Universal SNARGs for NP from Proofs of Correctness

Zhengzhong Jin Northeastern University

Alex Lombardi Princeton University Yael Tauman Kalai MIT

Surya Mathialagan MIT

December 13, 2024

Abstract

We give new constructions of succinct non-interactive arguments (SNARGs) for NP in the settings of both non-adaptive and adaptive soundness.

Our construction of non-adaptive SNARG is *universal* assuming the security of a (leveled or unleveled) fully homomorphic encryption (FHE) scheme as well as a batch argument (BARG) scheme. Specifically, for any choice of parameters ℓ and L, we construct a candidate SNARG scheme for any NP language \mathcal{L} with the following properties:

- the proof length is $\ell \cdot \mathsf{poly}(\lambda)$,
- the common reference string crs has length $L \cdot poly(\lambda)$, and
- the setup is *transparent* (no private randomness).

We prove that this SNARG has non-adaptive soundness assuming the *existence* of any SNARG where the proof size is ℓ , the crs size is L, and there is a size L Extended Frege (\mathcal{EF}) proof of completeness for the SNARG.

Moreover, we can relax the underlying SNARG to be any 2-message privately verifiable argument where the first message is of length L and the second message is of length ℓ . This yields new SNARG constructions based on any " \mathcal{EF} -friendly" designated-verifier SNARG or witness encryption scheme. We emphasize that our SNARG is *universal* in the sense that it does not depend on the argument system.

We show several new implications of this construction that do not reference proof complexity:

- a non-adaptive SNARG for NP with transparent crs from evasive LWE and LWE. This gives a candidate lattice-based SNARG for NP.
- a non-adaptive SNARG for NP with *transparent* crs assuming the (non-explicit) existence of *any* iO and LWE.
- a non-adaptive SNARG for NP with a short and transparent (i.e., uniform) crs assuming LWE, FHE and the (non-explicit) existence of *any* hash function that makes Micali's SNARG construction sound.
- a non-adaptive SNARG for languages such as QR and DCR assuming only LWE.

In the setting of adaptive soundness, we show how to convert any *designated verifier* SNARG into publicly verifiable SNARG, assuming the underlying designated verifier SNARG has an \mathcal{EF} proof of completeness. As a corollary, we construct an adaptive SNARG for UP with a *transparent* crs assuming subexponential LWE and evasive LWE.

We prove our results by extending the encrypt-hash-and-BARG paradigm of [Jin-Kalai-Lombardi-Vaikuntanathan, STOC '24].

Contents

1	Intr	oduction	1
	1.1	Result I: Non-Adaptive Unviersal SNARG for NP	2
	1.2	Result II: Adaptively Sound SNARGs	6
	1.3	Organization	7
2	Our	Techniques	7
	2.1	Prior work	8
	2.2	Our Non-adaptive SNARG and Analysis	10
	2.3	Adaptive Soundness	13
	2.4	Overview of Applications	14
		2.4.1 SNARGs for NP from Evasive LWE	15
		2.4.2 SNARGs for Quadratic Residuousity and (QR) and Nth-Residuousity (DCR)	17
		2.4.3 Transparent Non-Adaptive SNARG and Adaptive SNARG via iO	18
		2.4.4 Transparent adaptive SNARG for UP	19
		2.4.5 Universal Micali SNARG	19
3	Pre	liminaries	21
Ŭ	31	LWE Assumption	21
	3.2	Fully Homomorphic Encryption	22
	3.3	Succinct Non-Interactive Arguments	23
	3.4	Batch Arguments (BARGs)	24
	3.5	Propositional Logic Systems	26
	0.0	3.5.1 Extended Frege	26
		3.5.2 Cook's Theory PV	$\frac{20}{27}$
	3.6	Local Assignment Generators	20
	3.7	Relevant Theorems based on [JKLV24]	$\frac{25}{31}$
4	Uni	versal SNARG Construction	33
_	4.1	Main Theorem Statement	33
	4.2	Proof of Main Theorem	34
5	Con	structions of Adaptively Sound SNARGs	38
	5.1	Adaptively Sound SNARGs from Designated-Verifier SNARGs	38
	5.2	Proof of Adaptive Soundness	41
6	App	Dication I: Non-Adaptive SNARGs from Witness Encryption	44
	6.1	Witness Encryption	45
	6.2	Main theorem statement (WE)	45
	6.3	SNARG for NP from Evasive LWE	46
		6.3.1 PV Proofs for Properties of Linear Algebra	46
		6.3.2 Evasive LWE	48
		6.3.3 Matrix Branching Programs	49
		6.3.4 Matrix Branching Program Encoding of CNF	49
		6.3.5 Trapdoor and Pre-image Sampling	50

		6.3.6 GGH Encodings	50
		6.3.7 σ -PRF Obfuscation	51
	C 4	6.3.8 Witness Encryption from Evasive LWE with PV Proof	54
	0.4	SNARGS for QR and DCR from LWE.	Э <i>1</i>
7	App	plication II: Transparent and Adaptive SNARGs from iO and LWE	60
	7.1	Indistinguishability Obfuscation for Circuits	61
	7.2	Single-Key Functional Encryption	61
	7.3	Upgrading iO to have a PV proof of correctness	63
		7.3.1 Slow XiO with PV proof of correctness	63
		7.3.2 Slow XiO to Fast XiO with CRS	64
		7.3.3 Single-Key Compact FE for Bounded-Depth Circuits	66
		7.3.4 Single-Key Compact FE for all circuits	67
		7.3.5 Output-Compressing Randomized Encodings for Turing Machines	69 71
	74	1.3.6 IO from Output-Compressing RE	71
	1.4 75	Adaptive SNAPCs for NP from iO and LWE	(1 70
	1.5		12
8	App	plication III: Adaptive Transparent SNARGs for UP from Evasive LWE	73
9	App	plication IV: Universal Micali SNARGs	74
	9.1	Probabilistically Checkable Proofs	75
			10
	9.2	Merkle Tree Hash	76
	$\begin{array}{c} 9.2\\ 9.3\end{array}$	Merkle Tree Hash PV Proof of Completeness for Merkle Hash	76 77
10	9.2 9.3 Ack	Merkle Tree Hash	76 77 80
10 A	9.2 9.3 Ack	Merkle Tree Hash	76778088
10 A	9.2 9.3 Ack Def A.1	Merkle Tree Hash	 76 77 80 88 88 88
10 A	 9.2 9.3 Ack Def A.1 A.2 	Merkle Tree Hash PV Proof of Completeness for Merkle Hash cnowledgements ferred proofs and constructions from Section 6.3.2 Proof of Lemma 6.10 Read- $c \sigma$ -PRFs	 76 77 80 88 88 90
10 A	 9.2 9.3 Ack Def A.1 A.2 A.3 	Merkle Tree Hash PV Proof of Completeness for Merkle Hash cnowledgements ferred proofs and constructions from Section 6.3.2 Proof of Lemma 6.10 Read- $c \sigma$ -PRFs Modified Designated-Verifier SNARG for UP based on [MPV24]	 76 77 80 88 88 90 92
10 A B	9.2 9.3 Ack Def A.1 A.2 A.3 <i>PV</i>	Merkle Tree Hash PV Proof of Completeness for Merkle Hash PV Proof of Completeness for Merkle Hash cnowledgements $extremath{ferred proofs and constructions from Section 6.3.2$ Proof of Lemma 6.10 $extremath{Read}$ Read- $c \sigma$ -PRFs $extremath{Read}$ Modified Designated-Verifier SNARG for UP based on [MPV24] $extremath{Read}$ Proof for Leveled GSW Fully Homomorphic Encryption $extremath{Read}$	 76 77 80 88 88 90 92 96
10 A B C	9.2 9.3 Ack Def A.1 A.2 A.3 PV PV	Merkle Tree Hash PV Proof of Completeness for Merkle Hash PV Proof of Completeness for Merkle Hash cnowledgements C C Ferred proofs and constructions from Section 6.3.2 C Proof of Lemma 6.10 C Read- c σ -PRFs Modified Designated-Verifier SNARG for UP based on [MPV24] C Proof for Leveled GSW Fully Homomorphic Encryption Proof for Succinct Functional Encryption	 76 77 80 88 88 90 92 96 99
10 A B C	9.2 9.3 Ack Def A.1 A.2 A.3 <i>PV</i> <i>PV</i> C.1	Merkle Tree Hash PV Proof of Completeness for Merkle Hash PV Proof of Completeness for Merkle Hash cnowledgements $extremath{ferred proofs and constructions from Section 6.3.2$ Proof of Lemma 6.10 $extremath{Read}$ Nead- $c \sigma$ -PRFs $extremath{Nextremath{\mathsf{restructions from Section 6.3.2}}$ Modified Designated-Verifier SNARG for UP based on [MPV24] $extremath{Nextremath{\mathsf{restructions from Section 6.3.2}}$ Proof for Leveled GSW Fully Homomorphic Encryption $extremath{Proof for Succinct Functional Encryption}$ Proof for Garbled circuits $extremath{Nextremath{\mathsf{restructions from Section 6.3.2}}$	 76 77 80 88 88 90 92 96 99 99
10 A B C	9.2 9.3 Ack Def A.1 A.2 A.3 <i>PV</i> <i>PV</i> C.1	Merkle Tree Hash PV Proof of Completeness for Merkle Hash PV Proof of Completeness for Merkle Hash cnowledgements ferred proofs and constructions from Section 6.3.2 Proof of Lemma 6.10 Read- $c \sigma$ -PRFs Modified Designated-Verifier SNARG for UP based on [MPV24] Proof for Leveled GSW Fully Homomorphic Encryption Proof for Succinct Functional Encryption PV Proof for Garbled circuits C.1.1 Construction and PV proof	 76 76 77 80 88 88 90 92 96 99 99 100
10 A B C	9.2 9.3 Ack Def A.1 A.2 A.3 PV PV C.1 C.2	Merkle Tree Hash PV Proof of Completeness for Merkle Hash PV Proof of Completeness for Merkle Hash $extrema completeness for Merkle Hash cnowledgements extrema completeness for Merkle Hash ferred proofs and constructions from Section 6.3.2 extrema completeness for Merkle Hash Proof of Lemma 6.10 extrema completeness for Merkle Hash Read-c \sigma-PRFs extrema completeness for Merkle Hash Modified Designated-Verifier SNARG for UP based on [MPV24] extrema completeness for Merkle Me$	 76 76 77 80 88 88 90 92 96 99 99 100 103

1 Introduction

Succinct non-interactive arguments (SNARGs) are a cryptographic primitive in which a polynomialtime prover, given a witness w for an NP statement x along with a common reference string crs, generates a short, efficiently verifiable proof π that x is true. Since their introduction by Micali [Mic94], SNARGs have proved themselves to be a powerful, versatile tool both in theory and in practice. Despite their importance, constructing SNARGs, and proving their security, has remained a challenging task over the last three decades. In this work, we tackle this problem head-on and make progress on the following questions.

Do there exist SNARGs for all NP languages? Under what computational assumptions can we prove this? Which useful properties of SNARGs can we provably guarantee?

To describe the state-of-the-art, prior known SNARG constructions can be roughly broken down into the following three categories.

SNARGs based on unfalsifiable assumptions. SNARGs were first defined and constructed by Micali [Mic94], who constructed a SNARG and proved its soundness in the Random Oracle Model [BR94]. Starting with [BCS16], a line of research has attempted to optimize SNARGs in the random oracle model, mostly for practical efficiency. There are also many SNARG (or SNARK) constructions in the literature that are based on unfalsifiable "knowledge assumptions," which *assume* the existence of a knowledge extractor for every efficient adversary. These include assumptions such as knowledge-of-exponent [Gro10, Gro16], extractable collision resistant hash functions [BCC⁺17], linear-only encryption schemes [BCI⁺13], and more [DCL08].

A key drawback of all of these constructions is that their security is not proved based on a *falsifiable assumption* [Nao03], and thus fail to achieve the gold standard "win-win" security guarantee. Indeed, an impossibility result of Gentry and Wichs [GW11] demonstrates a serious barrier to achieving SNARGs for NP with *adaptive soundness* (where the cheating prover is allowed to choose the NP statement x as a function of the crs) based on falsifiable assumptions.

SNARGs based on iO. Despite the Gentry-Wichs barrier, there are several SNARG constructions for all of NP based on indistinguishability obfuscation (iO). This includes the work of Sahai and Waters that constructs a SNARG with non-adaptive soundness [SW14] and three recent works that construct SNARGs with adaptive soundness [WW24a, WZ24, WW24b]. These can either be based on heuristic iO constructions, or can be proved sound under the 2^n -security of certain falsifiable assumptions [BV15, AJ15, JLS21].¹

While the 2^n -security assumption is justifiable due to the Gentry-Wichs barrier, the iO-based constructions satisfy several other drawbacks:

- iO is currently only known assuming a combination of three specific hardness assumptions [JLS21, JLS22],
- In particular, there is no known iO scheme based on post-quantum falsifiable assumptions,

¹This use of complexity leveraging increases the length of the crs but does not affect the length of π in the above SNARG constructions.

• These SNARGs have a long, *structured* crs, whose size is proportional to that of the verification circuit of the underlying NP language.

SNARGs for restricted languages. The majority of theoretical SNARG constructions in the literature have restricted their scope to *sub-classes* of NP languages rather than working with NP-complete languages [KR09, KRR14, KP16, BHK17, BKK⁺18, CCH⁺19, KPY19, JKKZ21, CJJ21, CJJ22, WW22, DGKV22, PP22, JJ22, KLVW23, KLV23, CGJ⁺23, BBK⁺23, JKLV24]. These SNARGs all have soundness proofs under falsifiable assumptions such as learning with errors (LWE). The set of NP languages captured by this line of work is different in the cases of adaptive and non-adaptive soundness:

- Adaptively sound SNARGs are known for all NP languages that have (adaptive) non-signaling PCPs of low locality [KRR14, BHK17, CJJ22, KVZ21]. Such languages are known to be decidable in deterministic sub-exponential time [KRR14], which is consistent with the Gentry-Wichs barrier.
- Non-adaptively sound SNARGs are known for all NP languages that have a "co-nondeterministic variant" of non-signaling PCP of low locality [JKLV24]. Such languages are known to be decidable in *co-nondeterministic* sub-exponential time, and thus are likely to form a strict subclass of NP.

In this work, we apply the techniques from this third category of results – SNARGs for restricted languages – to help construct and analyze new candidate SNARGs for *all of* NP.

1.1 Result I: Non-Adaptive Unviersal SNARG for NP

We first describe our results for SNARGs with non-adaptive soundness. In this setting, we give a universal SNARG construction based on the encrypt-hash-and-BARG paradigm [JKLV24], which in turn can be constructed from the following cryptographic building blocks:

- a non-interactive batch argument system (BARG) for NP [CJJ22].
- a fully² homomorphic encryption scheme [Gen09, BV11].

Moreover, this SNARG can be instantiated with a *transparent setup*: the crs can be taken to be a uniformly random string along with a (public-coin) hash of that string.³ In particular, this means that one does not need a *trusted party* to set up the crs. The need for such a trusted setup has been a major bottleneck for using SNARGs in practice.

We prove the following theorem on the soundness of this SNARG.

Theorem 1.1 (Informal). For any NP language \mathcal{L} , our SNARG is non-adaptively sound assuming the security of the above building blocks as well as assuming the **existence** of any two-message

²There are versions of this construction using either leveled or unleveled FHE; the construction based on unleveled FHE has a shorter common reference string. We defer a more detailed discussion to the body of the paper.

 $^{^{3}}$ This requires that the FHE has pseudorandom ciphertexts and that the BARG has pseudorandom public parameters.

laconic⁴ argument system $(\mathcal{P}, \mathcal{V})$ for \mathcal{L} , provided that the completeness of $(\mathcal{P}, \mathcal{V})$ can be proved via a polynomial-length Extended Frege (\mathcal{EF}) propositional proof.

In particular, Theorem 1.1 reduces (modulo, e.g., LWE) the long-standing problem of constructing SNARGs for NP to the seemingly much easier problem of building 2-message laconic arguments! Moreover, this soundness reduction holds "instance-wise:" our SNARG is non-adaptively sound when restricted to any subset of instances for which such laconic arguments exist.

What is a proof of completeness? Let us specify more carefully what we mean by an \mathcal{EF} proof of completeness in the statement of Theorem 1.1. In the argument system $(\mathcal{P}, \mathcal{V})$, the prover and verifier may make use of random coins $(r_{\mathcal{P}}, r_{\mathcal{V}})$. The requirement is that for all strings x and all random coins $r_{\mathcal{V}}$, there is a polynomial-length \mathcal{EF} proof of the following claim:

For all $w \in \mathcal{R}_x^{\mathcal{L}}$, and all $r_{\mathcal{P}}$, if $\mathsf{ch} = \mathcal{V}(x; r_{\mathcal{V}})$ and $\pi = \mathcal{P}(x, w, \mathsf{ch}; r_{\mathcal{P}})$, then $\mathcal{V}(x, \pi; r_{\mathcal{V}}) = \mathsf{accept.}^5$

We elaborate on the definition of an \mathcal{EF} proof system in Section 3.5.

 \mathcal{EF} proofs of completeness should be "for free." Theorem 1.1 begs the following question: should we expect a laconic proof system to have a polynomial-length \mathcal{EF} proof of completeness? We first note that it must have *unconditional* completeness. That is, if a cryptographic construction only has completeness under a hardness (e.g. derandomization) assumption,⁶ we cannot hope to prove that it has an \mathcal{EF} proof of completeness (without proving the assumption to be true).

We believe that subject to this condition, most cryptographic constructions in the literature that have perfect correctness should be instantiable with \mathcal{EF} proofs of correctness. The reason for this is that the correctness of a cryptosystem – when all users behave honestly – typically follows from basic properties of arithmetic. Indeed, proofs of honest-user correctness that appear in papers often simply say that it "follows immediately from the definitions." Following a line of research including [Par71, Coo75, Bus86], it should be possible to formulate such "simple" proofs in bounded arithmetic, which yields polynomial-length \mathcal{EF} proofs by Cook's correspondence.

However, to derive formal corollaries, this meta-principle must be verified on a case-by-case basis. In this work, we show that Gentry-Sahai-Waters FHE scheme (Appendix B), Yao's garbled circuits (Appendix C.1.1), Merkle trees (Section 9.3) and many other cryptographic constructions (see [JJ22] for more examples) in fact have \mathcal{EF} proofs of correctness. For the special case of iO, we are also able to prove the following theorem (see Section 7), stating that \mathcal{EF} proofs of correctness can be generically added to any iO scheme:

Theorem 1.2. Assuming any subexponentially-secure iO scheme and subexponential LWE, there exists an iO scheme with \mathcal{EF} proofs of correctness.⁷

⁴A laconic argument system is an argument system in which the prover-to-verifier communication is small; however, the verifier's message may be long, and may depend on the NP statement x.

⁵Indeed, this is implied by a similar requirement where the quantifiers over x and $r_{\mathcal{V}}$ are inside the claim, but this weaker requirement suffices for our result.

⁶One example of a scheme that relies on a hardness assumption to argue completeness is the witness encryption construction of [CVW18, VWW22] from evasive LWE. Looking forward, we in fact have to modify this construction to obtain perfect completeness for our corollary of SNARG for NP from evasive LWE.

⁷In fact, we show a stronger claim that this scheme has a PV proof of correctness. This implies that the scheme has \mathcal{EF} proofs of correctness that can be uniformly generated.

Universality of our SNARG. One immediate corollary of Theorem 1.1 is an explicit construction of SNARGs for NP that is *universal* with respect to SNARGs with \mathcal{EF} proofs of completeness: if a SNARG for NP with an \mathcal{EF} proof of completeness *exists*, then our (transparent CRS) construction is secure.⁸

Due to this universality result, we view our specific candidate SNARG as a "best possible" SNARG (somewhat akin to [GR07]) among all SNARGs that have \mathcal{EF} proof of correctness (assuming FHE and BARG). For example, as we will see in Corollary 1.7, the mere existence of a post-quantum iO scheme is sufficient for our SNARG to be post-quantum secure (assuming LWE is post-quantum secure).

Moreover, our SNARG is non-adaptively sound if there exists any *privately verifiable* SNARG, and moreover, even if there exists a two-message laconic argument systems, which is potentially a far weaker object than (even privately verifiable) SNARGs for NP. Thus, our construction can be seen as a transformation boosting any of these weaker proof systems into a full-fledged transparent SNARG for NP. In more concrete terms, an adversary breaking the non-adaptive soundness of our SNARG effectively proves that a 2-message laconic argument system for NP (with poly-size \mathcal{EF} proofs of completeness and the appropriate size parameters) does not exist!

As we will see, this universality result turns out to have powerful implications.

Implication: SNARGs from Witness Encryption. Note that two-message laconic arguments for NP are immediately implied by any *witness encryption scheme* [GGSW13]; moreover, completeness of the argument system has an \mathcal{EF} proof if the underlying witness encryption has an \mathcal{EF} proof of *honest decryption correctness*. As a result, we obtain the following corollary.

Corollary 1.3 (Informal). Our SNARGs for NP is sound assuming the hardness of LWE as well as the existence of a witness encryption scheme for NP whose correctness property

 $\forall w, R(x, w) = 1 \rightarrow \mathsf{Dec}(w, \mathsf{Enc}(x, m)) = m$

has a family of polynomial-length \mathcal{EF} proofs.

Moreover, if the witness encryption is succinct (does not grow with the instance or witness length) and if there is a uniform machine that prints the \mathcal{EF} proof, then we can get a SNARG for NP with a short, uniformly random crs.

We can concretely instantiate such a witness encryption scheme based on the evasive LWE assumption, which has received significant attention recently [Wee22, Tsa22, VWW22, ARYY23, HLL23, MPV24]. Although the lattice-based witness encryption schemes of [CVW18, Tsa22, VWW22] only guarantee computational correctness (and hence do not have \mathcal{EF} proofs), we show that a modified construction (following [MPV24]) does have perfect (and \mathcal{EF} -provable) correctness. As a result, we obtain the following corollary.

Corollary 1.4 (Informal). Our SNARGs for NP is sound assuming the sub-exponential hardness of LWE as well as evasive LWE. The SNARG has transparent setup and a crs of length $poly(|x|, |w|, \lambda)$.

⁸We note that compared to Levin-style universal constructions of cryptographic primitives which enumerate over all possible constructions, our construction does not suffer from "galactic inefficiency;" moreover, our construction can also make use of the existence of SNARGs with *non-uniform* honest provers.

Since LWE and evasive LWE are both plausibly post-quantum secure, this yields a candidate post-quantum SNARG for NP. We note that SNARGs from evasive LWE, even in the designated verifier setting, were previously only known for a subclass of NP known as UP⁹ [MPV24]. This demonstrates the power of our relaxation from SNARGs to 2-message laconic arguments. While evasive LWE is not a falsifiable assumption, we hope that this relaxation will provide new insights towards constructions of SNARGs for NP from standard assumptions.

Finally, by using unconditional, statistical witness encryption schemes for explicit languages such as QR and $\overline{\text{DCR}}^{10}$ (with \mathcal{EF} proofs of correctness), we can obtain SNARGs for such languages assuming only LWE.

Corollary 1.5 (informal). Assuming the hardness of LWE, our SNARG is sound for the QR and $\overline{\text{DCR}}$ languages.

It was not previously known how to obtain SNARGs for these languages without iO or heuristics. In particular, the prior work [JKLV24] builds SNARGs for NP languages with \mathcal{EF} proofs of *unsatisfiability*, which are not known for these languages.

Implication: Transparent SNARGs for NP from iO. Combining Theorem 1.1 with the existing construction of SNARG for NP from iO [SW14], we have the following corollary:

Corollary 1.6. Our SNARG for NP is sound assuming the (non-explicit) existence of an iO with \mathcal{EF} proofs of correctness and LWE.

Combining this corollary with Theorem 1.2, which states that subexponential iO and subexponential LWE can be used to construct an iO scheme with \mathcal{EF} proofs of correctness, we can instead state the corollary as follows:

Corollary 1.7. Our transparent SNARG for NP is sound assuming the (non-explicit) existence of subexponential iO as well as subexponential LWE.

This is the first SNARG for NP with transparent setup that is provably sound under standard cryptographic assumptions. We emphasize that all previous SNARG constructions based on iO [SW14, WW24a, WZ24, WW24b] do not have transparent setup.

Implication: Universal Micali SNARGs. In addition to our evasive LWE-based instantiation, we show that Micali's protocol, with any hash function instantiation, has a poly-size \mathcal{EF} proof of completeness, as long as the underlying PCP has a poly-size \mathcal{EF} proof of completeness. By the work of [Pic15], the PCP construction of Dinur [Din07] in fact has an \mathcal{EF} proof of completeness. Thus, we obtain the following corollary.

Corollary 1.8 (Informal). Our SNARG for NP is sound with short and transparent crs assuming LWE, (unleveled) FHE and and the (non-explicit) existence of any hash function that makes Micali's SNARG construction non-adaptively sound (with the PCP from [Din07]).

⁹UP is a subclass of NP where every instance x has at most one witness.

 $^{^{10}\}overline{\text{DCR}}$ denotes the complement of the DCR promise language.

1.2 Result II: Adaptively Sound SNARGs

In the adaptive setting, we give the first generic transformation from *privately*-verifiable to *publicly*-verifiable SNARGs.

Theorem 1.9 (informal). Assume the existence of a privately verifiable adaptively sound SNARG for some language \mathcal{L} with an "instance-universal"¹¹ \mathcal{EF} proof of completeness. Then, assuming the existence of (leveled) FHE and BARGs for NP, there is a publicly verifiable adaptively sound SNARG for \mathcal{L} .

Moreover, if the BARG public parameters are pseudorandom, the FHE ciphertexts are pseudorandom, and the proving key of the privately verifiable SNARG is 2^n -indistinguishable from uniform, the publicly verifiable SNARG can be made to have transparent setup.

We emphasize that Theorem 1.9 also applies to *preprocessing* SNARGs, whose proving keys may be long and/or take a long time to generate. While the construction in Theorem 1.9 is not universal (for reasons that will become in our technical overview in Section 2), it does not depend on the \mathcal{EF} proof (only on its length).

Unlike our non-adaptive universal SNARG construction, the transformation in Theorem 1.9 does not generically give a transparent set-up. However, we obtain an adaptively sound SNARG with transparent set-up by plugging in a slight variant of the adaptively sound, privately verifiable SNARG for UP of [MPV24].

Corollary 1.10. Assuming the sub-exponential hardness of LWE and evasive LWE, there are transparent adaptively sound SNARGs for UP.

To the best of our knowledge, this is the only *transparent* adaptively sound SNARG candidate proven sound under an explicit assumption, aside from [Mic94] and its close relatives [BCS16] in the random oracle model. Moreover, it gives a plausibly post-quantum adaptively sound SNARG for UP. This is in contrast to the knowledge assumption underlying prior lattice-based (privately verifiable) candidates, whose underlying hardness assumption was recently broken [DFS24].

