verify}(\mathsf{vk}, m^*, \sigma^*) = 1 : \begin{matrix}(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda) \\ (m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk}, \cdot)}(1^\lambda, \mathsf{vk})\end{matrix}\right] = \mathsf{negl}(\lambda),$$
where we say $\mathcal{A}$ is admissible if it does not query the signing oracle $\mathsf{Sign}(\mathsf{sk}, \cdot)$ on the message m^* in the above security game.
- **Uniqueness:** For all $\lambda \in \mathbb{N}$, all $m \in \mathcal{M}$, all $(\mathsf{vk}, \mathsf{sk})$ in the support of $\mathsf{Gen}(1^\lambda)$ and all signatures $\sigma_1, \sigma_2 \in \{0, 1\}^*$, it holds that
$$\mathsf{Verify}(\mathsf{vk}, m, \sigma_1) = \mathsf{Verify}(\mathsf{vk}, m, \sigma_2) = 1 \Rightarrow \sigma_1 = \sigma_2.$$

Puncturable signatures. Next, we recall the definition of puncturable signatures, first introduced by [GVW19] (under the name all-but-one signature).

Definition B.2 (Puncturable Signature [GVW19, adapted])**.** An puncturable (or all-but-one) signature scheme with message space $\mathcal{M}$ is a tuple of efficient algorithms $\Pi_{\mathsf{PunctSig}} = (\mathsf{Gen}, \mathsf{GenPunc}, \mathsf{Sign}, \mathsf{Verify})$ with the following syntax:

- $\mathsf{Gen}(1^\lambda) \to (\mathsf{vk}, \mathsf{sk})$: On input the security parameter λ, the key-generation algorithm outputs a key pair $(\mathsf{vk}, \mathsf{sk})$.
- $\mathsf{GenPunc}(1^\lambda, m^*) \to (\mathsf{vk}, \mathsf{sk})$: On input a security parameter λ and a message $m^* \in \mathcal{M}$, the punctured key generation algorithm outputs a key pair $(\mathsf{vk}, \mathsf{sk})$.
- $\mathsf{Sign}(\mathsf{sk}, m) \to \sigma$: On input a signing key sk and a message $m \in \mathcal{M}$, the signing algorithm outputs a signature σ.

- $\mathsf{Verify}(\mathsf{vk}, m, \sigma) \to b$: On input a verification key vk, a message $m \in \mathcal{M}$, and a signature σ, the verification algorithm outputs a bit $b \in \{0,1\}$.

Moreover, the puncturable signature scheme should satisfy the following properties:

- **Correctness:** For all $\lambda \in \mathbb{N}$ and all $m \in \mathcal{M}$, it holds that

$$\Pr\left[\ \mathsf{Verify}(\mathsf{vk}, m, \sigma) = 1 \ \ : \ \begin{matrix}(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda) \\ \sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)\end{matrix}\ \right] = 1 - \mathsf{negl}(\lambda).$$

  Note that we allow a negligible correctness error.

- **Punctured correctness:** For all $\lambda \in \mathbb{N}$, all $m^* \in \mathcal{M}$, and all $\sigma^* \in \{0,1\}^*$, it holds that

$$\Pr\left[\ \mathsf{Verify}(\mathsf{vk}, m^*, \sigma^*) = 1 \ \ : \ \ (\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{GenPunc}(1^\lambda, m^*)\ \right] = 0.$$

- **Verification key indistinguishability:** For any adversary $\mathcal{A}$ and any $b \in \{0,1\}$, we define the verification key indistinguishability experiment $\mathsf{ExptVKI}_{\mathcal{A}}(\lambda, b)$ as follows:

  1. On input a security parameter λ, the adversary $\mathcal{A}$ outputs a message $m^* \in \mathcal{M}$ and sends it to the challenger.
  2. The challenger samples key pairs $(\mathsf{vk}_0, \mathsf{sk}_0) \leftarrow \mathsf{Gen}(1^\lambda)$ and $(\mathsf{vk}_1, \mathsf{sk}_1) \leftarrow \mathsf{GenPunc}(1^\lambda, m^*)$ and gives vk_b to the adversary.
  3. Next, the adversary can make signing queries on messages $m \in \mathcal{M} \setminus \{m^*\}$. On each signing query, the challenger replies with $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_b, m)$.
  4. The adversary outputs a bit $b' \in \{0,1\}$, which is the output of the experiment.