Implication: Adaptive SNARGs via Witness PRFs. More generally, [MPV24] showed that a subexponentially sound witness PRF (a primitive introduced by [Zha16]) for a language L can be used to construct adaptively sound designated-verifier SNARGs for L. A witness PRF, in short, can be seen as a generalization of witness encryption. Coupled with our result (Theorem 1.9), we have the following corollary.

Corollary 1.11 ((Informal)). Assuming LWE and a witness PRF with an \mathcal{EF} proof of completeness for a language L, there exists an adaptively sound SNARG for L.

In fact, [MPV24] show that the privately verifiable variant of the [SW14] SNARG based on sub-exponential iO and one-way functions gives such a witness PRF. Therefore, combined with Theorem 1.2, we have the following corollary.

Corollary 1.12. Assuming subexponential iO and subexponential LWE, there exists an adaptively sound SNARG for NP.

While this corollary is not new (a sequence of works [WW24a, WZ24, WW24b] construct such a SNARG from iO and OWFs), this gives an alternative pathway to obtaining adaptively sound publicly verifiable SNARGs for NP.

¹¹As discussed previously, we include the quantification over all statements x in the \mathcal{EF} tautology.

Comparison with [JJ22, JKLV24]. The idea of using \mathcal{EF} proofs in the analysis of SNARG schemes originated in [JJ22] and was recently used in [JKLV24]. In [JJ22], \mathcal{EF} proofs were used to obtain a non-adaptive SNARGs from iO with a reduction that does not run in time 2^n , and in fact is independent of the input length n. More recently, [JKLV24] constructed SNARGs via the *encrypt-hash-and*-BARG paradigm (which we elaborate on in Section 2).

However, the SNARGs of both [JJ22, JKLV24] are only proved sound for NP languages that have polynomial-size [JJ22] (or more generally, polynomial space [JKLV24]) \mathcal{EF} proofs of nonmembership. That is, for a fixed NP language \mathcal{L} with NP verification circuit C, these works required efficient \mathcal{EF} proofs that $C(x, \cdot)$ is unsatisfiable for each fixed $x \notin \mathcal{L}$. This places a complexitytheoretic constraint on the language \mathcal{L} , since such languages are decidable in co-nondeterministic sub-exponential time [JKLV24].

In contrast, we use a variant of the [JKLV24] encrypt-hash-and-BARG construction, but rather than requiring an \mathcal{EF} proof about *language non-membership*, we make use of an \mathcal{EF} proof of *correctness of a cryptographic primitive* that we introduce in the security analysis. Interestingly, the completeness property of a 2-message argument is required to hold for *true* statements $x \in \mathcal{L}$ rather than false $x \notin \mathcal{L}$, but we use these \mathcal{EF} proofs in our soundness analysis! Thus, no complexitytheoretic constraint is placed on the language and we obtain candidate SNARGs for NP.

1.3 Organization

In Section 2, we give a high level overview of our techniques used to achieve our main results: Theorem 1.1, Theorem 1.9, and their applications. In Section 3, we recall basic definitions and tools from prior work that we use throughout the paper. We present our universal SNARG construction and prove Theorem 1.1 in Section 4. We then present our transformation from adaptively sound privately-verifiable SNARGs to publicly verifiable SNARGs and prove Theorem 1.9 in Section 5. In Section 6, we show two constructions of SNARGs for NP via witness encryption: one from evasive LWE and LWE, and another for languages such as QR and DCR from just LWE. In Section 7, we first show that any subexponential iO scheme can be transformed into an iO scheme with a *PV* proof of correctness by assuming subexponential LWE (Theorem 1.2). As corollaries, we show transparent SNARG for NP as well as an adaptively sound (non-transparent) SNARG for NP. In Section 8, we construct an adaptive SNARG for UP with *transparent* crs. Finally, in Section 9, we construct a universal SNARG for NP with non-adaptive soundness, assuming the (non-explicit) existence of a hash function that can securely instantiate Micali's SNARG, and assuming LWE and FHE, thus proving Corollary 1.8. We defer many details to the appendix.

2 Our Techniques

In this section, we describe our main techniques. We recap prior work in Section 2.1. In Section 2.2, we give an overview of the security proof of our universal non-adaptive SNARG (Theorem 1.1). In Section 2.3, we discuss extending our approach to adaptive SNARGs (Theorem 1.9). Finally, in Section 2.4, we discuss how to obtain the various applications of Theorem 1.1 and Theorem 1.9.

2.1 Prior work

Our starting point is the work of Jin et al. [JKLV24], which developed the encrypt-hash-and-BARG paradigm for constructing SNARGs. Before explaining this paradigm, we briefly recall the simpler hash-and-BARG paradigm from [CJJ22, KVZ21], for constructing SNARGs for deterministic and (certain special) non-deterministic computations.

Hash-and-BARG. In the hash-and-BARG candidate SNARG for an NP language \mathcal{L} , the prover sends the verifier a hash v of an assignment to all wires of the verification circuit C_x , described as follows:

- C_x has the instance x as a hard-coded input.
- C_x takes as additional input the NP witness w (this input will be empty if the computation is deterministic).
- The output of C_x is a bit b indicating whether the NP verifier accepts (x, w).

The hash that is used is a somewhere extractable hash function with local opening, as defined in [HW15, OPWW15].¹² Finally, the prover appends to this hash value a batch argument (BARG) proof.¹³ The BARG proof asserts that all the gates in the verification circuit are satisfied, i.e., for every gate:

- 1. there are local openings of the hash value v to bits b_i, b_j, b_k in the three locations (two inputs, one output) associated to this gate, and
- 2. the wire values b_i, b_j, b_k are consistent with the gate.
- 3. Finally, it is asserted that the output wire value b_{out} is 1.

A hash-and-BARG scheme is known to satisfy *local soundness* [CJJ22, KVZ21]. Suppose the somewhere extractable hash function is extractable on ℓ locations. Then, one can convert any cheating prover for the BARG into a local assignment generator LocalGen that takes as input a set of ℓ wires, outputs ℓ values, and has the following two properties: (1) the output values are locally consistent (i.e., they satisfy all gates of C_x incident on this wire set, and the output wire value is assigned 1), and (2) the output values are non-signaling, which means that for any set of wires, S and T of size at most ℓ , the distribution of the the output of LocalGen on input S restricted to the wires in $S \cap T$ is indistinguishable from the output of LocalGen on input T restricted to the wires in $S \cap T$.

Unfortunately, it is not known, and is in fact believed to be false, that this *local soundness* property implies the full soundness of the Hash-and-BARG scheme in general.

¹²Loosely speaking, such hash functions have local openings (such as Merkle hashes [Mer88]), and have the additional property that a few coordinates of the preimage can be extracted from the hash value given a corresponding trapdoor. Moreover, these coordinates are hidden in the hash key.

¹³A BARG is a succinct argument for proving k claims $x_1, \ldots, x_k \in \mathcal{L}'$, for any given NP language \mathcal{L}' , where the length of the proof is proportional to that of a single witness. BARGs are known to exist under a variety of standard assumptions such as LWE [CJJ22], bilinear maps [WW22, KLVW23], and DDH [CGJ⁺23].

From local soundness to soundness for P. For deterministic computations, it was shown in [KRR14] how to introduce redundancies to, or "extend," a circuit $C_x(\perp)$ so that local soundness of the above scheme – applied to the extended circuit – does imply full soundness. However, they also argued that this technique is limited. In particular, if the hash is extractable on ℓ locations (which in particular means that the SNARG is of size $\geq \ell$) then soundness is guaranteed only if the underlying language has a non-signaling PCP with locality ℓ , and such languages are known to be computable in time roughly 2^{ℓ} . This posed a "non-signaling barrier" to obtaining SNARGs for NP that stood for roughly 10 years.

Encrypt-hash-and-BARG. The works [BBK⁺23, JKLV24] get around this "non-signaling barrier" in the non-adaptive setting using the following modification of the Hash-and-BARG scheme, called "Encrypt-hash-and-BARG" in [JKLV24]:

- The SNARG crs is defined to contain an FHE-encryption of the description of a circuit $\langle E \rangle$, in addition to the previous hash key and BARG public parameters.
- The honest prover homomorphically evaluates E on its NP-witness w, and generates a hashand-BARG proof of both the original verification circuit and the homomorphic extended computation.

We emphasize that this proof is is still publicly verifiable, and the secret key of the encryption scheme is not used in SNARG proof generation or verification.

The reason that this modified construction is useful is that in the non-adaptive soundness analysis, there is a fixed false statement $x^* \notin \mathcal{L}$, and one can consider circuit extensions E_{x^*} that depend non-uniformly on the instance x^* . Specifically, it was proven in [JKLV24] that a cheating prover for this SNARG implies the existence of a local assignment generator LocalGen_{E_{x^*}} for any circuit E_{x^*} extending C_{x^*} . Therefore, [JKLV24] reduced the problem of constructing a SNARG for \mathcal{L} to the problem of proving the existence of a suitable circuit extension E_{x^*} for every $x^* \notin \mathcal{L}$. Crucially for them, E_x may not be easy to construct given the instance x, and may not even be well-defined for $x \in L$.

Moreover, [JKLV24] showed the existence of such families $\{E_x\}_{x\notin\mathcal{L}}$ for any language \mathcal{L} that has poly-size \mathcal{EF} proofs of the non-membership tautology for each fixed $x\notin\mathcal{L}$:

$$\forall w, C_x(w) = 0.$$

Loosely speaking, their associated extended circuit E_x contains the gates corresponding to this \mathcal{EF} proof. Since an \mathcal{EF} proof can be checked by reading only constant many bits at a time, the existence of a local assignment generator implies that the output of the verification circuit, that checks the validity of a given NP witness, must be 0.

Unfortunately, this approach still has a crucial drawback for obtaining SNARGs for all of NP: any language with \mathcal{EF} proofs as above lies in the complexity class NP \cap coNP, which is believed to be strictly contained in NP. In addition, there is little hope of this idea achieving adaptive soundness: roughly speaking, adaptive soundness would at least require these \mathcal{EF} proofs to be efficiently generatable from the instance x, but this would place the language \mathcal{L} in P. **Beyond** NP \cap coNP? In an effort to capture languages outside NP \cap coNP, [JKLV24] consider certain structured circuit extensions E_x of super-polynomial size, which (in their case) correspond to \mathcal{EF} proofs with super-polynomial size but polynomial space. Their specific theorem statement is not important for our results, but they consider a slightly modified Encrypt-hash-and-BARG scheme that we build upon to obtain our results. We discuss this in more detail in Section 2.2.

Nevertheless, the improved SNARG of [JKLV24] is still limited to languages in the complexity class NP \cap coNTIME(2^{ℓ}), where their SNARG proofs have length greater than ℓ . This limitation is inherent to their approach, as formalized in [JKLV24] Theorem 5.8.

2.2 Our Non-adaptive SNARG and Analysis

The [JKLV24] approach cannot directly be extended to NP due to the following fundamental limitation: their security analysis crucially makes use of a form of efficient proof of the statement that $x^* \notin \mathcal{L}$. However, observe that their approach can be reformulated as working with the claim:

$$\forall w, \Big(C(x^*, w) = 1 \to 1 = 0\Big).$$

In this work, we consider a simple but powerful generalization of their approach that requires \mathcal{EF} proofs of claims of the form

$$\forall w, \Big(C(x,w) = 1 \to f(w) = 1\Big),$$

for an appropriately chosen function f. The advantage of this generalization is that one can hope to work with claims that can even hold for *all* statements x (true and false), which reopens the possibility of building SNARGs for all of NP.

Informally, we make use of the following paradigm generalizing [JKLV24]:

Any claim about *real* NP *witnesses* that can be proved in \mathcal{EF} can also be enforced to hold for *local assignment generators*.

We will use this approach to deduce the security of our SNARG assuming the security of another cryptographic primitive, which will only appear in the security analysis! This is accomplished by considering circuit extensions E_x that themselves employ cryptography, rather than being information-theoretic.

Cryptography to the rescue! For simplicity, we exemplify our approach assuming the existence of a witness encryption scheme. Recall that a witness encryption scheme consists of a randomized algorithm Enc(x,m) = Enc(x,m;r) along with an algorithm Dec(w,ct) such that Dec(w,Enc(x,m)) = m for all (x,w) such that $C_x(w) = 1$. Given such a scheme, consider a candidate SNARG defined as follows:

- The crs includes an FHE encryption of a witness encryption ciphertext ct = Enc(x, m) for a uniformly random message m ← {0,1}^λ.
- The honest prover homomorphically evaluates the witness decryption circuit Dec(w, ct) on the FHE ciphertext, and provides a Hash-and-BARG proof that (i) $C_x(w) = 1$ and (ii) the homomorphic computation was computed correctly.

We emphasize that, as before, the honest verifier does not have the FHE secret key and merely verifies the BARG proof.

Nevertheless, suppose (wishful thinking, for now) that it were possible to prove that for any cheating prover P^* for the SNARG, P^* outputs a valid FHE encryption of the message m (as the honest prover does when $x \in \mathcal{L}$). This would constitute a security reduction proving that the SNARG is secure assuming the security of the witness encryption scheme! This is because P^* , even it were additionally given the FHE secret key, cannot efficiently recover the message m without breaking the witness encryption.

Unfortunately, it is not clear how to instantiate this wishful thinking in general – after all, the verifier V does not decrypt the prover's ciphertext, so perhaps an adversary P^* can make V accept using an incorrect evaluated ciphertext.

 \mathcal{EF} proofs of correctness to the rescue. The final piece of the puzzle is to employ an appropriate \mathcal{EF} proof that forces a cheating P^* to *correctly* send an FHE encryption of m. Roughly speaking, we consider the following modification of the above SNARG:

- The crs includes (1) an FHE encryption of a witness encryption ciphertext ct = Enc(x, m), and
 (2) a separate FHE encryption of the description of a circuit (E).
- The honest prover homomorphically evaluates Dec(w, ct) on the first FHE ciphertext, homomorphically evaluates $\langle E \rangle \mapsto E(w)$ on the second ciphertext, and provides a Hash-and-BARG proof that:
 - (i) $C_x(w) = 1$, and
 - (ii) the homomorphic computations on ciphertexts (1) and (2) were computed correctly.

Our analysis assumes the existence of an \mathcal{EF} proof of the following decryption correctness statement for every fixed statement x, message m and randomness r:

$$\forall w, (C_x(w) = 1 \rightarrow \mathsf{Dec}(w, \mathsf{Enc}(x, m; r)) = m).$$

In our security analysis, we replace $\langle E \rangle$ with the description of a circuit $E_{x,m,r}$ described in Fig. 1. This circuit consists of the following sub-computations:

- 1. Hard-coded inputs: encryption randomness r, random message m, the instance x.
- 2. Input: a hypothetical NP witness w.
- 3. The encryption procedure $\mathsf{Enc}(x,m;r) \to \mathsf{ct}$.
- 4. The decryption procedure $(w, \mathsf{ct}) \mapsto \mathsf{Dec}(w, \mathsf{ct}) = m'$.
- 5. A circuit Equality_m that checks whether m' = m. Call the resulting output bit b.
- 6. An encoding of the \mathcal{EF} proof that if $C_x(w) = 1$ then b = 1.



Figure 1: Visual depiction of the extended circuit $E_{x,m,r}$. Note that here, w is an input to both the relation circuit C_x , as well as the decryption circuit Dec. The \mathcal{EF} proof of completeness takes as input all the wires in w, C_x , Enc, Dec and Equality_m (since these appear as variables in the \mathcal{EF} proof).

Similarly to before, a cheating prover P^* for this SNARG (in this hybrid) implies the existence of a local assignment generator for computations Items (i) and (ii). By our basic principle, one can deduce that this local assignment generator assigns 1 to the bit b (i.e., the local assignment generator is "as good as a real NP witness" in this respect). Finally, we argue that this fact actually implies that the FHE decryption of P^* 's output of computation (1) must actually be equal to m. This contradicts the security of the witness encryption scheme, and thus we can prove the soundness of our SNARG for any NP language.

We remark that this security proof makes use of the \mathcal{EF} proof of correctness – which intuitively is a claim about *true* $x \in \mathcal{L}$ – in our *soundness analysis*, where $x \notin \mathcal{L}$! The key point is that this \mathcal{EF} correctness proof is useful precisely under the assumption that a cheating P^* exists, because:

- 1. P^* implies the existence of a kind of "pseudo-witness" for x, and
- 2. The \mathcal{EF} proof of correctness also applies to this "pseudo-witness" (and false statement x) in the security proof.

Our Universal SNARG. We make the following brief observations about our presented analysis:

• Rather than using a witness encryption scheme, we could employ any 2-message laconic argument system for the language \mathcal{L} . What is required for the analysis to work is a family of \mathcal{EF} proofs of *completeness* for the argument system:

$$\forall w, \left(C_x(w) = 1 \to V(x, P(w, \mu), r) = 1 \right)$$

where r denotes the verifier's randomness, $\mu = V_1(x; r)$ denotes an honestly computed verifier message, and $V(\cdot)$ denotes the laconic argument verification procedure.

• Our construction does not need to depend on the choice of laconic argument: the relevant pieces of the crs in our SNARG can always be set to FHE encryptions of "dummy" all-zeroes strings, and the prover can be asked to homomorphically evaluate a universal circuit on this ciphertext.

As a result, our Encrypt-hash-and-BARG proof system is secure assuming that *any* 2-message laconic argument system for NP, with \mathcal{EF} proofs of correctness, exists! As usual, our scheme can also be made to have transparent **crs** under the assumption that the FHE has pseudorandom ciphertexts.

Technical details: universal local assignment generators. Although we have outlined the main new ideas behind Theorem 1.1, there are several details missing in the above security proof sketch. In the body of the paper, we handle these details by abstracting out a security property of the multi-ciphertext Encrypt-hash-and-BARG scheme of [JKLV24] Section 5 (which was originally employed for their "bounded space" result). Specifically, we observe that any cheating prover P^* for this scheme implies the existence of what we call a universal local assignment generator

LocalGen(E, S)

that takes as input the description of a circuit E (containing C_x) along with a wire set S (while an ordinary local assignment generator is defined with respect to a fixed circuit E and only takes S as input). This universal local assignment generator satisfies two properties:

- For every fixed extension E of C_x , we have that $\mathsf{LocalGen}(E, \cdot)$ is a valid local assignment generator for E, and
- LocalGen is also non-signaling with respect to the *choice of* E: for any two extensions E and E' (of some bounded size) and for every set of wires S that are present in both E and E' it holds that

 $LocalGen(E, S) \approx_c LocalGen(E', S).$

This formalism turns out to capture what is needed to carry out the above security analysis, without having to explicitly refer to multiple FHE ciphertexts encrypted under different keys. While [JKLV24] did not formalize this property of their construction, their security proof implicitly establishes it. Subsequent to this work, the authors of [JKLV24] updated their paper to include this formalism for ease of understanding.

2.3 Adaptive Soundness

We next show how to convert any dv-SNARG with a polynomial size \mathcal{EF} proof of correctness, into a (publicly verifiable) SNARG. The construction is simple: append to the dv-SNARG π an encrypthash-and-BARG proof that π was generated by (honestly) running the dv-SNARG prover on a valid witness w (i.e., on w for which $(x, w) \in \mathcal{R}_{\mathcal{L}}$). Now rather than verifying the dv-SNARG π , which requires a secret verification key, the verifier only verifies the encrypt-hash-and-BARG proof, which can be done publicly. At first it may seem unclear why the encrypt-hash-and-BARG is at all helpful in the adaptive setting since in this setting changing the crs in the analysis, may cause a change to the distribution of the instance provided by the cheating prover. In particular, the distribution of the instance may change from being not in the language to being in the language.¹⁴ Surprisingly, even though this may indeed be the case, we can still argue that the prover will be rejected, even if it did change the instance to one in the language.

In the analysis, we start with a cheating prover who successfully cheats with adaptively chosen (x^*, π^*) , where x^* is not in the language and π^* is a dv-SNARG. The adaptive soundness of the underlying dv-SNARG implies that π^* must be rejecting. Next, in the analysis we modify the crs in a similar way as in the non-adaptive case. Specifically, we change the ciphertext encrypting the all zero string to one that encrypts a description of the extended circuit, that on input (x^*, π^*) , computes the verdict $b = \mathcal{V}(\mathsf{vk}, x^*, \mathsf{pk}^*)$ and then adds gates corresponding to the \mathcal{EF} proof that if $C_x(w) = 1$ then b = 1.

We argue that after this change, π^* must still be rejecting, whether or not x^* is in the language. This is due to the security of the underlying FHE scheme which says that the distribution of π^* even given vk should look the same as before, and thus should remain rejecting. At this point, the adversary P^* is outputting an instance x^* (possibly true or false), a rejecting dv-SNARG proof π^* , and an accepting encrypt-hash-and BARG proof that π^* was computed by running the honest prover \mathcal{P} on a valid NP witness w for x^* , and that the extended circuit was homomorphically evaluated on this w.

From here, we can derive a contradiction similarly to our argument in the non-adaptive setting. The main difference is that we need to rely on the fact that in the adaptive setting, the encrypt-hash-and-BARG scheme can be used to convert an adaptive cheating prover into an adaptive universal and non-signaling local assignment generator.¹⁵

Obtaining a transparent crs. Finally we mention that if the public key pk of the dv-SNARG is $2^{|x|}$ -indistinguishable from uniform, then we can make our **crs** transparent! The idea is to replace pk with a truly random string, and thus obtain a transparent **crs**, since the **crs** of our encrypt-hash-and-BARG scheme is already transparent (under LWE). The observation is that since we changed the **crs** in a $2^{|x|}$ -indistinguishable way, we can use complexity leveraging to argue that adaptive soundness still holds.

2.4 Overview of Applications

In this section, we describe how to apply Theorems 1.1 and 1.9 to obtain several of our applications. Obtaining these applications requires two key ingredients, that we exemplify in this overview:

- Constructing \mathcal{EF} proofs of correctness for various instantiations of cryptographic primitives.
- Showing how transformations between primitives preserve \mathcal{EF} proofs of correctness.

We remark that similarly to [JJ22, JKLV24], almost all¹⁶ polynomial-length \mathcal{EF} proofs constructed

¹⁴This can be overcome by using standard complexity leveraging techniques, and assuming that the change in the crs is harder to detect than the hardness of the language. However, this will make our proofs grow at least poly-logarithmically with the hardness of the language, which limits the result.

¹⁵We mention that this was not done explicitly in [JKLV24], but their proof directly extends to this setting, similarly to how the hash-and-BARG scheme extends to the adaptive setting.

¹⁶The exceptions are the results about QR and DCR; see Section 6.4.

in this paper can be derived from a stronger object in proof complexity called a PV proof [Coo75]. While \mathcal{EF} proofs are asymptotic families of proofs about growing families of inputs, PV proofs are finite-length proofs about the behavior of a *Turing machine* on *all inputs* in $\{0, 1\}^*$. Roughly speaking, PV proofs are a restricted family of proofs in Peano Arithmetic that make use of *computationally efficient* proofs by induction – e.g., induction over $\{0, 1\}^*$ with respect to the input length n rather than standard induction over \mathbb{Z} . It is well known [Coo75] that PV proofs can be converted, in uniform poly(n) time, into an associated \mathcal{EF} proof about inputs of length n. For a more comprehensive introduction to the PV and \mathcal{EF} proof systems, we refer the reader to Section 3.5.

2.4.1 SNARGs for NP from Evasive LWE

To construct a SNARG for NP from evasive LWE, as argued in Section 2.2, it suffices to construct a witness encryption scheme from evasive LWE that has an \mathcal{EF} proof of decryption correctness.

Prior constructions of witness encryption from evasive LWE. While the works of [CVW18, VWW22, Tsa22] already construct witness encryption schemes from evasive LWE, these schemes do not have *perfect correctness*. We refer the reader to the respective works for the actual constructions, but we give intuition for why these schemes incur correctness error.

The scheme [VWW22, Tsa22] schemes on input x and message m output a program $P_{K,m,x}$: $\{0,1\}^m \to \mathbb{Z}_q$ which has the following (abridged) form:

$$P_{K,m,x}(w) = \begin{cases} m \cdot \mathsf{PRF}_K(w) + e_w & \text{If } R(x,w) = 1\\ \mathsf{PRF}_K(w) + e_w & \text{otherwise.} \end{cases}$$

where each e_w is some term bounded by some $|e_w| \leq B$, where $B \ll q$. The decryption algorithm of the witness encryption then computes $y \leftarrow P_{K,m,x}(w)$, and outputs 0 if $|y| \leq B$, and 1 otherwise. It is easy to see that in general, such a witness encryption scheme may not have perfect correctness if the encrypted message is m = 1. For example, if $\mathsf{PRF}_{K,m,x}(w) = 0$, then $|y| \leq B$ and the decoder will deduce that the message is 0. For appropriate parameter settings, this scheme achieves *statistical* correctness because with high probability, $|\mathsf{PRF}_K(w)| \geq B$ (otherwise, this allows us to distinguish PRF from a uniformly random function). In particular, it is not *perfectly correct*, and it is not clear that such schemes should have (appropriately defined) \mathcal{EF} proofs of correctness.

New construction admitting an \mathcal{EF} proof. Therefore, we instead construct a witness encryption scheme with *perfect correctness* using the techniques of the recent work of [MPV24]. We then show that the scheme does in fact have an \mathcal{EF} proof of correctness. We proceed in a few steps:

Step 1: \mathcal{EF} proof of approximate correctness of MPV Obfuscation. First, we show that " σ -PRF obfuscation", an object defined by [MPV24], in fact has a polynomial-sized \mathcal{EF} proof of *approximate* correctness:

for all circuits
$$f : \{0,1\}^h \to \mathbb{Z}_q$$
 and all randomness r ,
if $g \leftarrow \mathcal{O}(f;r)$, then for all $x \in \{0,1\}^h$, $|g(x) - f(x)| \leq B$

for some bound B. Since the construction is lattice-based, we show the correctness of the scheme by mainly appealing to basic arithmetic properties and linear algebra facts which are

known to be provable in \mathcal{EF} [Coo75, Bus86, SC04]. We refer the reader to Section 6.3.1 for more details.

Step 2: New witness encryption scheme with \mathcal{EF} proof, Consider the following keyed program:

$$P_{K,m,x}(w) = \begin{cases} m \cdot \lfloor q/2 \rfloor & \text{if } R(x,w) = 1\\ f_K(w) & \text{otherwise} \end{cases}$$

where f_K is some keyed function (which we will define explicitly in the next step). Let $\tilde{P} \leftarrow \mathcal{O}(P_{K,m,x})$, where \mathcal{O} is the obfuscation from the previous step. Set parameters of the obfuscation so that $B \ll q/4$ (recall B is the correctness bound from the previous step). The encryption works as follows: on input an instance x and message m, output \tilde{P} corresponding to $P_{K,m,x}(w)$ where K is uniformly random. Decryption then computes $\tilde{P}(w)$, and outputs 0 if $|\tilde{P}(w)| \leq B$, and 1 if $|\tilde{P}(w) - \lfloor q/2 \rfloor| \leq B$. It is easy to see that the above construction is correct via the following argument:

- By construction, if R(x, w) = 1, then $P_{K,m,x}(w) = m \cdot \lfloor q/2 \rfloor$.
- By approximate correctness of the obfuscation, we have $|\tilde{P}(w) P_{K,m,x}(w)| \leq B$.
- If m = 0, then $B \ge |\tilde{P}(w) P_{K,m,x}(w)| = |\tilde{P}(w) 0| = |\tilde{P}(w)|$. Hence, decryption succeeds.
- Similarly, if m = 1, then $B \ge |\tilde{P}(w) P_{K,m,x}(w)| = |\tilde{P}(w) \lfloor q/2 \rfloor|$. Hence, decryption succeeds.
- Therefore, the decryption algorithm is correct.

In fact, this argument can be easily formalized as a polynomial-sized \mathcal{EF} proof.

Step 3: Security of the new scheme. It now suffices to show security of the scheme. Recall that the security guarantee of the σ -PRF obfuscation of [MPV24] guarantee is as follows: if the family of functions $\{f_k\}_{k \in \mathcal{K}}$ satisfies that given $k \leftarrow \mathcal{K}$ and independent Gaussian samples $e_x \leftarrow \mathcal{D}_{\sigma}$ (shorthand denoting that the standard deviation σ), we have

$$\{f_k(x) + e_x\}_{x \in \mathcal{X}} \approx_c \mathcal{U},\tag{1}$$

then the obfuscation $\mathcal{O}(f_k)$ for $k \leftarrow \mathcal{K}$ is pseudorandom. In the case where $x \notin L$, we have that the program $P_{K,m,x}(w) = f_K(w)$. If we choose f_K from a family of functions such that (1) holds, then we can argue that the witness encryption corresponding to $x \notin L$ is pseudorandom and hence semantically secure. We instantiate f_K via the BLMR PRF [BLMR13] to obtain the desired guarantee¹⁷.

¹⁷Although any PRF would satisfy (1), the work of [MPV24] requires the program to be represented as a "matrix program". However, it is known that such a matrix PRFs do not exist [CHVW19]. Hence, we have to rely on matrix programs that satisfy (1), which [MPV24] calls ' σ -PRF'. We gloss over this detail in this technical overview for readability.

2.4.2 SNARGs for Quadratic Residuousity and (QR) and Nth-Residuousity (DCR)

Theorem 1.1 also implies new constructions of SNARGs for *specific* NP *languages*, where the soundness of the SNARG holds under the LWE assumption alone. The reason for this is that Theorem 1.1 can be applied to *any* NP *language*, and thus, assuming LWE, we build SNARGs for any NP language \mathcal{L} supporting a witness encryption scheme that has \mathcal{EF} proofs of decryption correctness. This opens the possibility of obtaining SNARGs for specific languages based on LWE alone, by making use of *unconditional, statistically secure* witness encryption schemes.

To see that this yields new SNARGs, we first consider a particular implementation of the Goldwasser-Micali encryption scheme [GM82], which can be viewed as a witness encryption scheme for the quadratic non-residuosity language:

- The encryption key has the form (N, y) where N = pq is a product of two primes and y is a quadratic non-residue modulo N with Jacobi symbol 1.
- The decryption key consists of the factorization (p,q) of N.
- To encrypt a message bit b, output $r^2 \cdot y^b$ modulo N.
- To decrypt a ciphertext c, compute $c^{\frac{p-1}{2}}$ modulo p (and check whether the result is 1).

This encryption scheme *nearly* has an \mathcal{EF} proof of decryption correctness, as basic algebra implies that:

$$\forall p, q, N, r: \text{ if } pq = N \text{ and } p, q > 1, \text{ and } y^{\frac{p-1}{2}} \equiv -1 \text{ mod } p,$$

then $(r^2y^b \mod N)^{\frac{p-1}{2}} \equiv r^{p-1} \cdot (-1)^b \mod p.$

This proves correctness of decryption modulo an invocation of Fermat's little Theorem. Unfortunately, this theorem is not known to have polynomial length \mathcal{EF} proofs.

In order to get around this issue, we consider a witness encryption scheme with respect to a particular Arthur-Merlin (AM) verifier for the QNR language: in addition to verifying the factorization N = pq, the AM verifier has access to random $g_1, \ldots, g_n \in \mathbb{Z}_N^{\times}$, and checks that $g_i^{p-1} \equiv 1 \pmod{p}$ (as long as $gcd(g_i, N) = 1$) for every *i*.

Then, we consider the modified witness encryption scheme that samples the encryption randomness r as a random product of powers of the g_i . This yields a \mathcal{EF} proof of decryption correctness, because Fermat's little Theorem was explicitly verified to hold on the g_i , and this implies (in \mathcal{EF}) that it also holds for r. Finally, security of this scheme is unaffected because with overwhelming probability over g_1, \ldots, g_n , the string r sampled in this way is uniform over \mathbb{Z}_N^{\times} .

Plugging this witness encryption scheme into Theorem 1.1 yields a new SNARG for the QNR promise language¹⁸ as well as (after suitable manipulation) the QR promise language; similarly, by considering an appropriate variant of the Paillier encryption scheme, one can obtain a SNARG for the decisional composite *non*-residuosity problem.

In particular, we emphasize that these languages were not known to have SNARGs based on [JKLV24]; the reason for this is that proofs of *non-membership* implicitly seem to require proving the uniqueness of the prime factorization of N, which we do not know how to argue in \mathcal{EF} . On the other hand, the decryption correctness of this scheme follows from simple properties of algebra and, in particular, needs only consider an arbitrary single factorization of N.

 $^{^{18}}$ The SNARG is still defined with respect to the standard NP verifier for QNR.

2.4.3 Transparent Non-Adaptive SNARG and Adaptive SNARG via iO

 \mathcal{EF} -proof for the Sahai-Waters SNARG. Recall that the work of Sahai and Waters [SW14] constructed a SNARG for NP via iO. Our first observation is the following: assuming that iO has a polynomial-size \mathcal{EF} proof of correctness, one can show that the Sahai-Waters scheme also has a polynomial-sized \mathcal{EF} proof of completeness. We describe this proof in Section 7.4. Additionally, the work of Mathialagan, Peters and Vaikuntanathan [MPV24] showed that the Sahai-Waters construction is *adaptively* sound in the *designated-verifier* setting. This scheme also has an \mathcal{EF} proof of completeness, as we show in Section 7.5. Therefore, assuming LWE as well as an iO scheme with a proof of correctness, we can construct:

- non-adaptively sound SNARG for NP with *transparent* setup via Theorem 1.1.
- adaptively sound SNARG for NP with *public verifiability* via Theorem 1.9.

Therefore, it remains to argue that there exist iO constructions with \mathcal{EF} proofs of correctness.

Boosting any iO to have an \mathcal{EF} proof. We handle this question completely generically: given any subexponentially secure iO (along with subexponentially secure LWE), we construct an iO scheme with an \mathcal{EF} proof of correctness. Formally, we say that an iO scheme has a polynomial-sized \mathcal{EF} proof of correctness if there is a polynomial-sized \mathcal{EF} proof of the fact that:

for all circuits C and randomness r, if $\Gamma = iO(C; r)$, then for all x, $C(x) = \Gamma(x)$.

In fact, we prove a stronger statement that the resulting iO has a *uniform* \mathcal{EF} proof (also known as PV proof), but we skip over this detail in this overview.

We now describe the transformation. Let iO be subexponentially secure obfuscation scheme.

- Step 1: Obtaining an inefficient iO with perfect correctness. We first use iO to construct the following inefficient algorithm which we call slowXiO which works as follows: On input a circuit $C : \{0, 1\}^n \to \{0, 1\}$ and randomness r,
 - 1. Compute $\Gamma = iO(C; r)$.
 - 2. Iterate over $x \in \{0, 1\}^n$, and check if $\Gamma(x) = C(x)$.
 - 3. If $\Gamma(x) \neq C(x)$ for any x, abort and output C.
 - 4. Else, output Γ .

Intuitively, slowXiO algorithm is verifying the correctness of the output of iO, and outputting the "trivial" obfuscation (i.e. C itself) if it is incorrect. It is easy to see that this scheme is in fact perfectly correct since either $\Gamma(x) = C(x)$ for all x, or the output is C itself. And in fact, this correctness property is easy to formalize in propositional logic (see Lemma 7.7).

Now, we need to argue that slowXiO is a secure obfuscation scheme. In fact, for functionally equivalent circuits C_1 and C_2

$$\mathsf{slowXiO}(C_1;r) \approx_s \mathsf{iO}(C_1;r) \approx_c \mathsf{iO}(C_2;r) \approx_s \mathsf{slowXiO}(C_2;r)$$

where the first and third indistinguishabilities follow from the statistical correctness of iO, and the second indistinguishability follows the security of iO. Therefore, slowXiO is in fact a secure scheme.

Step 2: XiO bootstrapping. The above construction of slowXiO, while secure and correct, is exponentially inefficient. In particular, the runtime of the algorithm is $2^n \cdot \text{poly}(|C|, \lambda)$, and the depth of the algorithm is $\text{poly}(|C|, \lambda)$ (since the check that $\Gamma(x) = C(x)$ can be conducted in parallel).

To now construct an *efficient* iO scheme, we follow the iO bootstrapping literature [AJ15, BV15, LPST16a, LPST16b, Vai24]. Since slowXiO is in fact an XiO scheme (following the terminology of [LPST16a]), we can use ideas from the bootstrapping literature to construct an efficient iO scheme. By showing that the chain of transformations *preserve* \mathcal{EF} proofs of correctness, we obtain our main result.

The main building block in the transformation is the succinct functional encryption scheme of cite [GKP⁺13], which uses fully homomorphic encryption (FHE), attribute-based encryption (ABE) and garbled circuits. Since FHE and ABE can be instantiated via LWE [BV11, GSW13, BGG⁺14], we use the fact that many properties in linear algebra can be proven in \mathcal{EF} to show that these instantiations have \mathcal{EF} proofs of correctness. For garbled circuits, we follow the construction of Yao, with a modification for perfect correctness as suggested in [LP09], to show that it has polynomial-sized \mathcal{EF} proof of correctness. We show that these instantiations have \mathcal{EF} proofs of correctness as suggested in stantiations have \mathcal{EF} proofs of correctness. We show that these instantiations have \mathcal{EF} proofs of correctness. We show that these instantiations have \mathcal{EF} proofs of correctness as suggested in [LP09], to show that it has polynomial-sized \mathcal{EF} proof of correctness. We show that these instantiations have \mathcal{EF} proofs of correctness as a suggested in [LP09], to show that it has polynomial-sized \mathcal{EF} proof of correctness. We show that these instantiations have \mathcal{EF} proofs of correctness and C.

2.4.4 Transparent adaptive SNARG for UP

The work of [MPV24] constructed a *designated-verifier* SNARG for UP from evasive LWE and LWE. Moreover, they showed that their SNARG is *adaptively sound*. If we additionally show that their construction has a polynomial-sized \mathcal{EF} proof of completeness, then we can apply our transformation from Theorem 1.9 to obtain a *publicly verifiable* SNARG for UP from evasive LWE and LWE. As discussed in Section 2.4.1, the work of [MPV24] relies primarily on lattice-based techniques, so we can argue similarly that their construction has a polynomial-sized \mathcal{EF} proof of completeness.

Obtaining transparent crs. Recall that the transformation from designated-verifier to publicly verifiable adaptive SNARGs does not generically guarantee that the crs is transparent (unlike in the non-adaptive setting). However, if we can show that the crs of the designated-verifier is subexponentially indistinguishable from uniformly random, then we can construct an adaptively sound SNARG with *transparent* crs. Although the crs in the construction of [MPV24] is an *obfuscated program*, it is a result of a σ -PRF obfuscation, which guarantees that the obfuscation is pseudorandom. Therefore, this allows us to achieve our desired result.

2.4.5 Universal Micali SNARG

Recall that Micali's SNARG [Mic94] took Killian's interactive argument system [Kil92] and compiled it with the Fiat-Shamir heuristic [FS87]. Micali's SNARG works as follows:

- The crs algorithm outputs a hash key hk.
- The prover does the following:
 - Compute a PCP proof $\pi \leftarrow \mathsf{PCP}.\mathcal{P}(x, w)$.

- Compute the Merkle hash root $\mathsf{rt} = \mathcal{M}.\mathsf{Hash}(\mathsf{hk}, \pi)$. Here, we use $\mathcal{M}.\mathsf{Hash}$ as shorthand Merkle tree-based hash where $\mathsf{Hash}(\mathsf{hk}, \cdot)$ is used at each internal node.
- Apply the hash function on the root rt to obtain $r \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathsf{rt})$.
- Parse r as randomness, and compute the output of the PCP query function $Q \leftarrow \mathsf{PCP}.\mathcal{Q}(r)$. Note that this is the Fiat-Shamir step.
- Compute $\pi|_Q$ (the locations of π corresponding to queries of Q) as well as the Merkle openings $\rho_Q = \{\rho_q\}_{q \in Q}$ of rt to $\pi|_Q$.
- Output $(\mathsf{rt}, \pi|_Q, \rho_Q)$.
- The verifier computes $\mathsf{PCP}.\mathcal{V}(x,Q,\pi_Q)$ and if ρ_Q are valid openings to $\pi|_Q$.

It is known that this scheme is secure if **Hash** were instantiated via a random oracle. However, we do not have any constructions of **Hash** from well-studied assumptions that are provably secure.

Proof of completeness. It turns out that the above scheme is *complete* regardless of what hash function Hash is used to instantiate the SNARG. To see this, note that it is sufficient to show that the verifier accepts if the prover computes π honestly following the PCP, and computes the Merkle root and openings honestly, then the verifier accepts. As a result, only need the following two facts:

• the PCP has a proof of completeness, i.e. a proof of the fact that:

for all x, w such that R(x, w) = 1, and $\pi = \mathsf{PCP}.\mathcal{P}(x, w)$, for all randomness rand queries $Q \leftarrow \mathsf{PCP}.\mathcal{Q}(r)$, we have that $\mathsf{PCP}.\mathcal{V}(x, Q, \pi_Q) = 1$.

• the opening correctness of Merkle trees, i.e., a proof of the fact that:

for all hash keys hk, if $\mathsf{rt} \leftarrow \mathcal{M}.\mathsf{Hash}(\mathsf{hk}, (x_1, \dots, x_N))$, and $\rho = \mathcal{M}.\mathsf{Open}(\mathsf{hk}, (x_1, \dots, x_N), i)$ is a Merkle opening to the *i*th bit $b = x_i$, then $\mathcal{M}.\mathsf{Ver}(\mathsf{hk}, \mathsf{rt}, i, b, \rho) = 1$.

The work of Pich [Pic15] showed that Dinur's PCP construction [Din07] in fact has an \mathcal{EF} proof of completeness. In Section 9.3, we further show that the opening completeness of Merkle trees, with respect to *any* underlying hash function, can be formalized and proven in Cook's theory PV. The approach involves first formalize the binary-tree construction of Merkle hash function as a function symbol in PV using recursive definition of function symbols. We then inductively prove the opening completeness property of the Merkle hash. Since PV supports the polynomial-time induction rule, this inductive proof can be fully formalized within PV. Therefore, the completeness of Micali's SNARG can be easily proven as a polynomial-size proof in \mathcal{EF} , regardless of what hash function is used.

Corollary: Universali Micali SNARG. If there exists *any* hash function with respect to which Micali's SNARG is secure, since the scheme also has an \mathcal{EF} proof of completeness, our universal SNARG is sound.

3 Preliminaries

Notations. We will let λ denote the security parameter throughout the paper. We use PPT to denote probabilistic polynomial-time, and denote the set of all positive integers up to n as $[n] := \{1, \ldots, n\}$. For any $x \in \{0, 1\}^n$ and any subset $J \subset [n]$ we denote by $x_J = (x_j)_{j \in J}$. For any finite set $S, x \leftarrow S$ denotes a uniformly random element x from the set S. Similarly, for any distribution $\mathcal{D}, x \leftarrow \mathcal{D}$ denotes an element x drawn from the distribution \mathcal{D} .

3.1 LWE Assumption

Given $n, m, q \in \mathbb{N}$ and $\sigma, \delta > 0$, the subexponential LWE assumption $\mathsf{LWE}_{n,m,q,\sigma}^{\delta}$ [Reg05] asserts that

$$(\mathbf{A}, \mathbf{sA} + \mathbf{e}) \approx_c (\mathbf{A}, \mathbf{b}),$$

with security parameter $\mu = 2^{n^{\delta}}$, where $\mathbf{s} \leftarrow \mathcal{U}(\mathbb{Z}_q^n)$, $\mathbf{A} \leftarrow \mathcal{U}(\mathbb{Z}_q^{n \times m})$, $\mathbf{e} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}^m$, and $\mathbf{b} \leftarrow \mathbb{Z}_q^m$. Here, $\mathcal{D}_{\mathbb{Z},\sigma}$ denotes the discrete Gaussian over \mathbb{Z} with parameter σ ; that is, the distribution which assigns mass proportional to $\exp(-\pi x^2/\sigma^2)$ to each $x \in \mathbb{Z}$.

Following [VWW22], we rely on the above assumption holding for some $\delta > 0$, for parameters such that $q/\sigma \leq 2^{n^{\delta}}$.

Statistical tools. The following lemma is a simple consequence of Banaczyzck's tail bound (see, e.g., [SD17, Corollary 1.3.11]).

Lemma 3.1. For any $\sigma > 0$ and $n \in \mathbb{N}$,

$$\Pr_{x \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}}[|x| \ge \sigma \sqrt{n}] \le 2^{-n}$$

The following simple noise flooding lemma is integral to our uses of subexponential LWE.

Lemma 3.2 ([CVW18, Lemma 3.2]). For all $y \in \mathbb{Z}$ and $\sigma \in \mathbb{R}$, the statistical distance between $\mathcal{D}_{\mathbb{Z},\sigma}$ and $\mathcal{D}_{\mathbb{Z},\sigma} + y$ is at most y/σ .

LWE-based PRFs. The following lemma adapts the PRF from [BLMR13, Theorem 5.1] to the setting of subexponential LWE.

Lemma 3.3. Let $n, n', q, h \in \mathbb{N}$ and $\mu, f, k, \sigma, \sigma' \in \mathbb{R}$ be functions of λ satisfying

• $\mu = 2^{(n')^{\delta}}$

•
$$\lceil (2\log q) \cdot n' \rceil \le n \le \operatorname{poly}(n'),$$

•
$$\sigma' \ge k \cdot (n^2 \sigma)^{h+1}$$
,

•
$$2^{-n^{\delta}} \leq 1/k(n) \leq \operatorname{negl}(2^{h} \cdot n \cdot f(n))$$

Let

$$\mathbf{a} \leftarrow \mathbb{Z}_q^n, \{\mathbf{e}_{\mathbf{x}} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma'}\}_{x \in \{0,1\}^h}.$$

Then, assuming LWE^{δ}_{n',poly[n'],q, σ},

$$\{\mathbf{S}_{i,b} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}^{n \times n}\}_{\substack{i \in [h] \\ b \in \{0,1\}}}, \left\{ \left(\prod_{i=1}^{h} \mathbf{S}_{i,\mathbf{x}_{i}}\right) \cdot \mathbf{a} + \mathbf{e}_{\mathbf{x}} \right\}_{\mathbf{x} \in \{0,1\}^{h}} \approx_{c} \{\mathbf{S}_{i,b} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}^{n \times n}\}_{\substack{i \in [h] \\ b \in \{0,1\}}}, \{\mathcal{U}\}_{\mathbf{x} \in \{0,1\}^{h}}.$$

where all poly[μ]-time distinguishers have advantage at most negl(f(n)). In particular, one can take $q = 2^{n^{\delta}}$ (to rely on subexponential LWE with $q/\sigma \leq 2^{n^{\delta}}$), $h = n^{c}$ for some $c < \delta/20$, $k = 2^{h^{3}\lambda}$, and $f(n) = 2^{-h^{2}\lambda}$.

3.2 Fully Homomorphic Encryption

In this section, we define the type of fully homomorphic encryption used in this paper. We make use of a *leveled*, *gate-by-gate* FHE satisfying a *malicious* correctness property: for any pair of ciphertexts ct_0, ct_1 that decrypt to bits x_0, x_1 , it should be the case that $Dec_{sk}(Eval_{ek}(f, ct_0, ct_1)) = f(x_0, x_1)$. Moreover, our definition explicitly describes the evaluation key as a sequence of d (small) keys, since this structure is relevant to our SNARG construction.

Syntax. A leveled, gate-by-gate fully homomorphic encryption scheme consists of a fixed key/ciphertext size $\ell = \ell(\lambda) = \text{poly}(\lambda)$ and the following polynomial time algorithms:

- $\mathsf{FHE.Setup}(1^n, 1^d) \to (\mathsf{pk}, \mathsf{ek}_1, \dots, \mathsf{ek}_d, \mathsf{sk}_0, \dots, \mathsf{sk}_d).$ This is a probabilistic algorithm that takes as input a security parameter 1^n and a circuit depth 1^d . It outputs a public key $\mathsf{pk} \in \{0, 1\}^{\ell}$, a sequence of d evaluation keys $\mathsf{ek}_1, \dots, \mathsf{ek}_d \in \{0, 1\}^{\ell}$, and a sequence of d + 1 secret keys $\mathsf{sk}_0, \dots, \mathsf{sk}_d \in \{0, 1\}^{\ell}$.
- $\mathsf{FHE}.\mathsf{Enc}_{\mathsf{pk}}(b) \to c$. This is a probabilistic algorithm that takes as input a public key pk and a bit $b \in \{0, 1\}$. It outputs a ciphertext $c \in \{0, 1\}^{\ell}$.
- $\mathsf{FHE.Dec_{sk}}(c) \to b$. This is a deterministic algorithm that takes as input a secret key sk and a ciphertext $c \in \{0, 1\}^{\ell}$. It outputs a bit $b \in \{0, 1\}$.
- FHE.GateEval_{ek} $(f, c_1, c_2) \rightarrow c^*$. This is a deterministic algorithm that takes as input an evaluation key ek, the truth table of a *two input bit* function $f: \{0, 1\} \times \{0, 1\} \rightarrow \{0, 1\}$ and two ciphertexts $c_1, c_2 \in \{0, 1\}^{\ell}$. It outputs a ciphertext $c^* \in \{0, 1\}^{\ell}$.

By iterating the $\mathsf{FHE}.\mathsf{GateEval}_{\mathsf{pk}}$ algorithm many times, one can generically build a *circuit* evaluation algorithm:

FHE.Eval_{ek1,...,ekd} $(f, c_1, \ldots, c_n) \to c^*$. This is a deterministic algorithm that takes as input a public key pk, a circuit representing a function $f: \{0, 1\}^n \to \{0, 1\}$ and n ciphertexts $c_1, \ldots, c_n \in \{0, 1\}^{\ell}$. It outputs a ciphertext $c^* \in \{0, 1\}^{\ell}$.

Definition 3.4 (FHE). A leveled, gate-by-gate fully homomorphic encryption scheme

FHE = (FHE.Setup, FHE.Enc, FHE.Dec, FHE.GateEval)

is required to satisfy the following properties:

- **Encryption Correctness.** For any choice of (pk, sk_0) in the support of FHE.Setup $(1^n, 1^d)$, any $b \in \{0, 1\}$ and any $c \leftarrow \text{FHE.Enc}_{pk}(b)$ we have $\text{FHE.Dec}_{sk_0}(c) = b$.
- Honest Evaluation Correctness. For any choice of $(\mathsf{pk}, \mathsf{ek}_1, \ldots, \mathsf{ek}_d, \mathsf{sk}_0, \ldots, \mathsf{sk}_d) \leftarrow \mathsf{FHE}.\mathsf{Setup}(1^n, 1^d)$, any honestly generated ciphertexts $c_1, \ldots, c_n \in \{0, 1\}^\ell$ such that $c_i = \mathsf{FHE}.\mathsf{Enc}_{\mathsf{pk}}(b_i)$, and any layered circuit $f: \{0, 1\}^n \to \{0, 1\}$ of depth d, if we set $c = \mathsf{FHE}.\mathsf{Eval}_{\mathsf{ek}_1, \ldots, \mathsf{ek}_d}(f, c_1, \ldots, c_n)$ then $\mathsf{FHE}.\mathsf{Dec}_{\mathsf{sk}_d}(c) = f(b_1, \ldots, b_n)$.
- **Malicious Gate Correctness.** For any choice of $(\mathsf{pk}, \mathsf{ek}_1, \dots, \mathsf{ek}_d, \mathsf{sk}_0, \dots, \mathsf{sk}_d) \leftarrow \mathsf{FHE}.\mathsf{Setup}(1^n)$, any index $0 \leq i \leq d-1$, any $f : \{0,1\}^2 \rightarrow \{0,1\}$, and any ciphertexts $c_1, c_2 \in \{0,1\}^\ell$, if $\mathsf{Dec}_{\mathsf{sk}_i}(c_1) = b_1$ and $\mathsf{Dec}_{\mathsf{sk}_i}(c_2) = b_2$, then $\mathsf{Dec}_{\mathsf{sk}_{i+1}}(\mathsf{FHE}.\mathsf{GateEval}(f, \mathsf{ek}_{i+1}, c_1, c_2)) = f(b_1, b_2)$.

Security. The encryption scheme is semantically secure in the presence of $(\mathsf{pk}, \mathsf{ek}_1, \ldots, \mathsf{ek}_d)$.

Theorem 3.5 ([BV11]). Assuming the hardness of Learning with Errors [Reg05], there exists a leveled, gate-by-gate homomorphic encryption scheme.

Specifically, the properties we require in Definition 3.4 all follow from the (leveled) *bootstrapping* construction of [Gen09], provided that the "base" somewhat homomorphic encryption scheme satisfies perfect correctness. Moreover, the public keys and ciphertexts in this LWE-based scheme will be pseudorandom under the LWE assumption.

Remark 3.6. Given a FHE scheme, one can extend the definition of the encryption algorithm FHE.Enc to take as input a longer message $m \in \{0,1\}^n$, as opposed to a single bit. FHE.Enc_{pk}(m)will simply run $c_i = \text{FHE.Enc}_{pk}(m_i)$ for every $i \in [n]$, and set $c = (c_1, \ldots, c_n) \in \{0,1\}^{n \cdot \ell}$. Similarly, we extend FHE.Eval to take as input a function $f : \{0,1\}^n \to \{0,1\}^u$ with multi-bit output.

3.3 Succinct Non-Interactive Arguments

We define succinct non-interactive argument systems (SNARG) [Mic94, GW11, BCCT13] for the computation of a non-deterministic polynomial-time Turing machine \mathcal{M} . Our definition allows for a long common reference string, maintaining verifier efficiency by requiring that the $crs = (crs_{\mathcal{P}}, crs_{\mathcal{V}})$ be separated into prover and verifier components (where the verifier component is required to be short).