  We say that Π_{PunctSig} satisfies verification key indistinguishability if for any efficient adversary $\mathcal{A}$ there exists a negligible function $\mathsf{negl}(\cdot)$ such that

$$\left|\Pr[\mathsf{ExptVKI}_{\mathcal{A}}(\lambda, 0) = 1] - \Pr[\mathsf{ExptVKI}_{\mathcal{A}}(\lambda, 1) = 1]\right| = \mathsf{negl}(\lambda).$$

Goldreich-Levin hardcore predicate. Our construction will use the Goldreich-Levin hardcore predicate [GL89]. Specifically, we define a hardcore predicate for a unique signature scheme as follows:

Definition B.3 (Hardcore Predicate for Unique Signature)**.** Let λ be a security parameter. Let $\Pi_{\mathsf{Sig}} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ be a unique signature scheme with signatures of length ℓ. Let $h\colon \{0,1\}^\ell \times \{0,1\}^z \to \{0,1\}$ be a binary function. We say that h is a hardcore predicate for Π_{Sig} if for all efficient and admissible algorithm $\mathcal{A}$ and any message $m^* \in \mathcal{M}$, it holds that

$$\left|\Pr\left[b = h(\sigma^*, \mathbf{r}) : \begin{matrix}\mathbf{r} \xleftarrow{R} \{0,1\}^z \\ (\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda) \\ \sigma^* \leftarrow \mathsf{Sign}(\mathsf{sk}, m^*) \\ b \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk},\cdot)}(1^\lambda, \mathsf{vk}, m^*, \mathbf{r})\end{matrix}\right] - \frac{1}{2}\right| = \mathsf{negl}(\lambda),$$

where we say $\mathcal{A}$ is admissible if it does not query the signing oracle $\mathsf{Sign}(\mathsf{sk}, \cdot)$ on the message m^*.

We can construct a hardcore predicate for a unique signature scheme using the classic Goldreich-Levin construction [GL89, HLR07]. Specifically, we state the theorem below (which can be formally obtained by using the fact that unforgeability for a unique signature implies that the signature σ^* for any message m^* is computationally unpredictable and then invoking [HLR07] with the [GL89] hard-core predicate):

Lemma B.4 (Hardcore Predicate for Unique Signature)**.** *Let $\Pi_{\mathsf{Sig}} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ be a unique signature scheme with signatures of length $\ell := \ell(\lambda)$. Then, the function $h\colon \{0,1\}^\ell \times \{0,1\}^\ell \to \{0,1\}$ defined as $h(\sigma, \mathbf{r}) = \langle \sigma, \mathbf{r} \rangle$ is a hardcore predicate for Π_{Sig}.*

B.2 Puncturable Signature from Unique Signature

Suppose Π_{Sig} is a unique signature scheme with signatures of length ℓ. To construct a puncturable signature from Π_{Sig}, we use the hardcore predicate $h\colon \{0,1\}^\ell \times \{0,1\}^\ell \to \{0,1\}$ associated with Π_{Sig} (Lemma B.4). Our puncturable signature will use λ copies of the unique signature scheme:

- The verification key for the puncturable signature scheme contains λ triples $(\mathsf{vk}_i, \mathbf{r}_i, b_i)$ for $i \in [\lambda]$, where vk_i is a verification key for the unique signature scheme, $\mathbf{r}_i \xleftarrow{R} \{0,1\}^\ell$ is a seed for the hard-core predicate, and $b_i \xleftarrow{R} \{0,1\}$ is a random bit.
- A signature on a message m consists of λ signatures $\sigma_1, \ldots, \sigma_\lambda$ on m with respect to $\mathsf{vk}_1, \ldots, \mathsf{vk}_\lambda$, respectively. The signature is valid if for all $i \in [\lambda]$, σ_i is a valid signature on m with respect to vk_i, and moreover, there exists *some* $j \in [\lambda]$, where $h(\sigma_j, \mathbf{r}_j) \neq b_j$.