A SNARG system for \mathcal{M} consists of polynomial-time algorithms (Gen, \mathcal{P}, \mathcal{V}) with the following syntax:

- The randomized setup algorithm Gen takes as input a security parameter $\lambda \in \mathbb{N}$ and an input length n, both in unary, and outputs a pair of common reference strings $\operatorname{crs} = (\operatorname{crs}_{\mathcal{P}}, \operatorname{crs}_{\mathcal{V}})$.
- The prover algorithm \mathcal{P} takes as input the prover reference string $\operatorname{crs}_{\mathcal{P}}$, an input $x \in \{0,1\}^n$ and its associated witness $w \in \{0,1\}^m$, and outputs a proof π .
- The verifier algorithm \mathcal{V} takes as input the verifier reference string $\operatorname{crs}_{\mathcal{V}}$, an input $x \in \{0, 1\}^n$ and a proof π . It outputs a bit indicating if it accepts or rejects.

Definition 3.7. A triple of algorithms (Gen, \mathcal{P}, \mathcal{V}) is a SNARG system for a Turing machine \mathcal{M} if the following hold:

Completeness. For every $\lambda, n \in \mathbb{N}$ and every $x \in \{0, 1\}^n$ and $w \in \{0, 1\}^m$ such that $\mathcal{M}(x, w) = 1$,

$$\Pr\left[\begin{array}{c} \mathcal{V}(\mathsf{crs}_{\mathcal{V}}, x, \pi) = 1 : \frac{(\mathsf{crs}_{\mathcal{P}}, \mathsf{crs}_{\mathcal{V}}) \leftarrow \mathsf{Gen}(1^{\lambda}, 1^{n}),}{\pi \leftarrow \mathcal{P}(\mathsf{crs}_{\mathcal{P}}, x, w)} \right] = 1.$$

- **Efficiency.** The length of $\operatorname{crs}_{\mathcal{V}}$ is $\operatorname{poly}(\lambda, \log n, \log m)$. The length of a proof π is also $\operatorname{poly}(\lambda, \log n, \log m)$. The runtime of \mathcal{V} is $\operatorname{poly}(|\operatorname{crs}_{\mathcal{V}}|, |\pi|) + \operatorname{poly}(\lambda, \log n, \log m) \cdot n$.
- **Non-adaptive Soundness.** For every $x \in \{0,1\}^n$ where $x \notin \mathcal{L}_M$ and every poly-size adversary Adv, there is a negligible function μ such that

$$\Pr\left[\begin{array}{c} \mathcal{V}(\mathsf{crs}_{\mathcal{V}}, x, \pi^*) = 1 \\ \pi^* \leftarrow \mathsf{Adv}(\mathsf{crs}_{\mathcal{P}}, \mathsf{crs}_{\mathcal{V}}, x) \end{array} \right] \leq \mu(\lambda)$$

Adaptive Soundness. For every poly-size adversary Adv, there is a negligible function μ such that

$$\Pr\left[\begin{array}{c} x^* \notin \mathcal{L}_{\mathcal{M}} \wedge \mathcal{V}(\mathsf{crs}_{\mathcal{V}}, x^*, \pi^*) = 1 & : \begin{array}{c} (\mathsf{crs}_{\mathcal{P}}, \mathsf{crs}_{\mathcal{V}}) \leftarrow \mathsf{Gen}(1^{\lambda}, 1^n), \\ (x^*, \pi^*) \leftarrow \mathsf{Adv}(\mathsf{crs}_{\mathcal{P}}, \mathsf{crs}_{\mathcal{V}}) \end{array}\right] \leq \mu(\lambda).$$

An additional efficiency parameter of interest is the length of $\operatorname{crs}_{\mathcal{P}}$; we do not impose any explicit requirements on it, but seek to minimize the length of $\operatorname{crs}_{\mathcal{P}}$ when possible.

Remark 3.8 (Transparent Setup). The SNARGs constructed in this paper can be made to satisfy an additional property called **transparent setup**: the non-adaptive soundness notion above holds even if Adv is given the random coins used by Gen(·). All of our SNARGs can be made to satisfy this property because in all of our constructions, the algorithm Gen $(1^{\lambda}, 1^{n})$ generates a polynomial number of ciphertexts for an LWE-based encryption scheme along with a hash key hk, and outputs these strings (for the prover) along with a tree hash of the ciphertexts (for the verifier). Since the encryption scheme we use has pseudorandom ciphertexts and the hash family may be chosen to have uniformly random keys, we may modify the Gen(·) algorithm to simply output a long, uniformly random string along with a tree hash of this string (this modification preserves non-adaptive soundness). This modified Gen algorithm has no private randomness and thus these SNARGs have transparent setup.

3.4 Batch Arguments (BARGs)

The following preliminaries on BARGs are due to [BBK⁺23].

A batch argument system BARG for an NP language \mathcal{L} enables proving that k NP statements are true with communication cost that is polylogarithmic in k. There are many BARG variants which are known to be existentially equivalent under mild computational assumptions (see, e.g., [CJJ22, KVZ21, KLVW23]). In this work, for simplicity in our constructions, we make use of an argument system for what we call "batch index Turing machine SAT" (BatchTMSAT), defined below.

Definition 3.9. The language BatchIndexTMSAT consists of instances of the form x = (M, z, k, T), where:

- M is the description of a Turing machine.
- z is an input string (to M)

- k is a batch size, and
- T is a running time.

An instance x = (M, z, k, T) is in BatchIndexTMSAT if for all $i \le i \le k$, there exists a string w_i such that $M(z, i, w_i)$ accepts within T steps.

We sometimes use the notation $\mathcal{R}(x, i, w_i)$ to denote the relation with instance (x, i) and corresponding witness w_i .

Syntax. A (publicly verifiable and non-interactive) batch argument system BARG for BatchIndexTMSAT consists of the following polynomial time algorithms:

- $\operatorname{\mathsf{Gen}}(1^{\lambda}, 1^n, 1^m, k) \to \operatorname{\mathsf{crs.}}$ This is a probabilistic polynomial-time algorithm that takes as input a security parameter 1^{λ} , input length 1^n , witness length 1^m , and a parameter k. It outputs a common reference string crs.
- $\mathcal{P}(\mathsf{crs}, M, z, 1^T, w_1, \ldots, w_k) \to \pi$. This deterministic polynomial-time algorithm takes as input crs , Turing machine M, input z, runtime 1^T , and k witnesses w_1, \ldots, w_k . It outputs a proof π .
- $\mathcal{V}(\operatorname{crs}, x, \pi) \to 0/1$. This deterministic polynomial-time algorithm takes as input a crs, instance x = (M, z, k, T), and a proof π . It outputs a bit (1 to accept, 0 to reject).

Definition 3.10 (BARG). A batch argument system BARG = (Gen, P, V) for BatchIndexTMSAT is required to satisfy the following properties:

Completeness. For any $n \in \mathbb{N}$, any $k(n), n(n), m(\lambda), T(\lambda) \leq 2^n$, any instance $x = (M, z, k, T) \in$ BatchIndexTMSAT with |M| + |z| = n, and any corresponding witnesses $w_1, \ldots, w_k \in \{0, 1\}^m$,

$$\Pr\left[\begin{array}{c} \mathcal{V}(\mathsf{crs}, x, \pi) = 1 : \begin{array}{c} (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^n, 1^n, 1^m, i^*), \\ \pi \leftarrow \mathcal{P}(\mathsf{crs}, M, z, 1^T, w_1, \dots, w_k) \end{array} \right] = 1.$$

Adaptive Soundness. For any poly-size adversary \mathcal{A} and any polynomials $k(\lambda), n(\lambda), m(\lambda), T(\lambda)$ there exists a negligible function $\operatorname{negl}(\cdot)$ such that for every $n \in \mathbb{N}$,

$$\Pr\left[\begin{array}{cc} \mathcal{V}(\mathsf{crs}, x, \pi) = 1 & \mathsf{crs} \leftarrow \mathsf{Gen}(1^{\lambda}, 1^{n}, 1^{m}, k) \\ \wedge (M, z, k, T) \not\in \mathsf{BatchIndexTMSAT} & (M, z, \pi) = \mathcal{A}(\mathsf{crs}) \\ & w^* \leftarrow \mathsf{Extract}(\mathsf{td}, \pi) \end{array}\right] \leq \mathsf{negl}(n).$$

Efficiency. In the completeness experiment above, $|crs| + |\pi| \le m \cdot poly(n, log(knT))$. The running time of the verifier is at most $poly(|crs| + |\pi|) + poly(n) \cdot |x|$.

Throughout this paper, when we refer to a BARG we implicitly mean a BARG for BatchIndexTMSAT.

Theorem 3.11 ([CJJ22, WW22, HJKS22, KLVW23]). *There exists an* BARG *for* BatchIndexTMSAT *assuming* LWE *or* DLIN *or (subexponential* DDH *and* QR).¹⁹

¹⁹The work [CGJ⁺23] also constructs a BARGs based on subexponential DDH, but not with efficiency parameters matching our definition.

3.5 Propositional Logic Systems

This section of the preliminaries is due to [JJ22].

3.5.1 Extended Frege

We use extended Frege systems (denote as \mathcal{EF}) for propositional logic. Such a system is described by a set of *variables*, a set of *connectives*, and a set of *inference rules*. *Variables* are the most basic elements, usually represented by letters such as x, y, z. *Connectives* are used to connect variables. We only use two connectives \rightarrow and \neg for "imply" and "negation", respectively. Other connectives such as \land, \lor, \oplus for "and", "or", "xor", can be defined using \rightarrow and \neg . We use \leftrightarrow to denote "if and only if", and formally, $a \leftrightarrow b$ is the abbreviation of $(a \rightarrow b) \land (b \rightarrow a)$. We use F to represent "false", and define "true" T as \neg F.

Formulas are defined inductively: F ("false") is a formula; any variable is a formula; if u, v are formulas, then $u \to v$, $\neg u$ are formulas. A formula can be treated as a labeled tree, where leaves are labeled with variables, and internal nodes are labeled with connectives. A subformula of A is defined as a subtree of A. We define the following complexity measures of formulas. For each formula A, we define the size of A as the number of nodes in the tree.

A substitution σ is a map from the set of variables to the set of formulas. If A is a formula, then the result of applying σ to A is denoted as $A\sigma$, which is a formula obtained by replacing each occurrence of the variables in A by its image under σ . For example, let $A = p \rightarrow (q \rightarrow p)$ and let a substitution σ be $p \mapsto a \land b, q \mapsto a \lor b$, then $A\sigma = (a \land b) \rightarrow ((a \lor b) \rightarrow (a \land b))$.

A Frege system is specified by a set of inference rules. Each inference rule is defined as $A_1, A_2, \ldots, A_k \vdash A_0$, where A_0, A_1, \ldots, A_k are formulas. Intuitively, it means that "if A_1, A_2, \ldots, A_k are valid, then A_0 is also valid". If k = 0, then we say such an inference rule is an axiom.

In this work, we use the following set of axioms and modus ponens as inference rules for propositional logic.

- Axiom 1: $p \rightarrow (q \rightarrow p)$
- Axiom 2: $(p \to (q \to r)) \to ((p \to q) \to (p \to r))$
- Axiom 3: $\neg \neg p \rightarrow p$
- Modus Ponens: $p, p \rightarrow q \vdash q$

We use the notation " $p_1, p_2, \ldots, p_n \vdash q$ to denote that q" is the *logical consequence* of p_1, \ldots, p_n . We refer to p_1, p_2, \ldots, p_n as the *premise* and refer to q as the *conclusion*. We say q is a logical consequence of p_1, \ldots, p_n if there exists a *derivation*, which is a series of formulas $\theta_1, \theta_2, \ldots, \theta_\ell$ with $\theta_\ell = q$, and for each $i \in [\ell], \theta_i$ is either

- A premise p_j with $j \in [k]$, or
- $A_0\sigma$, where $A_1, A_2, \ldots, A_k \vdash A_0$ is an inference rule, σ is a substitution, and $\{A_1\sigma, A_2\sigma, \ldots, A_k\sigma\}$ are a subset of the formulas $\{\theta_1, \theta_2, \ldots, \theta_{i-1}\}$.

A *proof* is a derivation with no premise.

The extended Frege system denoted as \mathcal{EF} is a logic system that additionally has the following extension axioms. Namely, for a derivation $(\theta_1, \theta_2, \ldots, \theta_\ell)$ in \mathcal{EF} , for each $i \in [\ell]$, θ_i needs to satisfy the aforementioned constraint or θ_i is of the form $t \leftrightarrow A$, where A is a formula, and t is a new variable that has not occurred in $\theta_1, \theta_2, \ldots, \theta_{i-1}$, and also does not occur in A.

Size of \mathcal{EF} Proofs. We define the *size* of a derivation $(\theta_1, \theta_2, \ldots, \theta_\ell)$ as the summation of the sizes of the formulas $\theta_1, \theta_2, \ldots, \theta_\ell$.

Remark 3.12. First note that since the axioms and Modus Ponens are of constant size, each line in the proof (i.e., each formula θ_i) depends on only a constant number of previous lines. Second, we can assume without loss of generality that each line of the proof is of constant size by using the extension axiom.

Definition 3.13. For every circuit C we define $\operatorname{Prop}[C]$ to be the set of formulas that represent each gate in C. Namely, denoting by G the set of all gates in C, $\operatorname{Prop}[C] \triangleq \{\varphi_g\}_{g \in G}$, where $\varphi_g = o \leftrightarrow g(l, r)$, where l and r are the input wires to g and o is the output wire o.

Definition 3.14. For any two Boolean circuits C and D, we define

 $C \rightarrow D$

as a shorthand for

$$\mathsf{Prop}[C] \cup \mathsf{Prop}[D] \vdash w_c \to w_d,$$

where w_c, w_d are the output wires of C, D, respectively.

Definition 3.15. We say that a circuit E extends a circuit C if the circuit E contains C as a subcircuit, and the input wires of E are identical to those of C. In other words, E extends C if E can be constructing by adding to C additional (non-input) wires and gates,

Polynomial-length \mathcal{EF} **proofs for arithmetic.** The work of [Bus86] shows that basic arithmetic operations such as $+, -, \cdot, /, \lfloor \cdot \rfloor, \leq$ can be introduced in Cook's theory PV [Coo75] as function symbols, and their basic laws such as commutative law, associative law, distributive law, etc. can be proven in PV. From Cook's propositional translation [Coo75], there exist polynomial-size \mathcal{EF} proofs for them.

3.5.2 Cook's Theory PV

This section of the preliminaries is due to [JJ22].

Cook introduced a theory PV [Coo75] to capture the intuition of feasibly constructive proofs (i.e. polynomial-time reasoning). PV is an equational theory, i.e., each statement in PV asserts that two terms are equal. Moreover, it allows the introduction of new function symbols by recursive definition (i.e. Cobham's definition of polynomial-time functions [Cob65]). Hence, any polynomialtime function is definable in PV [Coo75]. Moreover, common used arithmetical operations such as addition, multiplication, and modulus functions can also be defined in PV. Their related properties such as commutative law, associative law etc. can be proven in PV [Bus86].

Formally, Cook's theory PV [Coo75] is defined as follows. PV works on the natural numbers that are represented in the dyadic notation, where any natural number x is uniquely represented as a finite string of integers in $\{1,2\}^*$. Specifically, we represent x as the string $x_n x_{n-1} x_{n-2} \dots x_1 \in \{1,2\}^n$, if $\sum_{i=1}^n x_i 2^i = x$, and use an empty string to represent 0. It's easy to see that such presentation is unique for any natural number. The function $s_i(x) = 2x + i$, i = 1, 2 appends i to the string x. Hence, we also denote $s_i(x)$ as $x \parallel i$. We introduce the following terminologies. Terms are defined inductively as follows: any variable is a term; any function symbol of arity 0 is a term; if t_1, t_2, \ldots, t_k are terms, and f is a function symbol, then $f(t_1, t_2, \ldots, t_k)$ is a term. Equations are of the form t = u, where both t and uare terms. A derivation for the statement $E_1, E_2, \ldots, E_n \vdash_{PV} E$ in PV is a series of equations D_1, D_2, \ldots, D_ℓ such that $D_\ell = E$ and for any $i \in [\ell]$, the equation D_i is either a premise $E_j(j \in [n])$, or a defining equation for some function symbol that we will introduce later, or follows from some inference rule that we will introduce later. A proof in PV is a derivation with no premise (n = 0).

Introducing Function Symbols. A new function symbol f can be introduced in PV in the following two ways. The first way is to define

$$f(x_1, x_2, \dots x_k) = t,$$

where t is a term with variables x_1, x_2, \ldots, x_k .

The second way is to recursively define the function on the dyadic notion (i.e. Cobham's characterization of polynomial-time functions [Cob65]). Specifically, for existing function symbols g, h_1, h_2, k_1, k_2 in PV, define the following equations as defining equations

$$f(0, \mathbf{y}) = g(\mathbf{y}), \quad f(x||i, \mathbf{y}) = h_i(x, \mathbf{y}, f(x, \mathbf{y})), i = 1, 2,$$
(2)

where $\mathbf{y} = (y_1, \ldots, y_k)$ is a series of k variables. Then how f is computed for any x, \mathbf{y} is fully specified. PV further requires that the f can be computed in polynomial time. To ensure this, Cook requires that " $|h_i(x, \mathbf{y}, z)| \leq |z| + |k_i(x, \mathbf{y})|$ " is provable in PV, where $|\cdot|$ is the length of the dyadic presentation. To achieve this, Cook introduced the LESS function, and it is defined with other *initial functions* as follows. $s_i, i = 1, 2$ has no defining functions. 0 is also function symbol with arity 0, and has no defining function.

- TR: TR(0) = 0, TR(x||i) = x, i = 1, 2. It cuts off the least significant digit in the dyadic notion.
- \star : $\star(x,0) = x, \star(x,y||i) = s_i(x,y), i = 1, 2$. It concatenates the string x and y.
- \circledast : $\circledast(x,0) = x, \circledast(x,y||i) = \star(x, \circledast(x,y)), i = 1, 2$. It concatenates |y| copies of x.
- LESS : LESS(x, 0) = x, LESS(x, y||i) = TR(LESS(x, y)), i = 1, 2. It cuts off the |y| right most digits of x in the dyadic notion. Then we can use LESS(x, y) = 0 to express $|x| \le |y|$.

To complete the definition of function f, PV requires two proofs π_1, π_2 in PV for $\mathsf{LESS}(h_i(x, \mathbf{y}, z), z \star k_i(x, \mathbf{y})) = 0, i = 1, 2$. Then a function symbol f is defined as the tuple $(g, h_1, h_2, k_1, k_2, \pi_1, \pi_2)$.

The *inference rules* are in the following. Here, t, u, v are any terms, x is any variable, and $\mathbf{y} = (y_1, y_2, \ldots, y_k)$ is any tuple of $k \ge 0$ variables. f is any function symbol (we will define later).

- R_1 : $t = u \vdash u = t$.
- R_2 : $t = u, u = v \vdash t = u$
- $R_3: t_1 = u_1, t_2 = u_2, \dots, t_k = t_k \vdash f(t_1, t_2, \dots, t_k) = f(u_1, u_2, \dots, u_k).$
- R_4 : $t = u \vdash t(v/x) = u(v/x)$. Here, the notation "t(v/x)" means replacing each occurrence of the variable x with the term v. "u(v/x)" is defined in the same way.
- $R_5: E_1, E_2, \ldots, E_6 \vdash f_1(x, \mathbf{y}) = f_2(x, \mathbf{y})$, where E_1, E_2, \ldots, E_6 are the defining equations 2 for f_1, f_2 , with the same function symbols g, h_1, h_2 .

Propositional Translation. In the same work [Coo75], Cook showed that any proofs in PV can be translated to polynomial size propositional logic proofs. The original theorem statement uses *extended resolutions* logic. Later [CR79] showed that extended resolution and extended Frege system are essentially equivalent in terms of proof size. For simplicity, we use extended Frege system in this work, and state Cook's result in extended Frege system.

Before we formally state the theorem, we first describe how to transform a theorem statement in PV to proposition logic. The idea is to use variables in \mathcal{EF} to present each digit in the dyadic notation. Specifically, let m be an integer. For each term t in PV, let $P_0[t], P_1[t], \ldots, P_m[t]$ and $Q_0[t], Q_1[t], \ldots, Q_m[t]$ be a set of variables in \mathcal{EF} . For each $i \in [m]$, use $Q_i[t]$ to indicate whether thas *i*-th digit, and use $P_i[t]$ to indicate the *i*-th digit of t, i.e.

$$Q_i[x] = \begin{cases} \mathsf{T}, \text{if } t \ge 2^{i+1} - 1\\ \mathsf{F}, \text{otherwise} \end{cases} \quad P_i[x] = \begin{cases} \mathsf{T}, \text{if the } i\text{-th dyadic digit of } t \text{ is } 2\\ \mathsf{F}, \text{otherwise} \end{cases}$$

For the easy of representation, in this work we use the following notation $\operatorname{Var}_m[t]$ to denote the variables $\{P_i[t], Q_i[t]\}_{i=1}^m$ corresponds to t. For each term t, one can associate it with a proposition formula $\operatorname{prop}_m[t]$, asserting $\operatorname{Var}_m[t]$ is computed correctly from the variables $\operatorname{Var}_m[p_1], \ldots, \operatorname{Var}_m[p_k]$, where p_1, p_2, \ldots, p_k are all variables appear in t. For any variable x, $\operatorname{prop}_m[x]$ is the formula asserting $\operatorname{Var}_m[x]$ is well-formed, i.e. $\neg Q_i[x]$ implies $\neg Q_{i+1}[x]$ for $i \in [m-1]$. The definition of $\operatorname{prop}_m[t]$ can be inductively defined for any term t. For more details, see [Coo75].

For any integer n, if the computation of all terms in the proof only needs m dyadic digits, then m is called a *bounding value*. For any equation t = u, where t and u are both terms. $[t = u]_m^n$ is defined as the propositional formula asserting that if the variables in t are all less than n digit, then the value of t and u are equal. For its formal definition, see [Coo75].

Next, we present the theorem statement for Cook's propositional translation.

Theorem 3.16 (Corollary of ER Simulation Theorem in [Coo75]). For any two terms t and u, and any n and any polynomial bounding value m = m(n), if $\vdash_{PV} t = u$, then $[t = u]_m^n$ has polynomial size logic proofs in extended Frege logic.

The idea of Theorem 3.16 is to do an induction on the length of the proofs in PV, and translate each step of the proof in PV to a polynomial size proof in the extended Frege.

3.6 Local Assignment Generators

In this section we define the notion of a local assignment generator as defined in [PR17]. We also define an adaptive version as defined in [BHK17].

Definition 3.17 (Local Assignment Generator). Let $C : \{0,1\}^n \to \{0,1\}^m$ denote a Boolean circuit of size s where $n = n(\lambda), m = m(\lambda)$ and $s = s(\lambda)$. For $\ell = \ell(\lambda)$, we say that C has an ℓ -local assignment generator if there is a polynomial time (non-uniform and randomized) algorithm LocalGen = LocalGen_{λ} with the following properties.

• Syntax: The input to LocalGen is the description of a subset of wires $T \subseteq [s]$ of size at most ℓ and its output is an assignment $(\sigma_i)_{i \in T} \in \{0, 1\}^T$ to the corresponding wires of C.

• Local Consistency: for any $\lambda \in \mathbb{N}$ and any gate $g_i = (i, j, k, f)$ of C connecting input wires j, k to an output wire i,

$$\Pr\left[\sigma_i \neq f(\sigma_j, \sigma_k) : (\sigma_i, \sigma_j, \sigma_k) \leftarrow \mathsf{LocalGen}(\{i, j, k\})\right] = \mathsf{negl}(\lambda).$$

where the probability is over the randomness of LocalGen.

• Computational Non-Signaling: for any $\lambda \in \mathbb{N}$ and any sets $T_0, T_1 \subseteq [s]$ of size at most ℓ , the following distributions are computationally indistinguishable:

 $\left((\sigma_i)_{i\in T_0\cap T_1}: (\sigma_i)_{i\in T_0} \leftarrow \mathsf{LocalGen}(T_0)\right) \approx \left((\sigma_i)_{i\in T_0\cap T_1}: (\sigma_i)_{i\in T_1} \leftarrow \mathsf{LocalGen}(T_1)\right)$

Definition 3.18 (Adaptive Local Assignment Generator). Let $C : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ denote a Boolean circuit of size s where $n = n(\lambda), m = m(\lambda)$ and $s = s(\lambda)$. We think of the first n input bits to C as specifying an instance and the last m input bits as specifying a witness. For $\ell = \ell(\lambda)$, we say that C has an adaptive ℓ -local assignment generator if there is a polynomial time (non-uniform and randomized) algorithm LocalGen^{*} = LocalGen^{*} with the following properties.

- Syntax: The input to LocalGen^{*} is the description of a subset of wires $T \subseteq [s]$ of size at most ℓ and its output is an instance $x \in \{0,1\}^n$ and an assignment $(\sigma_i)_{i \in T} \in \{0,1\}^T$ to the corresponding wires of C.
- Local Consistency: For any $\lambda \in \mathbb{N}$ and any input wire $i \in [n]$ (corresponding to the *i*'th wire of the instance) it holds that

 $\Pr[\sigma_i \neq x_i : (x, \sigma_i) \leftarrow \mathsf{LocalGen}^*(\{i\})] = \mathsf{negl}(\lambda)$

and for any gate $g_i = (i, j, k, f)$ of C connecting input wires j, k to an output wire i,

$$\Pr\left[\sigma_i \neq f(\sigma_j, \sigma_k) : (x, (\sigma_i, \sigma_j, \sigma_k)) \leftarrow \mathsf{LocalGen}^*(\{i, j, k\})\right] = \mathsf{negl}(\lambda),$$

where both probabilities are over the randomness of LocalGen^{*}.

• Computational Non-Signaling: for any $\lambda \in \mathbb{N}$ and any sets $T_0, T_1 \subseteq [s]$ of size at most ℓ , the following distributions are computationally indistinguishable:

$$\left(x, (\sigma_i)_{i \in T_0 \cap T_1}\right) : (x, (\sigma_i)_{i \in T_0}) \leftarrow \mathsf{LocalGen}^*(T_0)\right) \approx \left((x, (\sigma_i)_{i \in T_0 \cap T_1}) : (x, (\sigma_i)_{i \in T_1}) \leftarrow \mathsf{LocalGen}^*(T_1)\right)$$

In our soundness proofs (both in the non-adaptive and adaptive settings) rely on the following lemma, which was proven in [KRR14] in the non-adaptive setting and in [BHK17] in the adaptive setting (stated below).

Lemma 3.19. [KRR14, BHK17] There exists a parameter $\ell(s) = \text{polylog}(s)$ and a function ECC computable in polynomial time that takes an input an circuit $C : \{0,1\}^n \to \{0,1\}$ of size s and depth d, and outputs an extension ECC(C) of C of size $s \cdot \text{polylog}(s)$ and depth $d \cdot \text{polylog}(s)$ such that for any adaptive ℓ -local assignment generator LocalGen^{*} (Definition 3.18) for the circuit ECC(C), denoting by out its output wire,

$$\Pr[\mathsf{LocalGen}^*(\mathsf{out}) = (x, b) \land b \neq C(x)] = \mathsf{negl}(\lambda).$$

3.7 Relevant Theorems based on [JKLV24]

In this work, we use several theorems based on the work [JKLV24]. Subsequently, [JKLV24] updated their paper to include formulations of Theorems 3.20 and 3.22 (for readability) because they follow immediately from the proofs of similar theorems in [JKLV24]. However, the particular formulations of the theorems that follow were introduced in this work.

To give context for the first theorem (Theorem 3.20) we use, we start by noting that if we construct a SNARG using the hash-and-BARG paradigm [CJJ22, KVZ21] the soundness guarantee we obtain is that one can convert any (efficient) cheating prover \mathcal{P}^* into an (efficient) local assignment generator LocalGen on the underlying verification circuit, and if the prover is adaptive (i.e., chooses the instance based on the crs) then LocalGen is also adaptive.

The work of [JKLV24] extended this, and implicitly showed (for an appropriate "Encrypt-Hashand-BARG" construction) how one can use any (efficient) cheating prover \mathcal{P}^* to construct an (efficient) "universal" local assignment generator LocalGen for any extension of the verification circuit up to certain size and depth bounds. Namely, [JKLV24] showed that if the SNARG is constructed using the encrypt-hash-and-BARG paradigm, then for any parameters T, d, D (that are chosen in the SNARG scheme) one can convert any (efficient) cheating prover \mathcal{P}^* into an (efficient) universal local assignment generator LocalGen that takes as input a circuit E that extends the verification circuit, where E is of size T, depth d, and can be uniformly generated with advice of size D, and has the guarantee that LocalGen (E, \cdot) is a local assignment generator. Though the focus of the work of [JKLV24] was on the non-adaptive setting (since their applications were inherently limited to the non-adaptive setting), this result holds (with identical proofs) in both the non-adaptive and adaptive settings.

In what follows, we consider circuits whose size depends both on the instance size n and the security parameter λ . The reason for this is that in our work, the size of the extended circuit below may depend on the security parameter (in addition to the instance size).

Theorem 3.20 ([JKLV24] Theorem 5.7). Assume the existence of a leveled, gate-by-gate FHE scheme (Definition 3.4) and a BARG scheme (Definition 3.10). Fix any NP language $\mathcal{L} \subseteq \{0,1\}^*$. Then for any parameters $T = T(\lambda, n)$, $d = d(\lambda, n)$, $D = D(\lambda, n)$, and $\ell = \ell(\lambda, n)$, such that T is larger than d, D, ℓ and is at least as large as the NP verification circuit, there exists a SNARG scheme (Gen, \mathcal{P}, \mathcal{V}) with the following properties:

- The crs is of size $(\ell + d + D) \cdot \operatorname{poly}(\log T, \lambda)$.
- The proof is of size $\ell \cdot \operatorname{poly}(\log T, \lambda)$.
- The prover runtime is $poly(T, \lambda)$.
- The verifier runtime is $\tilde{O}(n) + \operatorname{poly}(D, \ell, \log T, \lambda)$.
- It satisfies the following **non-adaptive** soundness guarantee:

For every $x \in \{0,1\}^n$ and every poly-size \mathcal{P}^* that generates an accepting SNARG proof for the statement " $x \in \mathcal{L}$ " with non-negligible probability, there exists a poly-size algorithm LocalGen = LocalGen_{λ} that satisfies the following two conditions:

1. For every depth d and size T extension E of the circuit C_x , that can be generated by a T-time uniform Turing machine with advice (aux, x), where aux is any advice of size D,

 $LocalGen(E, \cdot)$ is an ℓ -local assignment generator for E such that

$$\Pr[\mathsf{LocalGen}(E, \{\mathsf{out}\}) = 0] = \mathsf{negl}(\lambda),$$

where out denotes the output of C_x .

2. For every E as above and every set of wires S in E of size $\leq \ell$, consider the sub-circuit E_S extending C_x , and in addition, it only contains the wires needed to compute the wires in S. Then

$$LocalGen(E, S) \approx LocalGen(E_S, S)$$
(3)

- It satisfies the following adaptive soundness guarantees:²⁰ For every poly-size P* that given crs generates an instance x and an accepting SNARG proof for the statement "x ∈ L" with non-negligible probability, there exists a poly-size algorithm LocalGen* = LocalGen^{*}_λ that satisfies the following two conditions:
 - For every depth d and size T extension E of the circuit C_x, that can be generated by a T-time uniform Turing machine with advice (aux, x), where aux is any advice of size D, LocalGen*(E, ·) is an adaptive ℓ-local assignment generator for E, such that

$$(x, b) \leftarrow \mathsf{LocalGen}^*(E, {\mathsf{out}})$$

satisfies:

- (a) $\Pr[b=0] = \operatorname{negl}(\lambda)$.
- (b) the distribution of x is computationally indistinguishable from the distribution of x generated by $(x, \pi) \leftarrow P^*(crs)$ conditioned on $\mathcal{V}(crs, x, \pi) = 1$, where $crs \leftarrow Gen(1^{\lambda})$.
- 2. For every depth d and size T extension E of the circuit C_x , that can be generated by a T-time uniform Turing machine with advice (aux, x), where aux is any advice of size D, and for every set of wires S in E of size $\leq \ell$, consider the sub-circuit E_S extending C_x , and in addition, it only contains C_x and the wires needed to compute the wires in S. Then

$$\mathsf{LocalGen}^*(E,S) \approx \mathsf{LocalGen}^*(E_S,S) \tag{4}$$

Moreover, if we assume the existence of a (non-leveled) FHE then we can omit the depth restriction from the extended circuit and from the efficiency guarantees.

Remark 3.21. From now on we refer to an (adaptive/non-adaptive) ℓ -local assignment generator that satisfies the universality properties (Items 1 and 2) from Theorem 3.20 as an (adaptive/non-adaptive) (T, d, D)-universal ℓ -local assignment generator.

We emphasize that Theorem 3.20 does not argue that the SNARG scheme has the desired soundness guarantee, rather that it has only "local soundness" in the form of a local assignment generator. [JKLV24] uses Theorem 3.20 to argue soundness via the following theorem, which constructs extensions for which "local soundness" implies soundness. We use this theorem in addition to Theorem 3.20.

²⁰[JKLV24] did not define or consider adaptive local assignment generators, but the statement and proof of [JKLV24] extend immediately to the adaptive setting below. Crucially, note that we do *not* require or check that the adaptively chosen input x is not in \mathcal{L} .

Theorem 3.22 ([JKLV24] Theorem 6.3). There exists a polynomial poly such that the following holds. Fix parameters $T = T(n, \lambda)$, $d = d(n, \lambda)$, $D = D(n, \lambda)$, and $L = L(n, \lambda)$, all of size $\leq 2^{\lambda}$. Fix any circuit family $\{C_{x,\lambda}\}_{x \in \{0,1\}^*, \lambda \in \mathbb{N}}$ such that each $C_{x,\lambda}$ is of size $T = T(|x|, \lambda)$ and depth $d = d(|x|, \lambda)$, and can be generated by a (T + L)-time uniform Turing machine with advice $\mathsf{aux} = \mathsf{aux}_{x,\lambda}$. Then if $C_{x,\lambda}$ is unsatisfiable and has an \mathcal{EF} proof for the statement "for all w, $C_{x,\lambda}(w) = 0$ " of length $L(|x|, \lambda)$, and this \mathcal{EF} proof can be generated by a (T + L)-time uniform Turing machine with advice $\mathsf{aux}_{\mathcal{EF}} = \mathsf{aux}_{\mathcal{EF},x,\lambda}$ of length D, then there exists a circuit $E_{x,\lambda}$ that extends $C_{x,\lambda}$, such that $E_{x,\lambda}$ is of size $T' \leq \mathsf{poly}(T + L)$, depth $D' \leq (d + L) \cdot \mathsf{poly}(\lambda)$, can be generated by a T'-time Turing machine with advice $(\mathsf{aux}, \mathsf{aux}_{\mathcal{EF}})$, such that for $\ell = \mathsf{polylog}(T', L')$ and for every (T', d', D')-universal ℓ -local assignment generator LocalGen there exists a negligible function negl such that the following holds:

$$\Pr[\mathsf{LocalGen}(E_{x,\lambda},\mathsf{out})=1] = \mathsf{negl}(\lambda).$$

Finally, we will also use is an adaptive version of Theorem 3.22 whose proof is identical to that of Theorem 3.22.

Theorem 3.23. There exists a polynomial poly such that the following holds. Fix parameters $T = T(n, \lambda)$, $d = d(n, \lambda)$, $D = D(n, \lambda)$, and $L = L(n, \lambda)$, all of size $\leq 2^{\lambda}$. Fix any circuit family $\{C_{n,\lambda}\}_{n\in\mathbb{N}}$, where $C_{n,\lambda}: \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$, where $m = m(n,\lambda)$, such that each $C_{n,\lambda}$ is of size $T = T(n,\lambda)$ and depth $d = d(n,\lambda)$, and can be generated by a (T + L)-time uniform Turing machine with advice aux. Then if for every $n, \lambda \in \mathbb{N}$ there is an \mathcal{EF} proof for the statement: "for all x of length n, for all w of length m, $C_n(x,w) = 0$ " of length $L(|x|,\lambda)$, and this \mathcal{EF} proof can be generated by a (T + L)-time uniform Turing machine with advice $aux_{\mathcal{EF}}$ of length D, then there exists a circuit $E_{n,\lambda}$ that extends $C_{n,\lambda}$, such that $E_{n,\lambda}$ is of size $T' \leq poly(T + L)$, depth $D' \leq (d + L) \cdot poly(\lambda)$, can be generated by a T'-time Turing machine with advice $(aux, aux_{\mathcal{EF}})$, such that for $\ell = polylog(T', L')$ and for every (T', d', D')-universal adaptive ℓ -local assignment generator LocalGen there exists a negligible function negl such that the following holds:

 $\Pr[(x, b) \leftarrow \mathsf{LocalGen}(E_{n,\lambda}, \mathsf{out}) \ s.t. \ b = 1] = \mathsf{negl}(\lambda).$

4 Universal SNARG Construction

4.1 Main Theorem Statement

Theorem 4.1. Assume the existence of a leveled, gate-by-gate FHE scheme (Definition 3.4) and a BARG scheme (Definition 3.10). Fix any NP language $\mathcal{L} \subseteq \{0,1\}^*$ with a corresponding NP relation $\mathcal{R}_{\mathcal{L}}$. For any parameters $T = T(n, \lambda)$, $L = L(n, \lambda)$, and $\ell = \ell(n, \lambda)$, such that $T, L, \ell \leq 2^{\lambda}, T \geq \ell$ and T is larger than the size of NP verification circuit, and L is larger than the depth of the NP verification circuit. We construct a SNARG for \mathcal{L} with the following properties:

- The crs is of length $(L + \ell) \cdot \operatorname{poly}(\lambda)$.
- The proof length is $\ell \cdot \operatorname{poly}(\lambda)$.
- The prover runtime is $poly(T + L, \lambda)$
- The verifier runtime is $\tilde{O}(n) + \text{poly}(L + \ell, \lambda)^{21}$.
- Non-adaptive soundness assuming there exists a 2-message (publicly or privately verifiable) argument system $(\mathcal{P}, \mathcal{V})$ for \mathcal{L} , such that:
 - \mathcal{V} consists of two PPT algorithms $(\mathcal{V}_1, \mathcal{V}_2)$, where \mathcal{V}_1 is given an (inefficiently computable) advice aux about the input x, and generates a message $(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathcal{V}_1(\mathsf{aux}, 1^{\lambda})$ of length L.
 - \mathcal{P} is a time T algorithm that can be implemented by a uniform size T and depth L circuit. It takes as input a pair $(x, w) \in \mathcal{R}_{\mathcal{L}}$ and pk (generated according to $(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathcal{V}_1(\mathsf{aux}, 1^{\lambda})$), and generates a message $\pi = \mathcal{P}(\mathsf{pk}, x, w)$ of length ℓ .
 - $-\mathcal{V}_2$ is a time T algorithm that can be implemented by a uniform size T and depth L circuit. It takes as input a pair (vk, x, π) and outputs a verdict bit b.
 - For every $n, \lambda \in \mathbb{N}$ such that $n \leq 2^{\lambda}$, every $x \in \{0,1\}^n$ and every $(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathcal{V}_1(\mathsf{aux}, 1^{\lambda})$ there is an \mathcal{EF} proof of length L for the statement: "For every w, if $C_x(w) = 1$ then $\mathcal{V}_2(\mathsf{vk}, x, \mathcal{P}(\mathsf{pk}, x, w)) = 1$."

Moreover, if we assume an unleveled gate-by-gate FHE scheme then we can remove the depth bound on \mathcal{P} and \mathcal{V}_2 , remove the assumption that L is larger than the depth of the verification circuit, and replace the restriction that the \mathcal{EF} proof must be of length L with the restriction that it can be generated by a T-time uniform Turing machine with an advice string of size L.

Remark 4.2. We emphasize that in Theorem 4.1 if we assume an unleveled FHE scheme then we can take L to be of size $poly(\lambda)$, and thus obtain a fully succinct SNARG with non-adaptive soundness as long as exists a 2-message argument system $(\mathcal{P}, \mathcal{V})$ for \mathcal{L} where the size of $(pk, vk) \leftarrow \mathcal{V}(1^{\lambda})$ is at most $poly(\lambda)$, the runtime of the prover and verifier is bounded by T, and the completeness of this 2-message argument can be proven via an \mathcal{EF} proof of size T that can be generated by a T-time uniform Turing machine with advice of size $poly(\lambda)$.

4.2 **Proof of Main Theorem**

The proof makes use of the following lemma, which loosely speaking states that for any two circuit families $\{C_{x,\lambda}\}_{x\in\{0,1\}^*,\lambda\in\mathbb{N}}$, and $\{D_{x,\lambda}\}_{x\in\{0,1\}^*,\lambda\in\mathbb{N}}$, where each $D_{x,\lambda}$ takes as input the wires of $C_{x,\lambda}$, if there is a polynomial size \mathcal{EF} proof of the statement $C_{x,\lambda} \to D_{x,\lambda}$ (see Definition 3.14), then there is an extension (see Definition 3.15) of the concatenation of the the circuits $C_{x,\lambda}, D_{x,\lambda}$ such that for any local assignment generator for the extended circuit that extracts the output wires of both $C_{x,\lambda}$ and $D_{x,\lambda}$, if the extracted output wire of $C_{x,\lambda}$ is 1 then the extracted output wire of $D_{x,\lambda}$ must also be 1 with overwhelming probability.

Lemma 4.3. There exists a constant $A \in \mathbb{N}$ and a polynomial p such that the following holds. Let $\{C_{x,\lambda}\}_{x\in\{0,1\}^*,\lambda\in\mathbb{N}}$ and $\{D_{x,\lambda}\}_{x\in\{0,1\}^*,\lambda\in\mathbb{N}}$ be circuit families such that each $D_{x,\lambda}$ takes the wires of $C_{x,\lambda}$ as input, and let $T(n,\lambda)$, $d = d(n,\lambda)$ and $L = L(n,\lambda)$ be polynomials of size $\leq 2^{\lambda}$, that satisfy the following two conditions:

1. The circuit that concatenates $C_{x,\lambda}$ and $D_{x,\lambda}$ is of size $T = T(|x|,\lambda)$ and depth $d = d(|x|,\lambda)$ and can be generated by a (T + L)-time uniform Turing machine with advice $aux = aux_{x,\lambda}$.

²¹We note that the verifier runtime can be generically improved to $poly(n + \lambda + \ell)$ using a RAM delegation scheme as constructed in the works of [CJJ22, KVZ21, KLVW23]. See for example Remark 2.7 in [WW24a] for more details.

2. There exists an \mathcal{EF} proof of $C_{x,\lambda} \to D_{x,\lambda}$ (as defined in Definition 3.14) of length $L = L(|x|, \lambda)$ that can be generated by a (T+L)-time uniform Turing machine with advice $\mathsf{aux}_{\mathcal{EF}} = \mathsf{aux}_{\mathcal{EF},x,\lambda}$.

Then there exists a circuit family $\{E_{x,\lambda}\}_{x\in\{0,1\}^*,\lambda\in\mathbb{N}}$ such that the following holds:

- 1. There exists a polynomial $p = p(\cdot)$ such that each $E_{x,\lambda}$ is of size p(T+L) and depth $(d+L) \cdot p(\lambda)$ and can be generated by a p(T+L)-time uniform Turing machine with advice ($aux, aux_{\mathcal{EF}}$).
- 2. $E_{x,\lambda}$ extends the circuit that concatenates $C_{x,\lambda}$ and $D_{x,\lambda}$.
- 3. For every $(\log(T+L))^A$ -local assignment generator LocalGen for $E_{x,\lambda}$, we have

$$\Pr[c^* = 1 \land d^* = 0] = \mathsf{negl}(\lambda),$$

where the randomness is over $(c^*, d^*) \leftarrow \text{LocalGen}(\{w_c, w_d\})$, and w_c is the output wire of $C_{x,\lambda}$, w_d is the output wire of $D_{x,\lambda}$.

We defer the proof of Lemma 4.3 to the end of this section. We next show how to use Theorem 3.20, along with Lemmas 3.19 and 4.3, to prove Theorem 4.1.

Proof of Theorem 4.1. Fix any NP language $\mathcal{L} \subseteq \{0,1\}^*$ with a corresponding NP relation $\mathcal{R}_{\mathcal{L}}$. For every $x \in \{0,1\}^n$ let C_x be the circuit that takes as input w and outputs 1 if and only if $(x,w) \in \mathcal{R}_{\mathcal{L}}$. Fix parameters $T = T(n,\lambda)$, $L = L(n,\lambda)$ and $\ell = \ell(n,\lambda)$ such that $T, L, \ell \leq 2^{\lambda}$.

Our SNARG construction is exactly the one from Theorem 3.20 with parameters (T^*, d^*, D^*, ℓ^*) where

$$T^* = \operatorname{poly}(T+L), \quad d^* = L \cdot \operatorname{poly}(\lambda), \quad D^* = L \cdot \operatorname{poly}(\lambda) \text{ and } \ell^* = \ell + \operatorname{polylog}(T,L)$$
 (5)

and we define these parameters precisely below. To argue non-adaptive soundness fix any $x \notin \mathcal{L}$ and any poly-size (cheating) prover \mathcal{P}^* that attempts to generate an accepting SNARG proof. We will argue that it will succeed with negligible probability. Theorem 3.20 implies that there exists a PPT algorithm LocalGen that satisfies the conditions written in the theorem statement. We next build an extension circuit $E_{x,\lambda}$ of size T^* and depth d^* that can be described by a Turing machine of size D^* , and show how to use LocalGen (with locality ℓ^*) applied to $E_{x,\lambda}$ to argue that \mathcal{P}^* will be rejected with overwhelming probability.

Extension Circuit $E_{x,\lambda}$. Suppose \mathcal{L} has a 2-message privately verifiable argument system $(\mathcal{P}, \mathcal{V})$ that satisfies the properties from the theorem statement. We denote by $\mathcal{V} = (\mathcal{V}_1, \mathcal{V}_2)$. We sample a random string r uniformly at random and compute $(\mathsf{pk}, \mathsf{vk}) = \mathcal{V}_1(\mathsf{aux}, x, 1^\lambda; r)$. Then we construct an extension circuit $E_{x,\lambda} = E_{x,\mathsf{pk},\mathsf{vk}}$, as follows:

1. Let $D_{x,\mathsf{pk}}^{\mathcal{P}}$ be the circuit that takes as input w and computes $\mathcal{P}(\mathsf{pk}, x, w)$. Denote its output by π .

 $D_{x,\mathsf{pk}}^{\mathcal{P}}$ is a circuit of size T and depth L. By the uniformity of \mathcal{P} , it can be generated by a $\mathsf{poly}(T)$ -time uniform Turing machine with advice (pk, x) .

2. Let $\mathsf{ECC}(D_{x,\mathsf{vk}}^{\mathcal{V}})$ be the circuit obtained by applying ECC (from Lemma 3.19) to the circuit $D_{x,\mathsf{vk}}^{\mathcal{V}}$, where $D_{x,\mathsf{vk}}^{\mathcal{V}}$ is the circuit that takes as input π and computes $\mathcal{V}_2(\mathsf{vk}, x, \pi)$. Denote the output of $\mathsf{ECC}(D_{x,\mathsf{vk}}^{\mathcal{V}})$ by b.

 $\mathsf{ECC}(D_{x,\mathsf{vk}}^{\mathcal{V}})$ is a circuit of size $T \cdot \mathrm{polylog}(T)$ and depth $L \cdot \mathrm{polylog}(T)$. By the uniformity of \mathcal{V} and ECC it can be generated by a $\mathsf{poly}(T)$ -time uniform Turing machine with advice (vk, x) .

3. $E_{\mathsf{pk},\mathsf{vk},x}$ is the circuit obtained by applying Lemma 4.3 to $\{C_x\}$ and $\{\mathsf{ECC}(D_{x,\mathsf{vk}}^{\mathcal{V}})(\mathsf{vk}, D_{x,\mathsf{pk}}^{\mathcal{P}}(\mathsf{pk}, \cdot))\}$, where $D_{x,\mathsf{vk}}^{\mathcal{V}}(\mathsf{vk}, D_{x,\mathsf{pk}}^{\mathcal{P}}(\mathsf{pk}, \cdot))$ is the composition of $D_{x,\mathsf{vk}}^{\mathcal{V}}, D_{x,\mathsf{pk}}^{\mathcal{P}}$.

The reason we can apply Lemma 4.3 is that we have an \mathcal{EF} proof of length L for the statement

$$C_x(\cdot) \to D_{x,\mathsf{vk}}^{\mathcal{V}}(\mathsf{vk}, D_{x,\mathsf{pk}}^{\mathcal{P}}(\mathsf{pk}, \cdot)),$$

which is also an \mathcal{EF} proof for the statement

$$C_x(\cdot) \to \mathsf{ECC}(D_{x,\mathsf{vk}}^{\mathcal{V}})(\mathsf{vk}, D_{x,\mathsf{pk}}^{\mathcal{P}}(\mathsf{pk}, \cdot)),$$

since ECC merely adds some gates to the circuit.

Note that $\mathsf{ECC}(D_{x,\mathsf{vk}}^{\mathcal{V}})(\mathsf{vk}, D_{x,\mathsf{pk}}^{\mathcal{P}}(\mathsf{pk}, \cdot))$ is a circuit of size $T \cdot \mathrm{polylog}(T)$ and depth $L \cdot \mathrm{polylog}(T)$ and C has size at most T and depth at most L (by assumption). Moreover, both C_x and $\mathsf{ECC}(D_{x,\mathsf{vk}}^{\mathcal{V}})(\mathsf{vk}, D_{x,\mathsf{pk}}^{\mathcal{P}}(\mathsf{pk}, \cdot))$ can be generated by a uniform $T \cdot \mathrm{polylog}(T)$ -time Turing machine with advice $(x, \mathsf{pk}, \mathsf{vk})$.

Therefore, by Lemma 4.3, $E_{\mathsf{pk},\mathsf{vk},x}$ is a circuit of size $p(T+L) \cdot \mathsf{poly}(\lambda)$, depth $L \cdot \mathsf{poly}(\lambda)$, and can be generated by a uniform $p(T+L) \cdot \mathsf{poly}(\lambda)$ -time Turing machine with advice $(\mathsf{aux}_{\mathcal{EF}},\mathsf{pk},\mathsf{vk},x)$.

As mentioned above, we set the parameters T^* , d^* and D^* so that this extended circuit $E_{\mathsf{pk},\mathsf{vk},x}$ can be computed by size T^* and depth d^* circuit that can be generated by a T^* -time uniform Turing machine with advice (aux, x) where $|\mathsf{aux}| = D^*$. Let $\ell^* = \ell + (\log T^*)^A$ where A is the constant from Lemma 4.3. Note that these parameters satisfy Equation (5).

To prove the soundness, we will proceed in proving three claims, which together constitute a proof of soundness:

- 1. If we extract π from LocalGen, then the probability that π is accepted is negligible.
- 2. If we extract π and b from LocalGen, then the probability that π is rejected and b = 1 with is negligible.

Note that Items 1 and 2 imply that if we extract b from LocalGen, then with overwhelming probability b = 0.

3. If we extract b and the output bit of C_x , denoted by out, from LocalGen then the probability that b = 0 and out = 1 is negligible.

Note that this, together with Items 1 and 2, implies that If we extract out from LocalGen, then out = 1 with negligible probability, thus proving soundness.

Proof π is **Rejecting.** Denote the wires in $E_{\mathsf{pk},\mathsf{vk},x}$ corresponding to the ℓ output wires of $D_x^{\mathcal{P}}(\mathsf{pk},\cdot)$ by Π . Then by the soundness of the underlying argument system $(\mathcal{P},\mathcal{V})$ it holds that

$$\Pr[\mathcal{V}_2(\mathsf{vk}, x, \pi^*) = 1] = \mathsf{negl}(\lambda) \tag{6}$$

where the probability is over $r \leftarrow \{0,1\}^{\lambda}$ and over

$$\pi^* \leftarrow \mathsf{LocalGen}(D_x^{\mathcal{P}}(\mathsf{pk}, \cdot), \Pi)$$

where $(\mathsf{pk}, \mathsf{vk}) = \mathsf{Gen}_{\mathsf{aux}}(1^{\lambda}; r).$

By Theorem 3.20 (see Equation (3)), Equation (6) also holds when

$$\pi^* \leftarrow \mathsf{LocalGen}(E_{x,\mathsf{pk},\mathsf{vk}}(\cdot),\Pi).$$

and when

$$\pi^* \leftarrow \mathsf{LocalGen}(D_x^{\mathcal{V}}(\mathsf{vk},\cdot),\Pi)$$

Verdict Bit b is Zero. Next, let w_{verdict} denote the wire corresponding to the output $\text{ECC}(D_{\text{vk},x}^{\mathcal{V}})$ on input π^* , and let

$$(\pi^*, b^*) \leftarrow \mathsf{LocalGen}\left(\mathsf{ECC}(D^{\mathcal{V}}_{\mathsf{vk}, x})), \Pi \cup \{w_{\mathsf{verdict}}\}\right)$$

By Lemma 3.19, together with the fact that π^* is rejected with overwhelming probability, implies that

$$\Pr[b^* = 1] = \operatorname{\mathsf{negl}}(\lambda). \tag{7}$$

By Theorem 3.20 (see Equation (3)), Equation (7) also holds when

 $b^* \leftarrow \mathsf{LocalGen}\left(E_{x,\mathsf{pk},\mathsf{vk}}(\cdot), \{w_{\mathsf{verdict}}\}\right).$

Output of C_x is Zero. We next argue that this implies that

$$\Pr[\mathsf{out}^* = 1] = \mathsf{negl}(\lambda)$$

as desired, where

$$\mathsf{out}^* \leftarrow \mathsf{LocalGen}\left(E_{x,\mathsf{pk},\mathsf{vk}}(\cdot),\cdot\right), \{w_{\mathsf{out}}\}\right)$$

and where w_{out} is the wire corresponding to the output wire of C_x . To this end we use $D_x^{\mathcal{EF}}$. From Lemma 4.3, we have

$$\Pr[\mathsf{out}^* = 1 \land b^* = 0] = \mathsf{negl}(\lambda),\tag{8}$$

where

$$(\mathsf{out}^*, b^*) \leftarrow \mathsf{LocalGen}(E_{x,\mathsf{pk},\mathsf{vk}}(\cdot), \{w_{\mathsf{out}}, w_{\mathsf{verdict}}\}).$$

By Equation (7) (and the two lines following this equation) and by the non-signaling property, $\Pr[b^* = 1] = \operatorname{negl}(\lambda)$, which together with Equation (8) implies that indeed $\Pr[\operatorname{out}^* = 1] = \operatorname{negl}(\lambda)$, as desired.