Since the bits $b_1, \ldots, b_\lambda \xleftarrow{R} \{0,1\}$ are uniform, for any fixed message m, correctness holds with probability $1 - 1/2^\lambda$, as required. To puncture the verification key at a particular message m^*, we simply set $b_i = h(\sigma_i^*, \mathbf{r}_i)$ where σ_i^* is the (unique) signature on m^* with respect to vk_i. Pseudorandomness of the hard-core bits ensures that this verification key is computationally indistinguishable from the real verification key. Moreover, by construction, there does not exist a signature on m with respect to the punctured key. We now give the formal description:

Construction B.5 (Puncturable Signature)**.** Let $\Pi_{\mathsf{Sig}} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ be a unique digital signature scheme with message space $\mathcal{M}$ and signatures of length $\ell(\lambda)$. We construct a puncturable signature scheme $\Pi_{\mathsf{PunctSig}} = (\mathsf{Gen}', \mathsf{GenPunc}', \mathsf{Sign}', \mathsf{Verify}')$ as follows:

- $\mathsf{Gen}'(1^\lambda)$: On input a security parameter λ, the algorithm samples $(\mathsf{vk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}(1^\lambda)$, $\mathbf{r}_i \xleftarrow{R} \{0,1\}^\ell$, and $b_i \xleftarrow{R} \{0,1\}$ for each $i \in [\lambda]$. The algorithm outputs
$$\mathsf{vk} = \{(i, \mathsf{vk}_i, \mathbf{r}_i, b_i)\}_{i\in[\lambda]} \quad \text{and} \quad \mathsf{sk} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_\lambda).$$
- $\mathsf{GenPunc}'(1^\lambda, m^*)$: On input a security parameter λ and a message $m^* \in \mathcal{M}$, the algorithm samples $(\mathsf{vk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}(1^\lambda)$, $\mathbf{r}_i \xleftarrow{R} \{0,1\}^\ell$, $\sigma_i^* \leftarrow \mathsf{Sign}(\mathsf{sk}_i, m^*)$, and $b_i \leftarrow \langle \sigma_i^*, \mathbf{r}_i \rangle$ for each $i \in [\lambda]$. Then it outputs
$$\mathsf{vk} = \{(i, \mathsf{vk}_i, \mathbf{r}_i, b_i)\}_{i\in[\lambda]} \quad \text{and} \quad \mathsf{sk} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_\lambda).$$
- $\mathsf{Sign}'(\mathsf{sk}, m)$: On input a signing key $\mathsf{sk} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_\lambda)$ and a message $m \in \mathcal{M}$, the signing algorithm computes $\sigma_i \leftarrow \mathsf{Sign}(\mathsf{sk}_i, m)$ for all $i \in [\lambda]$ and outputs $\sigma = (\sigma_1, \ldots, \sigma_\lambda)$.
- $\mathsf{Verify}'(\mathsf{vk}, m, \sigma)$: On input a verification key $\mathsf{vk} = \{(i, \mathsf{vk}_i, \mathbf{r}_i, b_i)\}_{i\in[\lambda]}$, a message $m \in \mathcal{M}$, and a signature $\sigma = (\sigma_1, \ldots, \sigma_\lambda)$, the verification algorithm checks the following:
  1. For all $i \in [\lambda]$, it holds that $\mathsf{Verify}(\mathsf{vk}_i, m, \sigma_i) = 1$.
  2. There *exists* $i \in [\lambda]$ such that $b_i \neq \langle \sigma_i, \mathbf{r}_i \rangle$.

  If both checks pass, then the verification algorithm accepts with output 1. Otherwise, it rejects with output 0.

Theorem B.6 (Correctness)**.** *If Π_{Sig} is correct, then Construction B.5 is correct.*

Proof. Take any security parameter $\lambda \in \mathbb{N}$ and message $m \in \mathcal{M}$. Let $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Gen}'(1^\lambda)$. Then
$$\mathsf{vk} = \{(i, \mathsf{vk}_i, \mathbf{r}_i, b_i)\}_{i\in[\lambda]} \quad \text{and} \quad \mathsf{sk} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_\lambda).$$
Let $\sigma \leftarrow \mathsf{Sign}'(\mathsf{sk}, m)$. By construction, $\sigma = (\sigma_1, \ldots, \sigma_\lambda)$ where $\sigma_i \leftarrow \mathsf{Sign}(\mathsf{sk}_i, m)$. Consider $\mathsf{Verify}'(\mathsf{vk}, m, \sigma)$. By correctness of Π_{Sig}, we have that $\mathsf{Verify}(\mathsf{vk}_i, m, \sigma_i) = 1$ for all $i \in [\lambda]$. Next, since Gen' samples $b_1, \ldots, b_\lambda \xleftarrow{R} \{0,1\}$, with probability $1 - 2^\lambda$, there will exist some index $i \in [\lambda]$ where $b_i \neq \langle \sigma_i, \mathbf{r}_i \rangle$. Thus Construction B.5 satisfies statistical correctness. □