Proof of Lemma 4.3. We construct the extension circuit $C'_{x,\lambda}$ and then invoke Theorem 3.22 from [JKLV24] to prove the lemma. To this end, denote the output wire of $D_{x,\lambda}$ by out_d , and the output wire of $C_{x,\lambda}$ by out_c . Let $C'_{x,\lambda}$ be the circuit that computes $C_{x,\lambda}$ and $D_{x,\lambda}$ and appends to the output wires out_c and out_d the sub-circuit $\neg\mathsf{out}_d \land \mathsf{out}_c$ (which consists of only two gates). Set the output wire of $C'_{x,\lambda}$ to be the wire computing $\neg\mathsf{out}_d \land \mathsf{out}_c$.

We use the \mathcal{EF} proof of $C_{x,\lambda} \to D_{x,\lambda}$ to construct an \mathcal{EF} proof for the fact that $C'_{x,\lambda}$ is unsatisfiable, as follows: The first L lines is an \mathcal{EF} proof for $\mathsf{out}_c \to \mathsf{out}_d$, which is identical to the \mathcal{EF} proof for $C_{x,\lambda} \to D_{x,\lambda}$. The rest of the proof consists of constant number of lines that derive that $\neg \mathsf{out}_d \land \mathsf{out}_c = \mathsf{F}$ from $\mathsf{out}_c \to \mathsf{out}_d$. Note that this \mathcal{EF} proof can be generated by a $(T + L + \lambda)$ -time uniform Turing machine with advice $\mathsf{aux}_{\mathcal{EF}} = \mathsf{aux}_{\mathcal{EF},x,\lambda}$.

Next we apply Theorem 3.22 to conclude that there exists a fixed polynomial poly, independent of $C_{x,\lambda}$ and $D_{x,\lambda}$, such that $\{C'_{x,\lambda}\}_{x\in\{0,1\}^*,\lambda\in\mathbb{N}}$ has an extension $\{E_{x,\lambda}\}_{x\in\{0,1\}^*,\lambda\in\mathbb{N}}$, such that $E_{x,\lambda}$ has size $\leq \operatorname{poly}(T+L)$ for a fixed poly, depth $\leq (d+L) \cdot \operatorname{poly}(\lambda)$, can be generated by a $\operatorname{poly}(T+L)$ time uniform Turing machine with advice $(\operatorname{aux}_x, \operatorname{aux}_{x,\mathcal{EF}})$, and for every ℓ -local assignment generator LocalGen (where $\ell = \operatorname{polylog}(T+L)$).

$$\Pr[\mathsf{LocalGen}(E'_x, \mathsf{out}) = 1] = \mathsf{negl}(\lambda).$$

We prove that $\{E_{x,\lambda}\}_{x\in\{0,1\}^*,\lambda\in\mathbb{N}}$ achieves the required property in the claim. Denote **out'** as the output of $C'_{x,\lambda}$, and denote $w_{\mathsf{out'}}$ as the wire representing **out'**. Since we set the locality of LocalGen to be $(\log(T+L))^A$ with a large enough constant A, we can set it to be larger than ℓ and thus

$$\Pr[\mathsf{out}' = 1] = \mathsf{negl}(\lambda),$$

where

$$\mathsf{out}' \leftarrow \mathsf{LocalGen}(\{w_{\mathsf{out}'}\})$$

Applying the non-signaling property of LocalGen, we have that out' = 0 holds with overwhelming probability, if we further extract out_c , out_d , out'. Moreover, the local satisfiability guarantees that $out' = \neg out_d \land out_c$, except with negligible probability. Hence, we have

$$\Pr[\mathsf{out}_d = 0 \land \mathsf{out}_c = 1] = \mathsf{negl}(\lambda),$$

where $(\mathsf{out}_d, \mathsf{out}_c) \leftarrow \mathsf{LocalGen}(\{w_d, w_{\mathsf{out}}\}).$

This finishes the proof of the lemma.

5 Constructions of Adaptively Sound SNARGs

5.1 Adaptively Sound SNARGs from Designated-Verifier SNARGs

We show how to convert any adaptively sound designated-verifier SNARG (dv-SNARG) into an adaptively sound SNARG (which is publicly verifiable), as long as the dv-SNARG has a polynomial-size Extended Frege (\mathcal{EF}) proof of completeness. We emphasize that we do *not* require the reference string of the underlying dv-SNARG to be reusable for multiple executions.

Theorem 5.1. Let \mathcal{L} be an NP language with a corresponding NP relation $\mathcal{R}_{\mathcal{L}}$. Assume the existence of a leveled, gate-by-gate FHE scheme (Definition 3.4) and a BARG scheme (Definition 3.10).

Assume that for any parameters $T = T(\lambda, n), L = L(\lambda, n)$, and $\ell = \ell(\lambda, n)$, such that $T, L, \ell \leq 2^{\lambda}$, T is larger than the runtime of the Turing machine computing $\mathcal{R}_{\mathcal{L}}$, and L larger than the depth of the circuit computing $\mathcal{R}_{\mathcal{L}}$, there exists an adaptively sound dv-SNARG for \mathcal{L} , denoted by $(\mathcal{P}, \mathcal{V})$, with the following syntax and parameters:

- \mathcal{V} consists of two algorithms $\mathcal{V}_1, \mathcal{V}_2$, where \mathcal{V}_1 takes as input the security parameter, and generates a prover key and a verification key $(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathcal{V}_1(1^{\lambda})$ of length L.
- \mathcal{P} is a time T algorithm that can be implemented by a uniform circuit of size T and depth L. It takes the input (pk, x, w) where $(x, w) \in \mathcal{R}_{\mathcal{L}}$, and output a proof $\pi = \mathcal{P}(\mathsf{pk}, x, w)$ of length ℓ .
- \mathcal{V}_2 is a time T algorithm that can be implemented by a uniform circuit of size T and depth L. It takes the input $(\forall k, x, \pi)$ and outputs a verdict bit b.
- For every $n, \lambda \in \mathbb{N}$ such that $n \leq 2^{\lambda}$, there exists an \mathcal{EF} proof of length L for the completeness of $(\mathcal{P}, \mathcal{V})$, which is following statement: "For every $r \in \{0, 1\}^{\lambda}$ every $x \in \{0, 1\}^{n}$ and every w, and for $(\mathsf{pk}, \mathsf{vk}) = \mathcal{V}_1(1^{\lambda}; r)$, if $\mathcal{R}_{\mathcal{L}}(x, w) = 1$ then $\mathcal{V}_2(\mathsf{vk}, x, \mathcal{P}(\mathsf{pk}, x, w)) = 1$."

Then there exists an adaptively sound SNARG for \mathcal{L} with the following properties.

- The crs is of length $(L + \ell) \cdot \operatorname{poly}(\log T, \lambda)$.
- The proof is of length $\ell \cdot \operatorname{poly}(\log(T+L), \lambda)$.
- The prover runtime is $poly(T + L, \lambda)$.
- The verifier runtime is $\tilde{O}(n) + \text{poly}(L + \ell, \log T, \lambda)$.
- The crs is transparent, if pk is pseudorandom against distinguishers of size $T' \leq 2^{|x|}$, where T' is the minimum circuit size needed to decide the language \mathcal{L} .

Moreover, if we assume an unleveled gate-by-gate FHE scheme then we can avoid the restriction on the depth of the circuits \mathcal{P} and \mathcal{V}_2 and replace the restriction that the \mathcal{EF} proof must be of length L with the restriction that it can be generated by a T-time uniform Turing machine with an advice string of size L.

Before constructing the SNARG in Theorem 5.1, similar to Lemma 4.3, we need the following theorem for adaptive soundness. We defer the proof of this lemma to the end of this section.

Lemma 5.2. There exists a constant $A \in \mathbb{N}$ and a polynomial p such that the following holds. Let $\{C_{n,\lambda}\}_{n,\lambda\in\mathbb{N}}$ and $\{D_{n,\lambda}\}_{n,\lambda\in\mathbb{N}}$ be circuit families such that each $D_{n,\lambda}$ takes the wires of $C_{n,\lambda}$ as input, and let $T(n,\lambda)$, $d = d(n,\lambda)$ and $L = L(n,\lambda)$ be polynomials that satisfy the following two conditions:

- 1. The circuit that concatenates $C_{n,\lambda}$ and $D_{n,\lambda}$ is of size $T = T(n,\lambda)$ and depth $d = d(n,\lambda)$ and can be generated by a (T + L)-time uniform Turing machine with advice $aux = aux_{n,\lambda}$.
- 2. There exists an \mathcal{EF} proof of $C_{n,\lambda} \to D_{n,\lambda}$ (as defined in Definition 3.14) of length $L = L(n,\lambda)$ that can be generated by a (T+L)-time uniform Turing machine with advice $\mathsf{aux}_{\mathcal{EF}} = \mathsf{aux}_{\mathcal{EF},n,\lambda}$.

Then there exists a circuit family $\{E_{n,\lambda}\}_{n,\lambda\in\mathbb{N}}$ such that the following holds.

- Each $E_{n,\lambda}$ is of size p(T+L) and depth $(d+L) \cdot p(\lambda)$ and can be generated by a p(T+L)-time uniform Turing machine with advice (aux, aux_{EF}),
- $E_{n,\lambda}$ extends the circuit that concatenates $C_{n,\lambda}$ and $D_{n,\lambda}$,
- For every adaptive $(\log(T+L))^A$ -local assignment generator LocalGen for $E_{n,\lambda}$, we have

$$\Pr[c^* = 1 \land d^* = 0] = \mathsf{negl}(\lambda),$$

where the randomness is over $(x, (c^*, d^*)) \leftarrow \mathsf{LocalGen}(\{w_c, w_d\})$, and w_c is the output wire of $C_{n,\lambda}$, w_d is the output wire of $D_{n,\lambda}$.

Construction. We construct the (publicly-verifiable) SNARG scheme (Gen_{SNARG}, \mathcal{P}_{SNARG} , \mathcal{V}_{SNARG}) from the following ingredients.

- An adaptively-sound dv-SNARG scheme $(\mathcal{P}, \mathcal{V})$ for \mathcal{L} .
- An Encrypt-Hash-and-BARG scheme $\mathsf{EHB} = (\mathsf{Gen}_{\mathsf{EHB}}, \mathcal{P}_{\mathsf{EHB}}, \mathcal{V}_{\mathsf{EHB}})$ from Theorem 3.20.

Our construction of the (publicly-verifiable) SNARG for \mathcal{L} will make use of the following circuits.

- C_{pk} : Takes as input $(x, w, \pi_{\mathcal{P}})$ and computes $C_x(w) \land \pi_{\mathcal{P}} = \mathcal{P}(\mathsf{pk}, x, w)$. Suppose C_{pk} is a circuit of size $\leq T$ and depth $\leq L$. By the uniformity of \mathcal{P} and $\mathcal{R}_{\mathcal{L}}$, it can be generated by a $\mathsf{poly}(T)$ -time uniform Turing machine with advice pk .
- D_{vk} : Takes as input $(x, \pi_{\mathcal{P}})$ and outputs a verdict bit $b = \mathsf{ECC}(\mathcal{V}_{vk})(x, \pi_{\mathcal{P}})$, where $\mathsf{ECC}(\mathcal{V}_{vk})$ is the encoding of the circuit \mathcal{V}_{vk} from Lemma 3.19 and \mathcal{V}_{vk} is the verification circuit of $(\mathcal{P}, \mathcal{V})$ which on input $(x, \pi_{\mathcal{P}})$ outputs $\mathcal{V}_2(vk, x, \pi_{\mathcal{P}})$.

 D_{vk} is a circuit of size $T \cdot \operatorname{polylog}(T)$ and depth $L \cdot \operatorname{polylog}(T)$. By the uniformity of \mathcal{V} and ECC it can be generated by a $\operatorname{poly}(T)$ -time uniform Turing machine with advice vk .

• $E = E_{\mathsf{pk},\mathsf{vk}}$: This circuit is obtained by applying Lemma 5.2 to C_{pk} and D_{vk} . We will show that the \mathcal{EF} proof of completeness implies an \mathcal{EF} proof for $C_{\mathsf{pk}} \to D_{\mathsf{vk}}$. Hence, the premise of Lemma 5.2 is satisfied.

We next formally define our (publicly-verifiable) SNARG construction for \mathcal{L} . Our construction only uses the circuit C_{pk} and the *size* of E. The content of E is only used in the proof of adaptive soundness.

- Key Generation. The key generation algorithm $Gen_{SNARG}(1^{\lambda})$
 - Generate the prover key $(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathcal{V}_1(1^{\lambda})$ for the dv-SNARG.
 - Generate crs for the SNARG scheme from Theorem 3.20 with parameters (T^*, d^*, D^*, ℓ^*) , where

$$T^* = \operatorname{poly}(T), \quad d^* = L \cdot \operatorname{poly}(\lambda), \quad D^* = L \cdot \operatorname{poly}(\lambda) \text{ and } \ell^* = \operatorname{polylog}(T+L).$$
 (9)

These parameters are chosen so that the extended circuit E defined above is of depth d^* , size T^* , and can be generated by a T^* -time uniform Turing machine with advice of size D^* . We denote this SNARG scheme by EHB (for encrypt-hash-and-BARG), and denote its crs by crs_{EHB} \leftarrow Gen_{EHB} (1^{λ}) .

- Output $crs = (pk, crs_{EHB})$.
- **Prover** $\mathcal{P}_{SNARG}(crs, x, w)$. The prover algorithm does the following:
 - Compute a dv-SNARG proof $\pi_{\mathcal{P}} \leftarrow \mathcal{P}(\mathsf{pk}, x, w)$.
 - Let \mathcal{L}_{pk} be the following language.

 $\mathcal{L}_{\mathsf{pk}} = \{ (x, \pi_{\mathcal{P}}) \mid \text{there exists } w \text{ such that } C_{\mathsf{pk}}(x, w, \pi_{\mathcal{P}}) = 1 \}.$

Use the EHB SNARG to prove that $(x, \pi_{\mathcal{P}}) \in \mathcal{L}_{pk}$, using w as the witness. Namely, let

 $\pi_{\mathsf{EHB}} \leftarrow \mathcal{P}_{\mathsf{EHB}}(\mathsf{crs}_{\mathsf{EHB}}, (x, \pi_{\mathcal{P}}), w).$

- Output the proof $\pi = (\pi_{\mathcal{P}}, \pi_{\mathsf{EHB}}).$
- Verifier $\mathcal{V}(crs, x, \pi)$. The verifier does the following:
 - Parse $crs = (pk, crs_{EHB})$ and $\pi = (\pi_{\mathcal{P}}, \pi_{EHB})$.
 - Verify the EHB proof π_{EHB} using the instance $(x, \pi_{\mathcal{P}})$. If the EHB verification accepts, then accept (output 1). Otherwise, reject the proof (output 0). Namely, output

$$\mathcal{V}_{\mathsf{EHB}}(\mathsf{crs}_{\mathsf{EHB}}, (x, \pi_{\mathcal{P}}), \pi_{\mathsf{EHB}}).$$

The completeness of the above construction follows directly from the completeness of the dv-SNARG $(\mathcal{P}, \mathcal{V})$ and from the completeness of the EHB scheme.

5.2 **Proof of Adaptive Soundness**

We prove adaptive soundness by contradiction. For any cheating prover $\mathcal{P}^*_{\mathsf{SNARG}}$, suppose $\mathcal{P}^*_{\mathsf{SNARG}}$ can break the adaptive soundness with a non-negligible probability $\epsilon = \epsilon(\lambda)$. Before we present the formal proof, we first provide a sketch of our ideas to reach a contradiction.

1. Denote by $x^*, \pi_{\mathcal{P}}^*$ the instance and dv-SNARG chosen by $\mathcal{P}^*_{\mathsf{SNARG}}$ on input crs. By the soundness of the underlying dv-SNARG if $x^* \notin \mathcal{L}$ then $\mathcal{V}_2(\mathsf{vk}, x^*, \pi_{\mathcal{P}}^*) = 0$ with overwhelming probability. The fact that $\mathcal{P}^*_{\mathsf{SNARG}}$ breaks the adaptive soundness with non-negligible probability ϵ , implies that

 $\Pr[\mathcal{P}^*_{\mathsf{SNARG}} \text{ is accepted } \land \mathcal{V}_2(\mathsf{vk}, x^*, \pi^*_{\mathcal{P}}) = 0] \ge \epsilon(\lambda) - \mathsf{negl}(\lambda)$

- By Theorem 3.20, we can build an adaptive local assignment generator LocalGen from P^{*}_{SNARG}. Since LocalGen computationally preserves the distribution of the instance, this implies that Pr[V₂(vk, x^{*}, π^{*}_P) = 0] ≥ ε − negl for (x^{*}, π^{*}_P) which are outputted by LocalGen.
- 3. Use Lemma 3.19 to argue that if we run LocalGen on the extended circuit E, and extract the output wire value d^* of D_{vk} , then $d^* = 0$ with probability ϵnegl .
- 4. Invoke Lemma 5.2 to argue that if we extract the output wire of C_{pk} as out^* and the output wire of D_{vk} as d^* , then with overwhelming probability, $out^* \to d^*$ holds. The premise of Lemma 5.2 is satisfied because the polynomial-size \mathcal{EF} for the completeness of $(\mathcal{P}, \mathcal{V})$ is also a \mathcal{EF} proof for $C_{pk} \to D_{vk}$.

Thus we derive a contradiction by concluding that $\mathsf{out}^* = 0$ with probability at least $\epsilon - \mathsf{negl}$. This follows from the fact that since $d^* = 0$ with probability $\epsilon - \mathsf{negl}$, and $\mathsf{out}^* \to d^*$ with overwhelming probability.

We proceed with the formal proof. We first argue that

$$\Pr[x^* \notin \mathcal{L} \land \mathcal{V}_2(\mathsf{vk}, x^*, \pi_{\mathcal{P}}^*) = 1] = \mathsf{negl}(\lambda)$$
(10)

where the probability is over $\operatorname{crs} = (\operatorname{pk}, \operatorname{crs}_{\mathsf{EHB}}) \leftarrow \operatorname{Gen}_{\mathsf{SNARG}}(1^{\lambda})$ and over $(x^*, \pi^* = (\pi_{\mathcal{P}}^*, \pi_{\mathsf{EHB}}^*)) \leftarrow \mathcal{P}_{\mathsf{SNARG}}^*(\operatorname{crs})$. This follows from the adaptive soundness of dv-SNARG. Combining Equation (10) with the fact that the cheating prover succeeds in cheating with probability $\epsilon(\lambda)$, i.e.,

$$\Pr[x^* \notin \mathcal{L} \land \mathcal{V}_{\mathsf{SNARG}}(\mathsf{crs}, x^*, \pi^*) = 1] \ge \epsilon(\lambda),$$

we conclude that

$$\Pr[\mathcal{V}_{\mathsf{SNARG}}(\mathsf{crs}, x^*, \pi^*) = 1 \land \mathcal{V}_2(\mathsf{vk}, x^*, \pi^*_{\mathcal{P}}) = 0] \ge \epsilon(\lambda) - \mathsf{negl}(\lambda).$$
(11)

Next, we invoke Theorem 3.20, which states that there exists a universal adaptive local assignment generator LocalGen with the following property. The distribution of $(x^*, \pi_{\mathcal{P}}^*)$ extracted from LocalGen is computationally indistinguishable from $(x^*, \pi_{\mathcal{P}}^*)$ outputted by the adversary conditioned on the event that $\mathcal{V}_{\mathsf{EHB}}(\mathsf{crs}_{\mathsf{EHB}}, (x^*, \pi_{\mathcal{P}}^*), \pi_{\mathsf{EHB}}^*) = 1$. Hence, we have

$$\Pr[\mathcal{V}_2(\mathsf{vk}, x^*, \pi_{\mathcal{P}}^*) = 0] \ge \epsilon(\lambda) - \mathsf{negl}(\lambda),$$

where $(x^*, \pi_{\mathcal{P}}^*) \leftarrow \mathsf{LocalGen}(\emptyset)$, where \emptyset denotes the empty set.

Denote the output wire of D_{vk} by w_d , then we can argue that

$$\Pr[d^* = 0] \ge \epsilon(\lambda) - \operatorname{\mathsf{negl}}(\lambda), \tag{12}$$

where $((x^*, \pi_{\mathcal{P}}^*), d^*) \leftarrow \mathsf{LocalGen}(D_{\mathsf{vk}}, \{w_d\})$. This follows from Lemma 3.19 and the fact that $\mathcal{V}_2(\mathsf{vk}, x^*, \pi_{\mathcal{P}}^*) = 0$ with probability $\epsilon(\lambda) - \mathsf{negl}(\lambda)$. By the non-signaling condition of $\mathsf{LocalGen}$, Equation (12) also holds where $((x^*, \pi_{\mathcal{P}}^*), d^*) \leftarrow \mathsf{LocalGen}(E, \{w_d\})$.

Since we have a length $L \mathcal{EF}$ proof of the completeness of the dv-SNARG, which states that

$$\left((\mathsf{pk},\mathsf{vk}) = \mathcal{V}_1(1^{\lambda};r) \land C_x(w) = 1 \right) \to \mathcal{V}_2\left(\mathsf{vk},x,\mathcal{P}(\mathsf{pk},x,w)\right) = 1,$$

we also have an \mathcal{EF} proof for the following statement of length O(L) (which we assume is smaller than d^* and D^*): for any $(\mathsf{pk}, \mathsf{vk}) = \mathcal{V}_1(1^{\lambda}; r)$,

$$C_{\mathsf{pk}}(x, w, \pi_{\mathcal{P}}^*) \to D_{\mathsf{vk}}(x, \pi_{\mathcal{P}}^*).$$

Hence, the premise of Lemma 5.2 is satisfied. Therefore, denoting the output wires of C_{pk} , D_{vk} by w_{out} and w_d respectively,

$$\Pr[\mathsf{out}^* = 1 \land d^* = 0] = \mathsf{negl}(\lambda) \tag{13}$$

where $((x^*, \pi_{\mathcal{P}}^*), (\mathsf{out}^*, d^*)) \leftarrow \mathsf{LocalGen}(E, \{w_{\mathsf{out}}, w_d\})$. Combining Equations (12) and (13) we conclude that $\Pr[\mathsf{out}^* = 0] \geq \epsilon(\lambda) - \mathsf{negl}(\lambda)$, implying a contradiction to the fact that $\Pr[\mathsf{out}^* = 0] = \mathsf{negl}(\lambda)$.

Proof of Lemma 5.2. The proof of the lemma follows from the same idea as the proof of Lemma 4.3. Namely, we will construct a circuit $C'_{n,\lambda} = \neg D_{n,\lambda} \wedge C_{n,\lambda}$ for every n, λ , and then invoke Theorem 3.23 to prove the lemma.

Formally, $C'_{n,\lambda}$ is constructed by taking the output wire out_D of $D_{n,\lambda}$ and the output wire out_C of $C_{n,\lambda}$, and build 2 new gates to compute $\neg \operatorname{out}_D \land \operatorname{out}_C$. Then $C'_{n,\lambda}$ can be implemented by a uniform circuit of size T + 2 and depth d + 2 that can be generated by a $(T + \lambda)$ -time Turing machine with advise $\operatorname{aux}_{n,\lambda}$.

We prove that $\{C'_{n,\lambda}\}_{n,\lambda\in\mathbb{N}}$ has polynomial-size \mathcal{EF} proofs of unsatisfiability, by using the \mathcal{EF} proofs of $C_{n,\lambda} \to D_{n,\lambda}$. The proof of unsatisfiability consists of a length $L \mathcal{EF}$ proof of $\neg \mathsf{out}_D \to \mathsf{out}_C$ and an additional constant size \mathcal{EF} proof of $\neg \mathsf{out}_D \land \mathsf{out}_C = \mathsf{F}$ derived from $\neg \mathsf{out}_D \to \mathsf{out}_C$. The total length of the \mathcal{EF} proof of unsatisfiability is bounded by $L + \lambda$, and it can be generated by a $(T + L + \lambda)$ Turing machine with advise $(\mathsf{aux}, \mathsf{aux}_{\mathcal{EF}})$.

From Theorem 3.23, there exists exists an extension circuit $E_{n,\lambda}$ of $C_{n,\lambda}$ of size $T' \leq \operatorname{poly}(T,L)$, depth $D' \leq \operatorname{poly}(d,L)$ that can be generated by T'-time uniform Turing machine with advise (aux, aux_{\mathcal{EF}}), such that for any $\ell = \operatorname{polylog}(T',L')$, and for any (T',d',D')-universal adaptive local assignment generator LocalGen, we have that

$$\Pr[(x, \mathsf{out}_{C'}) \leftarrow \mathsf{LocalGen}(E_{n,\lambda}, \mathsf{out}_{C'}) : \mathsf{out}_{C'} = 1] = \mathsf{negl}(\lambda).$$
(14)

Now, if we further extract the wires out_C , out_D , and $out_{C'}$, then from the local consistency of LocalGen, we have

$$\Pr[\mathsf{out}_{C'} \neq \neg \mathsf{out}_D \land \mathsf{out}_C] = \mathsf{negl}(\lambda), \tag{15}$$

where $(x, \mathsf{out}_{C'}, \mathsf{out}_D, \mathsf{out}_C) \leftarrow \mathsf{LocalGen}(E_{n,\lambda}, \{\mathsf{out}_{C'}, \mathsf{out}_D, \mathsf{out}_C\}).$

Combining Equation (14) and Equation (15) and the non-signaling property of LocalGen, we derive that

$$\Pr[(x, \mathsf{out}_C, \mathsf{out}_D) \leftarrow \mathsf{LocalGen}(E_{n,\lambda}, \{\mathsf{out}_C, \mathsf{out}_D\}) : \mathsf{out}_C = 1 \land \neg \mathsf{out}_D = 0] = \mathsf{negl}(\lambda).$$

This finishes the proof.

Lemma 5.3 (Transparent CRS). If the pk in the underlying adaptively sound dv-SNARG for \mathcal{L} is pseudorandom even against distinguishers of size T', there is an adaptively sound SNARG for \mathcal{L} with an transparent common reference string.

Proof. Recall that in the construction of Theorem 5.1, the crs is sampled as follows:

- Generate a prover key $(pk, vk) \leftarrow \mathcal{V}_1(1^{\lambda})$ from the dv-SNARG. The verification key vk is discarded.
- The rest of the crs comprises crs_{EHB}, which can be sampled transparently.