Theorem B.7 (Punctured Correctness)**.** *If* Π_{Sig} *satisfies uniqueness and correctness, then Construction B.5 satisfies punctured correctness.*

Proof. Fix a security parameter $\lambda \in \mathbb{N}$ and a message $m^* \in \mathcal{M}$. Let $(\mathsf{vk}, \mathsf{sk})$ be a key pair in the support of $\mathsf{GenPunc}'(1^\lambda, m^*)$ and parse

$$\mathsf{vk} = \{(i, \mathsf{vk}_i, \mathbf{r}_i, b_i)\}_{i\in[\lambda]} \quad \text{and} \quad \mathsf{sk} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_\lambda).$$

By construction of the punctured key, for each $i \in [\lambda]$ there exists a signature $\sigma_i^* \leftarrow \mathsf{Sign}(\mathsf{sk}_i, m^*)$ such that $b_i = \langle \sigma_i^*, \mathbf{r}_i \rangle$. By correctness of Π_{Sig}, it holds that $\mathsf{Verify}(\mathsf{vk}_i, m^*, \sigma_i^*) = 1$ for each such i. Assume towards a contradiction that there exists $\sigma = (\sigma_1, \ldots, \sigma_\lambda)$ such that $\mathsf{Verify}'(\mathsf{vk}, m^*, \sigma) = 1$. By construction of Verify, there exists $i \in [\lambda]$ such that $b_i \neq \langle \sigma_i, \mathbf{r}_i \rangle$ and $\mathsf{Verify}(\mathsf{vk}_i, m^*, \sigma_i) = 1$. However, by uniqueness of Π_{Sig}, we conclude that $\sigma_i = \sigma_i^*$ which contradicts $b_i = \langle \sigma_i^*, \mathbf{r}_i \rangle \neq \langle \sigma_i, \mathbf{r}_i \rangle$. The claim follows. □

Theorem B.8 (Verification Key Indistinguishability)**.** *If* Π_{Sig} *satisfies unforgeability, then Construction B.5 satisfies verification key indistinguishability.*

Proof. Let $\mathcal{A}$ be an efficient and admissible (non-uniform) adversary for the verification key indistinguishability game. We use a hybrid argument. For each $j \in [0, \lambda]$ we define the experiment Hyb_j as follows:

1. On input a security parameter 1^λ, the adversary $\mathcal{A}$ outputs a message $m^* \in \mathcal{M}$ and sends it to the challenger.
2. For each $i \in [\lambda]$, the challenger samples $(\mathsf{vk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}(1^\lambda)$, and $\mathbf{r}_i \xleftarrow{R} \{0,1\}^\ell$, $\sigma_i^* \leftarrow \mathsf{Sign}(\mathsf{sk}_i, m^*)$. Next, it samples the bit b_i as follows:
   - If $i \leq j$ it samples $b_i \xleftarrow{R} \{0,1\}$.
   - If $i > j$, it sets $b_i = \langle \sigma_i^*, \mathbf{r}_i \rangle$.

   The challenger gives $\mathsf{vk} = \{(i, \mathsf{vk}_i, \mathbf{r}_i, b_i)\}_{i\in[\lambda]}$ to $\mathcal{A}$.
3. Adversary $\mathcal{A}$ can make signing queries on messages $m \in \mathcal{M} \setminus \{m^*\}$. On each signing query, the challenger replies with $\sigma = (\sigma_1, \ldots, \sigma_\lambda)$ where $\sigma_i \leftarrow \mathsf{Sign}(\mathsf{sk}_i, m)$ for all $i \in [\lambda]$.
4. The adversary outputs a bit $b' \in \{0,1\}$, which is the output of the experiment.

Since $\mathcal{A}$ is non-uniform, we assume without loss of generality that the challenge message m^* is fixed for each security parameter λ. We now prove that the advantage of $\mathcal{A}$ in any game will be negligibly close to the adjacent game. Formally:

Lemma B.9. *If* Π_{Sig} *satisfies unforgeability then for all* $j \in [\lambda]$ *it holds that*

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

Proof. Suppose for some index $j \in [\lambda]$ that $|\Pr[\mathsf{Hyb}_j(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{j-1}(\mathcal{A}) = 1]| = \varepsilon$ for some non-negligible ε. Observe that in Hyb_{j-1} if the random bit $b_j \xleftarrow{R} \{0,1\}$ satisfies $b_j = \langle \sigma_j^*, \mathbf{r}_j \rangle$, then the adversary's view in Hyb_{j-1} is identical to the adversary's view in Hyb_{j-1}. This means

$$\Pr[\mathsf{Hyb}_{j-1}(\mathcal{A}) = 1 \mid b_j = \langle \sigma_j^*, \mathbf{r}_j \rangle] = \Pr[\mathsf{Hyb}_j(\mathcal{A}) = 1].$$

This event $b_j = \langle \sigma_j^*, \mathbf{r}_j \rangle$ happens with probability $1/2$, therefore:

$$\Pr[\mathsf{Hyb}_{j-1}(\mathcal{A}) = 1] = \frac{1}{2}\left(\Pr[\mathsf{Hyb}_{j-1}(\mathcal{A}) = 1 \mid b_j = \langle \sigma_j^*, \mathbf{r}_j \rangle] + \Pr[\mathsf{Hyb}_{j-1}(\mathcal{A}) = 1 \mid b_j \neq \langle \sigma_j^*, \mathbf{r}_j \rangle]\right)$$

This means

$$\begin{aligned} \varepsilon &= |\Pr[\mathsf{Hyb}_j(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{j-1}(\mathcal{A}) = 1]| \\ &= \frac{1}{2} \cdot \left|\Pr[\mathsf{Hyb}_{j-1}(\mathcal{A}) = 1 \mid b_j = \langle \sigma_j^*, \mathbf{r}_j \rangle] - \Pr[\mathsf{Hyb}_{j-1}(\mathcal{A}) = 1 \mid b_j \neq \langle \sigma_j^*, \mathbf{r}_j \rangle]\right| \end{aligned} \tag{B.1}$$

We now use $\mathcal{A}$ to construct an algorithm $\mathcal{B}$ for the hardcore predicate game (with message m^*):

1. At the beginning of the game, algorithm $\mathcal{B}$ receives the security parameter 1^λ, the seed $\mathbf{r}_j \xleftarrow{R} \{0,1\}^\ell$, the message m^* and a verification key vk_j from the challenger.

2. For all $i \neq j$, algorithm $\mathcal{B}$ samples $(\mathsf{vk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}(1^\lambda)$, $\mathbf{r}_i \xleftarrow{R} \{0,1\}^\ell$, and $\sigma_i^* \leftarrow \mathsf{Sign}(\mathsf{sk}_i, m^*)$. Then, it constructs b_i as follows:

   - If $i \leq j$ it samples $b_i \xleftarrow{R} \{0,1\}$.
   - If $i > j$, it sets $b_i = \langle \sigma_i^*, \mathbf{r}_i \rangle$.

   Algorithm $\mathcal{B}$ gives $\mathsf{vk} = \{(i, \mathsf{vk}_i, \mathbf{r}_i, b_i)\}_{i \in [\lambda]}$ to $\mathcal{A}$.

3. Whenever algorithm $\mathcal{A}$ makes a signing query on a message $m \in \mathcal{M} \setminus \{m^*\}$, algorithm $\mathcal{B}$ computes $\sigma_i \leftarrow \mathsf{Sign}(\mathsf{sk}_i, m)$ for all $i \neq j$. Algorithm $\mathcal{B}$ then queries the signing oracle on message m to get σ_j. Algorithm $\mathcal{B}$ responds with $\sigma = (\sigma_1, \ldots, \sigma_\lambda)$.

4. At the end of the game, algorithm $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$. If $b' = 1$ then $\mathcal{B}$ outputs b_j. Otherwise $\mathcal{B}$ outputs $1 - b_j$.