Consider the **crs** that is instead sampled in the following way:

- Generate $\mathsf{pk} \leftarrow \{0,1\}^r$, where r is the length of the prover key output by $\mathcal{V}_1(1^{\lambda})$.
- The rest of the **crs** is sampled transparently as before.

The prover algorithm \mathcal{P} and verifier algorithm \mathcal{V} remain the same.

Completeness. The completeness follows directly from the completeness of the underlying EHB scheme.

Adaptive soundness. Suppose otherwise, and there exists a cheating prover $\mathcal{P}^*_{\mathsf{SNARG}}$:

$$\Pr\left[\begin{array}{cc} x^* \notin \mathcal{L} \land \mathcal{V}_{\mathsf{SNARG}}(\mathsf{crs}, x^*, \pi^*) = 1 & : & \frac{\mathsf{crs} \leftarrow \mathsf{Gen}_{\mathsf{SNARG}}(1^{\lambda}), \\ (x^*, \pi^*) \leftarrow \mathcal{P}^*_{\mathsf{SNARG}}(\mathsf{crs}) \end{array}\right] \ge \varepsilon(\lambda)$$

We use $\mathcal{P}^*_{\mathsf{SNARG}}$ to construct a distinguisher \mathcal{A} that distinguishes between pk sampled and $\mathsf{pk} \leftarrow \mathcal{V}_1(1^{\lambda})$ from $\mathsf{pk} \leftarrow \{0,1\}^r$ as follows:

- Obtain pk from the challenger (sampled either from $(\mathsf{pk},\mathsf{sk}) \leftarrow \mathcal{V}_1(1^{\lambda})$ or $\mathsf{pk} \leftarrow \{0,1\}^r$).
- Sample the rest of the common reference string crs_{EHB} independently, and set $crs = (pk, crs_{EHB})$.
- Send crs to $\mathcal{P}^*_{\mathsf{SNARG}}$ and obtain (x^*, π^*) .
- If $x^* \notin L$ (this can be checked by a circuit of size T') and $\mathcal{V}_{\mathsf{SNARG}}(\mathsf{crs}, x^*, \pi^*) = 1$, output 1. Else, output 0.

If pk were sampled from $(pk, vk) \leftarrow \mathcal{V}_1(1^{\lambda})$, then by the soundness of the scheme in Theorem 5.1, $\mathcal{P}^*_{\mathsf{SNARG}}$ produces a cheating proof (x^*, π^*) for $x^* \notin L$ with negligible probability.

$$\Pr_{(\mathsf{pk},\mathsf{sk})\leftarrow\mathcal{V}_1(1^\lambda)}[\mathcal{A}(\mathsf{pk})=1]=\mathsf{negl}(\lambda).$$

Otherwise, if $\mathsf{pk} \leftarrow \{0,1\}^r$, then $\mathcal{P}^*_{\mathsf{SNARG}}$ produces proofs with probability $\varepsilon(\lambda)$. Hence,

$$\Pr_{\mathsf{pk} \leftarrow \{0,1\}^r}[\mathcal{A}(\mathsf{pk}) = 1] = \varepsilon(\lambda).$$

Therefore, \mathcal{A} distinguishes the two distributions with probability $\varepsilon(\lambda) - \mathsf{negl}(\lambda)$, which is in non-negligible if $\varepsilon(\cdot)$ is non-negligible, contradicting Theorem 5.1. Therefore, we have $\epsilon(\lambda)$ is negligible and the modified scheme is sound.

6 Application I: Non-Adaptive SNARGs from Witness Encryption

In this section, we first state our main result in Theorem 6.4 that one can construct a non-adaptive SNARG for a language L given a witness encryption scheme for L with a \mathcal{EF} proof of completeness. We then show the following two implications:

Theorem 6.1. Assuming subexponential LWE and evasive LWE, there exists a non-adaptive SNARG for all $\mathcal{L} \in \mathsf{NP}$ with relation circuit of size T with the following parameters:

- The crs is of length $poly(n, T, \lambda)$ with transparent set-up.
- The proof is of size $poly(\lambda, \log T)$.
- The prover runtime is $poly(n, T, \lambda)$.
- The verifier runtime is $poly(n, \lambda, \log T)$.

We show this theorem in Section 6.3.

Theorem 6.2. Assuming LWE, there exists non-adaptive SNARGs for the QR language, the $\overline{\text{QR}}$ language, and the $\overline{\text{DCR}}$ language (see Definitions 6.19 and 6.20).

We show this theorem in Section 6.4.

6.1 Witness Encryption

In this section, we recall the definition of witness encryption, introduced by [GGSW13].

Syntax. A witness encryption scheme for an NP language \mathcal{L} is a tuple of algorithms (WE.Enc, WE.Dec) satisfying

- WE.Enc $(1^{\lambda}, x, b) \to c$. This is a probabilistic algorithm that takes as input a security parameter λ , and instance x and a bit b. It outputs a ciphertext $c \in \{0, 1\}^{\ell}$.
- WE.Dec(c, w). This is a deterministic algorithm that takes as input a ciphertext c and a witness w, and outputs a bit m or \perp .

Definition 6.3. A tuple of algorithms (WE.Enc, WE.Dec) is a witness encryption scheme for an NP language L if the following hold:

Completeness. For any w such that $\mathcal{R}_{\mathcal{L}}(x, w) = 1$, for $b \in \{0, 1\}$, we have that

 $\Pr[\mathsf{WE}.\mathsf{Dec}(\mathsf{WE}.\mathsf{Enc}(1^{\lambda}, x, b), w)) = b] = 1.$

Semantic security. For every $x \notin L$ and every poly-size adversary \mathcal{A} , there exists a negligible function μ such that

$$\left|\Pr[\mathcal{A}(\mathsf{WE}.\mathsf{Enc}(1^{\lambda}, x, 0)) = 1] - \Pr[\mathcal{A}(\mathsf{WE}.\mathsf{Enc}(1^{\lambda}, x, 1)) = 1]\right| \le \mu(\lambda).$$

6.2 Main theorem statement (WE)

In this section, we state a theorem that immediately follows from Theorem 4.1 about constructing SNARGs from *witness encryption* schemes.

Theorem 6.4. Assume the existence of a leveled, gate-by-gate FHE scheme (Definition 3.4) and a BARG scheme (Definition 3.10). Fix any NP language $\mathcal{L} \subseteq \{0,1\}^*$ with a corresponding NP relation $\mathcal{R}_{\mathcal{L}}$. For any parameters $T = T(n, \lambda)$, $L = L(n, \lambda)$, and $\ell = \ell(n, \lambda)$, such that $T, L, \ell \leq 2^{\lambda}, T \geq \ell$ and T is larger than the size of NP verification circuit, and L is larger than the depth of the NP verification circuit. We construct a SNARG for \mathcal{L} with the following properties:

- The crs is of length $(L + \ell) \cdot \operatorname{poly}(\lambda)$.
- The proof length is $\ell \cdot \operatorname{poly}(\lambda)$.
- The prover runtime is $poly(T + L, \lambda)$

- The verifier runtime is $\tilde{O}(n) + \text{poly}(L + \ell, \lambda)^{22}$.
- Non-adaptive soundness assuming there exists a witness encryption scheme (Enc, Dec) with messages of length λ , ciphertexts of length at most L, encryption and decryption algorithms of depth at most L, and \mathcal{EF} proofs of decryption correctness:
 - For every $n, \lambda \in \mathbb{N}$ such that $n \leq 2^{\lambda}$, every $x \in \{0,1\}^n$, every encryption randomness r, and every message m, there is an \mathcal{EF} proof of length L for the statement: "For every w, if $C_x(w) = 1$ then $\mathsf{Dec}(w, \mathsf{Enc}(x, m; r)) = m$."

Moreover, if we assume an unleveled gate-by-gate FHE scheme then we can remove the depth bounds on encryption and decryption, and replace the restriction that the \mathcal{EF} proof must be of length L with the restriction that it can be generated by a T-time uniform Turing machine with an advice string of size L.

Theorem 6.4 holds because given such a witness encryption scheme for a language \mathcal{L} , one can construct a 2-message laconic argument system for \mathcal{L} satisfying the hypotheses of Theorem 4.1 in the usual way:

- Verifier message: sample a uniformly random message $m \leftarrow \{0,1\}^{\lambda}$ and send $\mathsf{Enc}(x,m)$.
- Prover message: given ct, compute $m' = \mathsf{Dec}(w, \mathsf{ct})$ and return m'.

Crucially, $(\mathcal{EF}$ -)correctness of the witness encryption scheme corresponds exactly to $(\mathcal{EF}$ -)completeness of the laconic argument system.

Given Theorem 6.4, we next construct two publicly verifiable SNARG schemes via explicit witness encryption schemes.

6.3 SNARG for NP from Evasive LWE

In this section, we build a witness encryption scheme from evasive LWE by expanding on the techniques from [GGH15, CVW18, VWW22, MPV24].

6.3.1 PV Proofs for Properties of Linear Algebra

In showing the completeness of our construction, we rely on the fact that many basic properties of linear algebra have PV proofs. To see this, our starting point is the work of [SC04] defines theory LA (linear algebra), and shows that the theorems that can be proven in LA can be translated into PV proofs. The theory comprises three types of objects: indices (natural numbers), field elements for a fixed field, and matrices with entries from that field. Then, they formalize standard field and matrix axioms for their theory (see their work for a formal treatment, and we give an informal overview for readability). They then show the following basic properties about matrices (if the matrix dimensions match up) over fields in LA, including:

- $\mathbf{0} \cdot \mathbf{A} = \mathbf{0}$ and $\mathbf{A} \cdot \mathbf{0} = \mathbf{0}$.
- $\mathbf{A} + \mathbf{B} = \mathbf{B} + \mathbf{A}$

²²As before, we note that the verifier runtime can be generically improved to $poly(n+\lambda+\ell)$ using a RAM delegation scheme.

- $\mathbf{A} + (\mathbf{B} + \mathbf{C}) = (\mathbf{A} + \mathbf{B}) + \mathbf{C}$
- $\mathbf{AI} = \mathbf{A}$ and $\mathbf{IA} = \mathbf{A}$
- A(BC) = (AB)C
- A(B+C) = AB + AC and (B+C)A = BA + CA.

Along with usual rules of PV, a proof in LA can also use the following two additional rules which informally are as follows:

- Matrix equality: If matrices A and B have the same dimensions, and A[v, w] = B[v, w] for all v, w, then A = B.
- Induction rule: For any formula α , if $\alpha(i) \rightarrow \alpha(i+1)$, then $\alpha(0) \rightarrow \alpha(n)$.²³

For the purpose of this section, we extend theory LA to also include the following function symbols:

• Kronecker product \otimes : Binary function symbol which takes two matrices **A** and **B** of size $m \times n$ and $p \times q$ respectively, the output is a $mp \times nq$ matrix as follows:

$$(\mathbf{A} \otimes \mathbf{B})[p(r-1) + v, q(s-1) + w] = \mathbf{A}[r, s] \cdot \mathbf{B}[v, w].$$

• Diagonal concatenation diag: Binary function symbol which takes two matrices A and B of size $m \times n$ and $p \times q$ respectively, the output is a

$$\operatorname{diag}(\mathbf{A}, \mathbf{B}) = \begin{pmatrix} \mathbf{A} & \mathbf{0}_{m \times q} \\ \mathbf{0}_{p \times n} & \mathbf{B} \end{pmatrix}$$

Then, it is easy to derive the following properties of the Kronecker product and diag in LA since they only require basic associative and distributive properties of matrices.

- $\mathbf{A} \otimes (\mathbf{B} + \mathbf{C}) = \mathbf{A} \otimes \mathbf{B} + \mathbf{A} \otimes \mathbf{C}$ (see below for a proof of this property)
- $(\mathbf{B} + \mathbf{C}) \otimes \mathbf{A} = \mathbf{B} \otimes \mathbf{A} + \mathbf{C} \otimes \mathbf{A}$
- $(\mathbf{A} \otimes \mathbf{B}) \otimes \mathbf{C} = \mathbf{A} \otimes (\mathbf{B} \otimes \mathbf{C})$
- $\mathbf{A} \otimes \mathbf{0} = \mathbf{0}$ and $\mathbf{0} \otimes \mathbf{A} = \mathbf{0}$
- $(\mathbf{A} \otimes \mathbf{B}) \cdot (\mathbf{C} \otimes \mathbf{D}) = (\mathbf{A}\mathbf{C}) \otimes (\mathbf{B}\mathbf{D})$
- $\operatorname{diag}(\mathbf{A}, \mathbf{B}) \cdot \operatorname{diag}(\mathbf{C}, \mathbf{D}) = \operatorname{diag}(\mathbf{A}\mathbf{C}, \mathbf{B}\mathbf{D}).$

We provide a proof of the first property as an explicit example.

 $^{^{23}}$ As long as *n* is polynomially bounded, the proof is polynomially bounded.

Proof. Note that

$$\mathbf{A} \otimes (\mathbf{B} + \mathbf{C})[p(r-1) + v, q(s-1) + w] = \mathbf{A}[r, s] \cdot (\mathbf{B} + \mathbf{C})[v, w]$$

= $\mathbf{A}[r, s] \cdot \mathbf{B}[v, w] + \mathbf{A}[r, s] \cdot \mathbf{C}[v, w]$
= $(\mathbf{A} \otimes \mathbf{B})[p(r-1) + v, q(s-1) + w]$
+ $(\mathbf{A} \otimes \mathbf{C})[p(r-1) + v, q(s-1) + w]$

where we used the distributive property of matrix multiplication. We can then conclude by matrix equality that $\mathbf{A} \otimes (\mathbf{B} + \mathbf{C}) = \mathbf{A} \otimes \mathbf{B} + \mathbf{A} \otimes \mathbf{C}$.

In particular, since all of the above properties can be proven in LA, all of the above properties can be proven using PV proofs. We use these properties extensively in the rest of this section, sometimes coupled with induction, e.g. to prove that $\prod_{i=1}^{n} \operatorname{diag}(\mathbf{A}_i, \mathbf{B}_i) = \operatorname{diag}(\prod_i \mathbf{A}_i, \prod_i \mathbf{B}_i)$.

6.3.2 Evasive LWE

In this section, we recall the evasive LWE assumption introduced in [Wee22], following the formalization by [VWW22]. Then, we recall some related tools from the works of [GGH15, CVW18, VWW22, MPV24].

Evasive LWE. Let $\sigma, \sigma' \in \mathbb{R}_{>0}$, and let Samp be a PPT algorithm that on input 1^{λ} outputs

$$\mathbf{S} \in \mathbb{Z}_q^{n' imes n}, \mathbf{P} \in \mathbb{Z}_q^{n imes t},$$
 aux $\in \{0,1\}^*$.

We define the following advantage functions:

$$\begin{aligned} \mathsf{Adv}_{\mathcal{A}_0}^{\mathrm{pre}}(\lambda) &:= \Pr[\mathcal{A}_0(\mathbf{SB} + \mathbf{E}, \mathbf{SP} + \mathbf{E}', \mathsf{aux}) = 1] - \Pr[\mathcal{A}_0(\mathbf{C}, \mathbf{C}', \mathsf{aux}) = 1], \\ \mathsf{Adv}_{\mathcal{A}_1}^{\mathrm{post}}(\lambda) &:= \Pr[\mathcal{A}_1(\mathbf{SB} + \mathbf{E}, \mathbf{D}, \mathsf{aux}) = 1] - \Pr[\mathcal{A}_1(\mathbf{C}, \mathbf{D}, \mathsf{aux}) = 1] \end{aligned}$$

where

$$\begin{split} & (\mathbf{S}, \mathbf{P}, \mathsf{aux}) \leftarrow \mathsf{Samp}(1^{\lambda}) \\ & \mathbf{B} \leftarrow \mathbb{Z}_q^{n \times m}, \mathbf{E} \leftarrow \mathcal{D}_{\mathbb{Z}, \sigma}^{n' \times m}, \mathbf{E}' \leftarrow \mathcal{D}_{\mathbb{Z}, \sigma'}^{n' \times t}, \\ & \mathbf{C} \leftarrow \mathbb{Z}_q^{n' \times m}, \mathbf{C}' \leftarrow \mathbb{Z}_q^{n' \times t}, \\ & \mathbf{D} \leftarrow \mathbf{B}^{-1}(\mathbf{P}, \sigma) \;. \end{split}$$

We say that the *evasive LWE* assumption $\mathsf{eLWE}(\mathsf{Samp}, \sigma, \sigma')$ holds if there exists some polynomial $Q(\cdot)$ such that for every PPT \mathcal{A}_1 there exists another PPT \mathcal{A}_0 such that

$$\mathsf{Adv}_{\mathcal{A}_0}^{\mathrm{pre}}(\lambda) \geq \mathsf{Adv}_{\mathcal{A}_1}^{\mathrm{post}}(\lambda)/Q(\lambda) - \mathsf{negl}(\lambda)$$

and time $(\mathcal{A}_0) \leq \text{time}(\mathcal{A}_1) \cdot Q(\lambda)$. In this work, we will assume evasive LWE with $\sigma = \sigma'$.

Notation. For the rest of this section, we use n to denote the LWE parameter, and we instead use ℓ or h to denote the instance/witness length (this should be clear from context). Additionally, we will be assuming $2^{n^{\delta}}$ hardness of LWE for $\delta > 0$.

6.3.3 Matrix Branching Programs

We will work with matrix branching programs (MBPs) that compute functions $f_k : \{0,1\}^{\ell} \to \mathbb{Z}_q$ for some prime q. In this paper we only consider **read-once MBPs** specified by a collection of matrices $(\mathbf{M}_{i,b} : i \in [\ell], b \in \{0,1\})$ and two vectors \mathbf{u}, \mathbf{v} (all over some ring \mathcal{R} , which, for us, will always be \mathbb{Z}_q for a prime q) such that for all $\mathbf{x} \in \{0,1\}^{\ell}$,

$$f(\mathbf{x}) = \mathbf{u}^T \left(\prod_{i=1}^{\ell} \mathbf{M}_{i,x_i}\right) \mathbf{v} ,$$

We use the short-hand $\mathbf{M}_{\mathbf{x}}$ to denote the subset product $\prod_{i=1}^{\ell} \mathbf{M}_{i,x_i}$.

6.3.4 Matrix Branching Program Encoding of CNF

Recall that a CNF formula Φ is a conjunction clauses, where each clause is a disjunction. Looking forward, our witness encryption construction will rely on the following branching program for CNFs (also used in [CVW18, VWW22]).

Construction 6.5 (Branching program for CNFs). Given a CNF formula $\Phi = \phi_1 \land \phi_2 \land \cdots \land \phi_c$ with c clauses and h variables x_1, \ldots, x_h , we define CNFEncode to do the following:

- Choose some $q \ge 2c$.
- Set $\mathbf{u} = (1 \ 1 \dots 1)^T \in \mathbb{Z}_q^{c \times 1}$.
- Initialize $\mathbf{M}_{i,b} = \mathbf{I}_c$. Iterate over ϕ_1, \ldots, ϕ_c and do the following:
 - For each x_i that appears in ϕ_j , set the *j*th entry of the diagonal of $\mathbf{M}_{i,1}$ to 0.
 - For each $\neg x_i$ that appears in ϕ_j , set the *j*th entry of the diagonal of $\mathbf{M}_{i,0}$ to 0.

CNFEncode outputs the program $\{\mathbf{u}, \{\mathbf{M}_{i,b}\}_{i \in [h], b \in \{0,1\}}\}$.

Lemma 6.6. Suppose Φ is a CNF formula on c clauses and h variables, and suppose If $\Phi(\mathbf{x}) = 1$, where \mathbf{x} is a bit vector corresponding to a variable assignment, then the program Γ_{ϕ} output by CNFEncode (in Construction 6.5) satisfies that

$$\Phi(\mathbf{x}) = 1 \iff \mathbf{u}^T \mathbf{M}_{\mathbf{x}} = \mathbf{0}$$

and $\|\mathbf{u}^T \mathbf{M}_{\mathbf{x}}\|_{\infty} \leq 1$. Moreover, there is a PV proof of the following claim:

For all CNF Formula
$$\Phi$$
, and variable assignment $\mathbf{x} \in \{0, 1\}^h$,
 $\{\mathbf{u}, \{\mathbf{M}_{i,b}\}_{i \in [h], b \in \{0,1\}}\} \leftarrow \mathsf{CNFEncode}(\Phi),$
 $if \Phi(\mathbf{x}) = 1 \ then \ \mathbf{u}^T \mathbf{M}_{\mathbf{x}} = 0.$

Proof (sketch). For a clause ϕ , let $\mathbb{1}_{\phi,\mathbf{x}}$ be the indicator denoting if variable assignment \mathbf{x} satisfies ϕ . Let $\Phi = \phi_1 \wedge \phi_2 \wedge \cdots \wedge \phi_c$. We claim that: $\mathbf{M}_{\mathbf{x}} = \operatorname{diag}(1 - \mathbb{1}_{\phi_1,\mathbf{x}}, \dots, 1 - \mathbb{1}_{\phi_c,\mathbf{x}})$.

To see this, note that since \mathbf{M}_{i,x_i} are all diagonal matrices, we can rewrite

$$\mathbf{M}_{\mathbf{x}}[i,i] = \prod_{j=1}^{n} \mathbf{M}_{j,x_j}[i,i]$$

Note that this property of diagonal matrices can be proven using basic properties of matrix multiplication and polynomial-time induction on the number of matrices, and hence has a PV proof.

Therefore, note $\mathbf{M}_{\mathbf{x}}[i, i]$ is equal to 1 unless $\mathbf{M}_{j, x_j}[i, i]$ for some j is 0. By construction, this must mean that if $x_j = 0$, then $\neg x_j \in \phi_i$ and hence \mathbf{x} satisfies ϕ_i . Otherwise, if $x_j = 1$, then $x_j \in \phi_i$ and \mathbf{x} once against satisfies ϕ_i . Therefore, $\mathbf{M}_{\mathbf{x}}[i, i] = 1 - \mathbb{1}_{\phi_i, \mathbf{x}}$ as desired.

Therefore, if $\Phi(\mathbf{x}) = 1$, then \mathbf{x} satisfies all ϕ_i and $\mathbf{M}_{\mathbf{x}} = \mathbf{0}^{c \times c}$, and hence $\mathbf{u}^T \mathbf{M}_{\mathbf{x}} = \mathbf{0}$. Otherwise, \mathbf{x} doesn't satisfy some ϕ_i , and hence $\mathbf{u}^T \mathbf{M}_{\mathbf{x}} \neq \mathbf{0}$. Moreover, it is clear that every entry of $\mathbf{u}^T \mathbf{M}_{\mathbf{x}}$ is either 0 or 1, and hence the norm bound follows. Since we used basic properties of matrix multiplication in the above proof, it indeed has a *PV* proof.

6.3.5 Trapdoor and Pre-image Sampling

Given a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ for $m \geq 2n \log q$, a vector $\mathbf{y} \in \mathbb{Z}_q^n$, and $\sigma > 0$, we use $\mathbf{A}^{-1}(\mathbf{y}, \sigma)$ to denote the distribution of a vector \mathbf{d} sampled from $\mathcal{D}_{\mathbb{Z},\sigma}^m$ conditioned on $\mathbf{A}\mathbf{d} = \mathbf{y} \pmod{q}$. (Vectors satisfying the condition exist except with probability $\operatorname{negl}(\mu)$.) We extend this notation to matrices $\mathbf{Y} \in \mathbb{Z}_q^{n \times k}$ in the natural way (i.e., columnwise). We sometimes suppress σ when it is clear from context. The following convenient lemma statements are adapted from [CVW18].

Lemma 6.7 ([CVW18, Lemma 3.10], originally [Ajt96, AP11, MP12]). Assume $\mathsf{LWE}_{n,m,q,\sigma}^{\delta}$. There is a PPT algorithm $\mathsf{TrapSam}(1^n, 1^m, q)$ that, on input the modulus $q \ge 2$ and dimensions n, m such that $m \ge 2n \log q$, outputs a matrix \mathbf{A} such that $\mathbf{A} \approx_s \mathcal{U}(\mathbb{Z}_q^{n \times m})$ with security parameter μ , along with a trapdoor τ (referenced in the next lemma).

Lemma 6.8 ([CVW18, Lemma 3.11]). There is a PPT algorithm that, on input $\mathbf{y} \leftarrow \mathbb{Z}_q^n$ and $(\mathbf{A}, \tau) \leftarrow \operatorname{TrapSam}(1^n, 1^m, q, \sigma)$ for some $\sigma \geq 2\sqrt{n \log q}$, outputs a sample from $\mathbf{A}^{-1}(\mathbf{y}, \sigma)$. We will abuse notation and denote this algorithm by $\mathbf{A}^{-1}(\mathbf{y}, \sigma)$.

6.3.6 GGH Encodings

We now recall the GGH encoding [GGH15] construction, as presented in [CVW18]. We make a minor modification to instead sample both trapdoor inverses and errors from *truncated Gaussian distributions* (our modifications are marked in blue).

Construction 6.9. Given as input matrices $\{\mathbf{M}_{i,b} \in \mathbb{Z}_q^{n_{i-1} \times n_i}\}_{i \in [h], b \in \{0,1\}}$, ggh.encode does the following:

- Set $\sigma = 2\sqrt{n \log q}$ and $m_i := 4n_i \log q$.
- Sample $\mathbf{E}_{i,b} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}^{n_{i-1} \times m_i}$. If any $\|\mathbf{E}_{i,b}\|_{\infty} \ge \sigma \sqrt{n}$, set to $\mathbf{0}^{n_{i-1} \times m_i}$.
- Sample $\mathbf{A}_i \in \mathbb{Z}_a^{n_i \times m_i}$ using Lemma 6.7.
- Sample the following using Lemma 6.8:

$$\begin{aligned} \{ \mathbf{D}_{1,b} &:= \mathbf{M}_{1,b} \mathbf{A}_1 + \mathbf{E}_{1,b} \}_{b \in \{0,1\}}, \\ \{ \mathbf{D}_{i,b} &:= \mathbf{A}_{i-1}^{-1} \left(\mathbf{M}_{i,b} \mathbf{A}_i + \mathbf{E}_{i,b}, \sigma \right) \}_{2 \le i \le h-1, b \in \{0,1\}}, \\ \{ \mathbf{D}_{h,b} &:= \mathbf{A}_{h-1}^{-1} \left(\mathbf{M}_{h,b} + \mathbf{E}_{h,b}, \sigma \right) \}_{b \in \{0,1\}}. \end{aligned}$$

If any $\|\mathbf{D}_{i,b}\|_{\infty} \ge \sigma \sqrt{n}$, output \perp . Additionally, if $\mathbf{A}_{i-1}\mathbf{D}_{i,b} \ne \mathbf{M}_{i,b}\mathbf{A}_i + \mathbf{E}_{i,b}$ for any $2 \le i \le h-1$, $b \in \{0,1\}$ or $\mathbf{A}_{h-1}\mathbf{D}_{h,b} \ne \mathbf{M}_{h,b} + \mathbf{E}_{h,b}$ for $b \in \{0,1\}$, output \perp .²⁴

• Output $\{\mathbf{D}_{i,b}\}_{i\in[h],b\in\{0,1\}}$.