By construction, $\mathcal{B}$ perfectly simulates $\mathsf{Hyb}_{j-1}(\mathcal{A})$. If $\mathcal{A}$ is admissible, that is it does not query the signing oracle on the challenge message m^*. This means $\mathcal{B}$ is also admissible. Finally, let $\sigma_j^* \leftarrow \mathsf{Sign}(\mathsf{sk}_j, m^*)$. By construction, algorithm $\mathcal{B}$ outputs the correct value of $h(\sigma^*, \mathbf{r}_j) = \langle \sigma^*, \mathbf{r}_j \rangle$ in the following two cases:

- Algorithm $\mathcal{A}$ outputs $b' = 1$ and $b_j = \langle \mathbf{r}, \sigma_j^* \rangle$.
- Algorithm $\mathcal{A}$ outputs $b' = 0$ and $b_j \neq \langle \mathbf{r}, \sigma_j^* \rangle$.

Therefore $\mathcal{B}$ wins the hardcore predicate game with probability:

$$\begin{aligned}
&\Pr[\mathsf{Hyb}_{j-1}(\mathcal{A}) = 1 \wedge b_j = \langle \sigma_j^*, \mathbf{r}_j \rangle] + \Pr[\mathsf{Hyb}_{j-1}(\mathcal{A}) = 0 \wedge b_j \neq \langle \sigma_j^*, \mathbf{r}_j \rangle] \\
&\quad = \frac{1}{2}\left(\Pr[\mathsf{Hyb}_{j-1}(\mathcal{A}) = 1 \mid b_j = \langle \sigma_j^*, \mathbf{r}_j \rangle] + \Pr[\mathsf{Hyb}_{j-1}(\mathcal{A}) = 0 \mid b_j \neq \langle \sigma_j^*, \mathbf{r}_j \rangle]\right) \\
&\quad = \frac{1}{2}\left(\Pr[\mathsf{Hyb}_{j-1}(\mathcal{A}) = 1 \mid b_j = \langle \sigma_j^*, \mathbf{r}_j \rangle] + 1 - \Pr[\mathsf{Hyb}_{j-1}(\mathcal{A}) = 1 \mid b_j \neq \langle \sigma_j^*, \mathbf{r}_j \rangle]\right) \\
&\quad = \frac{1}{2} + \frac{1}{2}\left(\Pr[\mathsf{Hyb}_{j-1}(\mathcal{A}) = 1 \mid b_j = \langle \sigma_j^*, \mathbf{r}_j \rangle] - \Pr[\mathsf{Hyb}_{j-1}(\mathcal{A}) = 1 \mid b_j \neq \langle \sigma_j^*, \mathbf{r}_j \rangle]\right)
\end{aligned}$$

Taking the absolute difference with $1/2$, we appeal to Eq. (B.1) and conclude that algorithm $\mathcal{B}$ succeeds with advantage

$$\frac{1}{2} \cdot \left|\Pr[\mathsf{Hyb}_{j-1}(\mathcal{A}) = 1 \mid b_j = \langle \sigma_j^*, \mathbf{r}_j \rangle] - \Pr[\mathsf{Hyb}_{j-1}(\mathcal{A}) = 1 \mid b_j \neq \langle \sigma_j^*, \mathbf{r}_j \rangle]\right| = \varepsilon,$$

Thus, algorithm $\mathcal{B}$ breaks security of the hardcore predicate h with the same non-negligible advantage ε. The claim follows. □

By construction, $\mathsf{Hyb}_0(\mathcal{A}) \equiv \mathsf{ExptVKI}_{\mathcal{A}}(\lambda, 0)$ and $\mathsf{Hyb}_\lambda(\mathcal{A}) \equiv \mathsf{ExptVKI}_{\mathcal{A}}(\lambda, 1)$. The proof of Theorem B.8 now follows from a standard hybrid argument. □

Remark B.10 (Invariant Signatures)**.** Although Construction B.5 relies on unique signatures, we can replace the unique signature with an invariant signatures instead [GO92]. In an invariant signature, there can be many signatures for each message, but all such signatures on a particular message share an invariant core (e.g., a common prefix). One way to obtain an invariant signature by composing a pseudorandom function (PRF) with a (simulation-sound) NIZK proof: the verification key contains a commitment to a PRF key and the signature on a message is the PRF evaluation on the message together with a NIZK proof that the PRF value was computed correctly. In this construction, the PRF evaluation on the message is the invariant part of the signature while the NIZK proof (which is randomized) is needed for verification. We can easily adapt Construction B.5 to work with invariant signatures instead of unique signatures by simply taking the hard-core predicate over the invariant core associated with the message rather than the full signature.