We extend the construction to MBPs (as defined in Section 6.3.3) $P = ({\mathbf{M}_{i,b}}_{i \in [h], b \in \{0,1\}}, \mathbf{u}, \mathbf{v})$ via

$$ggh.encode(P) := ggh.encode({\mathbf{M}'_{i,b}}_{i,b})$$
,

where $\mathbf{M}'_{1,b} := \mathbf{u}\mathbf{M}_{1,b}, \ \mathbf{M}'_{h,b} := \mathbf{M}_{h,b}\mathbf{v}$, and for $2 \le i \le h - 1, \ \mathbf{M}'_{i,b} := \mathbf{M}_{i,b}$.

We then prove the following lemma (also proved in [CVW18]), and argue that it has a PV proof of completeness.

Lemma 6.10. Except with probability $2^{-n} \cdot \operatorname{poly}(n)$ over $\{\mathbf{D}_{i,b}\}_{i \in [h], b \in \{0,1\}} \leftarrow \operatorname{ggh.encode}(\{\mathbf{M}_{i,b}\}_{i,b}, \sigma),$ letting $B := \max\{\sigma\sqrt{n}, \max_{i \leq h-1, b} \|\mathbf{M}_{i,b}\|_{\infty}\}$ we have that for all $\mathbf{x} \in \{0,1\}^h$,

$$\left\|\mathbf{D}_{\mathbf{x}} - \mathbf{M}_{\mathbf{x}}\right\|_{\infty} \le h \cdot \left(\prod_{i=1}^{h-1} m_i\right) \cdot B^{h-1} ,$$

where recall that we use the shorthand $\mathbf{M}_{\mathbf{x}}$ to denote the subset product $\prod_{i=1}^{h} \mathbf{M}_{i,x_{i}}$. In fact, if ggh.encode does not output \perp , then the output matrices $\{\mathbf{D}_{i,b}\}$ satisfy the above inequality with probability 1. There is a PV proof of the fact that:

for all randomness r, if ggh.encode({ $\mathbf{M}_{i,b}$ }_{$i \in [h], b \in \{0,1\}$}; r) does not output \bot , then { $\mathbf{D}_{i,b}$ }_{$i \in [h], b \in \{0,1\}$} \leftarrow ggh.encode({ $\mathbf{M}_{i,b}$ }_{$i \in [h], b \in \{0,1\}$}; r) satisfies:

$$\|\mathbf{D}_{\mathbf{x}} - \mathbf{M}_{\mathbf{x}}\|_{\infty} \le h \cdot \left(\prod_{i=1}^{h-1} m_i\right) \cdot B^{h-1}.$$

We outline the proof in Appendix A.

6.3.7 σ -PRF Obfuscation

Mathialagan et al. [MPV24] define a notion of σ -PRFs (see Section 4 of [MPV24]), and construct an average-case obfuscation notion for such PRFs.

We first recall the definition of a pseudorandom function (PRF). Below, $O(\cdot)$ denotes a truly random function.

Definition 6.11. A family of deterministic functions $\mathcal{F} := \{f_k : \mathcal{X}_\lambda \to \mathcal{Y}_\lambda\}$ is called pseudorandom if for all PPT adversaries \mathcal{A} ,

$$\left|\Pr_{k,\mathcal{A}}[\mathcal{A}^{f_k(\cdot)}(1^{\lambda})=1] - \Pr_{k,\mathcal{A}}[\mathcal{A}^{O(\cdot)}(1^{\lambda})=1]\right| \le \mathsf{negl}(\lambda) \ .$$

 $^{^{24}}$ Note that this additional check should always pass by the correctness of trapdoor sampling, and should not change the functionality of the algorithm. Checking these equalities enables us to assert these equalities without needing an PV proof for trapdoor sampling.

Mathialagan et al. [MPV24] define a relaxation of the notion of a pseudorandom function with outputs in \mathbb{Z}_q . Informally, it is pseudorandom against adversaries who are only permitted to observe the output values after independent Gaussian noise has been added to each value.

Definition 6.12. Let $q = q(\lambda) \in \mathbb{N}$ and $\sigma = \sigma(\lambda) > 0$. A family of deterministic functions $\mathcal{F} := \{f_k : \mathcal{X}_\lambda \to \mathbb{Z}_{q(\lambda)}\}$ is called σ -pseudorandom if for all PPT adversaries \mathcal{A} ,

$$\left|\Pr_{k,\mathcal{A},O'}[\mathcal{A}^{O'(\cdot)}(1^{\lambda})=1] - \Pr_{k,\mathcal{A}}[\mathcal{A}^{O(\cdot)}(1^{\lambda})=1]\right| \le \mathsf{negl}(\lambda) \ ,$$

where the function O' is chosen by sampling discrete Gaussian errors $\{e_{\mathbf{x}} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}\}_{\mathbf{x}\in\mathcal{X}_{\lambda}}$, and on input $\mathbf{x}\in\mathcal{X}_{\lambda}$, O' outputs $O'(\mathbf{x}) = f_k(\mathbf{x}) + e_{\mathbf{x}}$.

Given a PPT function aux, we say that \mathcal{F} is additionally *pseudorandom in the presence of* aux (resp. σ -pseudorandom in the presence of aux) if the inequality in Definition 6.11 (resp. Definition 6.12) holds even when the adversary is additionally given aux(k) as input.

We will call a σ -PRF computable by a poly(λ)-sized MBP a σ -matrix PRF.

Now, we recall the σ -PRF obfuscation algorithm and guarantee of Mathialagan et al. [MPV24], restricted to the read-once setting. Looking forward, we only need the version of their result for read-once σ -matrix PRFs. We slightly modify the algorithm in blue so that we can argue an PV proof of completeness.

In the description below, we use the notation:

$$\operatorname{diag}(\mathbf{M},\mathbf{S}) := \begin{pmatrix} \mathbf{M} & \\ & \mathbf{S} \end{pmatrix}.$$

Construction 6.13 (Obfuscation for read-once σ -matrix PRFs.). Given a read-once σ -matrix PRF

$$\{\{\mathbf{M}_{i,b}\}_{i\in[h],b\in\{0,1\}},\mathbf{u},\mathbf{v}\}$$

with auxiliary information aux, algorithm \mathcal{O} does the following:

- 1. Sample $\mathbf{S}_{i,b} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}^{n \times n}$, where $\sigma = 2\sqrt{n}$. If any $\|\mathbf{S}_{i,b}\|_{\infty} \ge \sigma\sqrt{n}$, set $\mathbf{S}_{i,b} = \mathbf{0}^{n \times n}$.
- 2. Set $\widehat{\mathbf{S}}_{i,b} := \operatorname{diag}(\mathbf{M}_{i,b}, \mathbf{S}_{i,b}) \in \mathbb{Z}_q^{(n+w) \times (n+w)}$.
- 3. Set $\mathbf{u}' = (\mathbf{u}^T \mid \mathbf{1}_n)^T \in \mathbb{Z}_q^{(n+w) \times 1}$ and $\mathbf{v}' = (\mathbf{v}^T \mid \mathbf{0}_n)^T \in \mathbb{Z}_q^{(n+w) \times 1}$.
- 4. Set $P = (\{\widehat{\mathbf{S}}_{i,b}\}_{i,b}, \mathbf{u}', \mathbf{v}').$
- 5. Compute $\{\mathbf{D}_{i,b}\}_{i \in [h], b \in \{0,1\}} \leftarrow \mathsf{ggh.encode}(P)$.

Then, given an input $\mathbf{x} \in \{0,1\}^h$, \mathcal{O} . Eval $(\{\mathbf{D}_{i,b}\}_{i \in [h], b \in \{0,1\}}, \mathbf{x})$ computes and outputs $\mathbf{D}_{\mathbf{x}} = \prod_{i=1}^h \mathbf{D}_{i,x_i}$.

Theorem 6.14 ([MPV24, Theorem 4.9]). Let $\sigma \ge \sqrt{2n}$ and $B \ge \sigma \sqrt{n}$, and suppose $\mathcal{F} := \{f_k\}_{k \in \mathcal{K}_{\lambda}}$ is a MBP on inputs of length h with all entries in \mathbf{u} and $\{\mathbf{M}_{i,b}\}_{i \in [h], b \in \{0,1\}}$ bounded by B. Let $P \leftarrow \mathcal{O}(f_k)$ be the output of Construction 6.13 on input f_k , and let $m := 4(n+w) \log q$.

Completeness: For all $k \in \mathcal{K}_{\lambda}$, $\mathbf{x} \in \{0, 1\}^h$,

$$|f_k(\mathbf{x}) - \mathcal{O}.\mathsf{Eval}(P, \mathbf{x})| \le h(mB)^{h-1}$$
,

Moreover, there is a PV proof of the following statement:

for all randomness $r, \mathbf{x} \in \{0,1\}^h$, and $f_k \in \mathcal{F}$, if $\mathcal{O}(f_k; r)$ does not output \perp , then the output $P \leftarrow \mathcal{O}(f_k; r)$ satisfies:

$$|f_k(\mathbf{x}) - \mathcal{O}.\mathsf{Eval}(P)| \le h(mB)^{h-1}$$

 σ -PRF Indistinguishability: Letting $\sigma' = 2^{h^3\lambda} \cdot (n^2\sigma)^{h+2}$, and assuming subexponential LWE and evasive LWE²⁵, if \mathcal{F} is a σ' -matrix PRF such that $\mathsf{poly}(2^{n^{\delta}})$ -time adversaries achieve distinguishing advantage at most $\mathsf{negl}(2^{h^2\lambda})$, then there exists a transparent distribution \mathcal{D} (independent of k) such that for $k \leftarrow \mathcal{K}_{\lambda}$,

$$\mathcal{O}(f_k), \operatorname{aux}(k) \approx_c \mathcal{D}, \operatorname{aux}(k)$$
,

such that $\operatorname{poly}(2^{n^{\delta}})$ -time adversaries achieve distinguishing advantage at most $\operatorname{negl}(2^{h^{2}\lambda})$.²⁶ In particular, \mathcal{D} is the distribution:

$$\mathcal{U}(\mathbb{Z}_q^{1\times m}), \{\mathcal{D}_{\mathbb{Z},\sigma}^{m\times m}\}_{i\in[h],b\in\{0,1\}}, \{\mathcal{D}_{\mathbb{Z},\sigma}^{m\times 1}\}.$$

Proof (sketch). We refer the reader to [MPV24] for the proof of indistinguishability. We note that our modifications to both Construction 6.9 and Construction 6.13 only affect the distribution of the output by $2^{-n} \cdot \text{poly}(n)$ (by Lemma 3.1).

Now, we argue that there is a polynomial-sized extended Frege proof of completeness. First, we note that:

$$\mathbf{u}^{T}\widehat{\mathbf{S}}_{\mathbf{x}}\mathbf{v}^{\prime} = (\mathbf{u}^{T}|\mathbf{1}_{n}) \cdot \operatorname{diag}(\mathbf{M}_{\mathbf{x}}, \mathbf{S}_{\mathbf{x}}) \cdot (\mathbf{v}^{T}|\mathbf{0}_{n})^{T} = \mathbf{u}^{T}\mathbf{M}_{\mathbf{x}}\mathbf{v} = f_{k}(\mathbf{x}).$$

Here, we used basic associative properties of matrix multiplication, which have PV proofs.

By construction, $\|\mathbf{S}_{i,b}\|_{\infty} \leq \sigma \sqrt{n}$. Hence, $\|\mathbf{S}_{i,b}\|_{\infty} \leq B$, and by the completeness of ggh.encode (Lemma 6.10), we have that for all \mathbf{x} , $P = \{\mathbf{D}_{i,b}\}_{i \in [h], b \in \{0,1\}}$ satisfies:

$$\|\mathbf{D}_{\mathbf{x}}^{(k)} - \mathbf{u}^{T} \widehat{\mathbf{S}}_{\mathbf{x}} \mathbf{v}^{T}\|_{\infty} \le h(mB)^{h-1}.$$

The above follows from Lemma 6.10 via a PV proof.

Combining the two equations (and corresponding PV proofs), we have

$$|f_k(\mathbf{x}) - \mathcal{O}.\mathsf{Eval}(P)| \le h(mB)^{h-1}$$
,

as desired.

 $^{^{25}}$ We assume evasive LWE for specific distributions used in the sampling procedure in Construction 6.9. See [MPV24] for explicit details on these distributions.

²⁶There exist heuristic counterexamples to evasive LWE [VWW22]. See the discussion in [MPV24] for a discussion on why their regime avoids this for "natural" choices of aux. Looking forward, our choice of aux will be public coin.

6.3.8 Witness Encryption from Evasive LWE with PV Proof

In this section, we construct a witness encryption scheme with a PV proof of completeness from evasive LWE. Since witness encryption gives a 2-message argument system with a short prover message, we can then use Theorem 4.1 to construct a SNARG for 3SAT.

Construction of witness encryption. In this section, we assume subexponential LWE with security parameter $\mu = 2^{n^{\delta}}$.

Construction 6.15. Given a 3SAT formula Φ on h variables and $c \leq h^3$ clauses (if it has more clauses, delete duplicate clauses), and create a read-once branching program $\{\mathbf{u}, \{\mathbf{M}_{i,b}\}_{i \in [h], b \in \{0,1\}}\} \leftarrow CNFEncode(\Phi)$ following Construction 6.5.

- $\operatorname{Enc}(1^{\lambda}, \Phi, m)$:
 - Choose *n* such that $n^{\delta/10} \ge h^3 \lambda$. Set $q = 2^{n^{\delta}}$ and $\sigma = \sqrt{2n}$.
 - Sample $\mathbf{S}_{i,b} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}^{n \times n}$, and $\mathbf{v} \leftarrow \mathbb{Z}_q^{cn}$,

$$\mathbf{u}' = \left(\mathbf{u}^T \otimes (1 \ 0 \dots \ 0) | m \cdot \left\lfloor \frac{q}{2} \right\rfloor\right) \in \mathbb{Z}_q^{cn+1}$$
$$\widehat{\mathbf{S}}_{i,b} = \operatorname{diag}\left(\mathbf{M}_{i,b} \otimes \mathbf{S}_{i,b}, 1\right) \in \mathbb{Z}_q^{(cn+1) \times (cn+1)}$$
$$\mathbf{v}' = (\mathbf{v}^T | 1)^T \in \mathbb{Z}_q^{cn+1}.$$

- Apply the obfuscation in Construction 6.13 to obtain $P \leftarrow \mathcal{O}(\{\widehat{\mathbf{S}}_{i,b}\}_{i \in [h],b}, \mathbf{u}', \mathbf{v}')$.
- If \mathcal{O} outputs \perp , output (\perp, m) . Else, output P.
- Dec(ct, $\mathbf{w} \in \{0,1\}^h$) : If ct is of the form (\perp, m) , output m. Else, if it is a program $P = \{\mathbf{D}_{i,b}\}_{i \in [h], b \in \{0,1\}}$,
 - If $\Phi(\mathbf{w}) = 0$, output \perp .
 - Else, if $|\mathcal{O}.\mathsf{Eval}(P) \pmod{q}| \le q/4$, output 0. Else, output 1.

Proof of Theorem 6.1. We now prove that Construction 6.15 has a PV proof of completeness (Lemma 6.16) and is semantically secure (Lemma 6.17). Then, combined with Theorem 6.4, we achieve the main theorem.

Lemma 6.16. Construction 6.15 is a perfectly complete witness encryption scheme with a polynomialsized Extended Frege proof of completeness, i.e.

for all r, if $\mathcal{R}_{\mathcal{L}}(x, w) = 1$ then $\mathsf{Dec}(\mathsf{Enc}(1^{\lambda}, \Phi, m; r), \mathbf{w}) = m$.

Proof (sketch). First, if $\mathsf{Enc}(1^{\lambda}, \Phi, m; r) = (\bot, m)$, then $\mathsf{Dec}(\mathsf{Enc}(1^{\lambda}, \Phi, m; r), \mathbf{w}) = m$, as desired. Suppose $\mathsf{Enc}(1^{\lambda}, \Phi, m; r) \neq (\bot, m)$, then $\mathcal{O}(\cdot)$ did not output \bot . By Theorem 6.14, we have

$$|\mathbf{u}'^T \mathbf{\widehat{S}}_{\mathbf{w}} \mathbf{v}' - \mathcal{O}.\mathsf{Eval}(P, \mathbf{w})| \le h(4(n+c)\log q \cdot B)^{h-1}.$$

By Lemma 6.6, if $\Phi(\mathbf{w}) = 1$, then, $\mathbf{u}^T \mathbf{M}_{\mathbf{w}} = \mathbf{0}$ and this fact has a polynomial-sized extended Frege proof. Therefore,

$$\begin{aligned} \mathbf{u}^{T} \widehat{\mathbf{S}}_{\mathbf{w}} \mathbf{v}^{\prime} &= \mathbf{u}^{T} \left(\prod_{i=1}^{h} \operatorname{diag}(\mathbf{M}_{i,w_{i}} \otimes \mathbf{S}_{i,w_{i}}, 1) \right) \mathbf{v}^{\prime} \\ &= \mathbf{u}^{T} \cdot \operatorname{diag} \left(\prod_{i=1}^{h} \left(\mathbf{M}_{i,w_{i}} \otimes \mathbf{S}_{i,w_{i}} \right), 1 \right) \cdot \mathbf{v}^{\prime} \\ &= \mathbf{u}^{T} \cdot \operatorname{diag}(\mathbf{M}_{\mathbf{w}} \otimes \mathbf{S}_{\mathbf{w}}, 1) \cdot \mathbf{v}^{\prime} \\ &= \left(\mathbf{u}^{T} \otimes (1 \ 0 \dots \ 0) | m \cdot \left\lfloor \frac{q}{2} \right\rfloor \right) \cdot \operatorname{diag}(\mathbf{M}_{\mathbf{w}} \otimes \mathbf{S}_{\mathbf{w}}, 1) \cdot (\mathbf{v}^{T} | 1)^{T} \\ &= (\mathbf{u}^{T} \otimes (1 \ 0 \dots \ 0)) \cdot (\mathbf{M}_{\mathbf{w}} \otimes \mathbf{S}_{\mathbf{w}}) \cdot \mathbf{v} + m \cdot \left\lfloor \frac{q}{2} \right\rfloor \\ &= (\mathbf{u} \mathbf{M}_{\mathbf{w}} \otimes (1 \ 0 \dots \ 0) \cdot \mathbf{S}_{\mathbf{w}}) \cdot \mathbf{v} + m \cdot \left\lfloor \frac{q}{2} \right\rfloor \\ &= (\mathbf{0} \otimes (1 \ 0 \dots \ 0) \cdot \mathbf{S}_{\mathbf{w}}) \cdot \mathbf{v} + m \cdot \left\lfloor \frac{q}{2} \right\rfloor \\ &= m \cdot \left\lfloor \frac{q}{2} \right\rfloor. \end{aligned}$$

Here, we relied on the distributive and associative laws of matrix multiplication and the mixed tensor property, all of which have a polynomial sized Extended Frege proof.

By our choice of parameters, we have that $q/8 \ge h(4(n+c)\log q \cdot B)^{h-1}$. Then, if m = 0, then $\mathcal{O}.\mathsf{Eval}(P, \mathbf{w}) \le q/8$, and $\mathsf{Dec}(P, \mathbf{w}) = 0$. If m = 1, then

$$\left\lfloor \frac{q}{2} \right\rfloor - q/8 \le \mathcal{O}.\mathsf{Eval}(P) \le \left\lfloor \frac{q}{2} \right\rfloor + q/8$$

and hence Dec(P, w) = 1. Hence, there is a PV proof of completeness of the scheme.

Lemma 6.17. Construction 6.15 is a semantically secure witness encryption scheme assuming subexponential LWE and evasive LWE.

Proof. Fix an unsatisfiable CNF formula Φ on c clauses and h variables. Denote by

$$f_{\mathsf{sk},m}(\mathbf{w}) = \left(\mathbf{u}^T \otimes (1 \ 0 \dots \ 0) | m \cdot \left\lfloor \frac{q}{2} \right\rfloor\right) \widehat{\mathbf{S}}_{\mathbf{w}}(\mathbf{v}^T | 1)^T$$

where $sk = \{\{\mathbf{S}_{i,b}\}_{i,b}, \mathbf{v}\}\$ is sampled as in the encryption algorithm in Construction 6.15.

Claim 6.18. The family $\mathcal{F} = \{f_{\mathsf{sk},m}\}$ is a σ' -PRF family with auxiliary information $\mathsf{aux}_{\mathsf{sk},m} = m$ if Φ is unsatisfiable where $\sigma' = 2^{3h^3\lambda} \cdot (n^2\sigma)^{h+2}$.

Proof. It suffices to show that for $\{e_{\mathbf{w}} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma'}\}_{\mathbf{w} \in \{0,1\}^h}$:

$$\left\{ \left(\mathbf{u}^T \otimes (1 \ 0 \dots \ 0) | m \cdot \left\lfloor \frac{q}{2} \right\rfloor \right) \widehat{\mathbf{S}}_{\mathbf{w}} (\mathbf{v}^T | 1)^T + e_{\mathbf{w}} \right\}_{\mathbf{w} \in \{0,1\}^h}, m \approx_c \{ \mathcal{U}(\mathbb{Z}_q) \}_{\mathbf{w} \in \{0,1\}^h}, m \in \{\mathcal{U}(\mathbb{Z}_q)\}_{\mathbf{w} \in \{0,1\}^h}, m$$

Choose $\sigma'' = 2^{h^3\lambda} \cdot (n^2\sigma)^{h+1}$ (as in Lemma 3.3). Now, we make the following indistinguishability arguments (where each indistinguishability holds against $\operatorname{poly}(2^{n^{\delta}})$ adversaries with probability

$$\operatorname{negl}(2^{-h^2\lambda})).$$

$$\left\{ \left(\mathbf{u}^T \otimes (1 \ 0 \dots \ 0) | m \cdot \left\lfloor \frac{q}{2} \right\rfloor \right) \widehat{\mathbf{S}}_{\mathbf{w}} (\mathbf{v}^T | 1)^T + e_{\mathbf{w}} \right\}_{\mathbf{w} \in \{0,1\}^h}$$
(16)

$$= \left\{ \left(\mathbf{u}^T \mathbf{M}_{\mathbf{w}} \otimes (1 \ 0 \dots \ 0) \right) \left(\mathbf{I}_{cn} \otimes \mathbf{S}_{\mathbf{w}} \right) \cdot \mathbf{v} + m \cdot \left\lfloor \frac{q}{2} \right\rfloor + e_{\mathbf{w}} \right\}_{\mathbf{w} \in \{0,1\}^h}$$
(17)

$$\approx_{s} \left\{ \left(\mathbf{u}^{T} \mathbf{M}_{\mathbf{w}} \otimes (1 \ 0 \dots \ 0) \right) \left[\left(\mathbf{I}_{cn} \otimes \mathbf{S}_{\mathbf{w}} \right) \cdot \mathbf{v} + \mathbf{e}'_{\mathbf{w}} \right] + m \cdot \left\lfloor \frac{q}{2} \right\rfloor + e_{\mathbf{w}} \right\}_{\mathbf{w} \in \{0,1\}^{h}}, \quad \mathbf{e}'_{\mathbf{w}} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma''}^{cn} \quad (18)$$

$$\approx_{c} \left\{ \left(\mathbf{u}^{T} \mathbf{M}_{\mathbf{w}} \otimes (1 \ 0 \dots \ 0) \right) \mathcal{U}_{\mathbf{w}} + m \cdot \left\lfloor \frac{q}{2} \right\rfloor + e_{\mathbf{w}} \right\}_{\mathbf{w} \in \{0,1\}^{h}}, \qquad \qquad \mathcal{U}_{\mathbf{w}} \leftarrow \mathbb{Z}_{q}^{cn} \quad (19)$$

$$= \left\{ \mathcal{U}'_{\mathbf{w}} + m \cdot \left\lfloor \frac{q}{2} \right\rfloor + e_{\mathbf{w}} \right\}_{\mathbf{w} \in \{0,1\}^{h}}, \qquad \qquad \mathcal{U}'_{\mathbf{w}} \leftarrow \mathbb{Z}_{q} \quad (20)$$

$$= \{\mathcal{U}''_{\mathbf{w}}\}_{\mathbf{w}\in\{0,1\}^h}, \qquad \qquad \mathcal{U}''_{\mathbf{w}}\leftarrow\mathbb{Z}_q \quad (21)$$

We prove each step as follows:

- (16) = (17) by linear algebra.
- (17) \approx_s (18) by the following noise-flooding argument. Here, we use the fact that

$$|(\mathbf{u}\mathbf{M}_{\mathbf{w}}\otimes(1\ 0\ldots\ 0))\mathbf{e}'_{\mathbf{w}}| \leq cn\cdot\sigma''\sqrt{n}$$

for all $\mathbf{w} \in \{0,1\}^h$ with $1 - 2^h \cdot \mathsf{negl}(2^{-n})$ probability by Lemma 3.1. Then, we use Lemma 3.2 to argue that the statistical distance is at most $2^h \cdot (cn \cdot \sigma''\sqrt{n})/\sigma' \leq 2^{-h^2\lambda}$.

• (18) \approx_c (19) by applying Lemma 3.3 to obtain that:

$$\{(\mathbf{I}_{cn}\otimes\mathbf{S}_{\mathbf{w}})\cdot\mathbf{v}+\mathbf{e}'_{\mathbf{w}}\}_{\mathbf{w}\in\{0,1\}^{h}}\approx_{c}\{\mathcal{U}(\mathbb{Z}_{q})\}_{\mathbf{w}\in\{0,1\}^{h}}.$$

- (19) = (20) since for all $\mathbf{w}, \mathbf{u}^T \mathbf{M}_{\mathbf{w}} \neq \mathbf{0}$.
- (20) = (21) by a one-time pad argument.

Therefore, we indeed have the desired σ' -PRF indistinguishability, since we do not use $aux_{sk,m} = m$ in (21).

Recall that WE.Enc Construction 6.15 in fact outputs $\mathcal{O}(f_{\mathsf{sk},m})$ as an encryption of m, where sk . Therefore, combining Claim 6.18 with the obfuscation guarantee from Theorem 6.14, we have that the following distributions are indistinguishable:

- H_0 : Sample randomness r, and bit $m \in \{0, 1\}$. Compute $\mathsf{ct} \leftarrow \mathsf{WE}.\mathsf{Enc}(\Phi, m; r)$. Output the pair (ct, m) .
- H_1 : Sample $\mathsf{ct} \leftarrow \mathcal{D}$ and $m \in \{0, 1\}$. Output the pair (ct, m) .

Consider an adversary \mathcal{A} breaks the semantic security of the witness encryption scheme, i.e.

$$\Pr_{m,r}[\mathcal{A}[\mathsf{WE}.\mathsf{Enc}(1^{\lambda},\Phi,m;r)] = m] = \frac{1}{2} + \rho(\lambda)$$

for some non-negligible function ρ . We construct \mathcal{A}' to distinguish H_0 and H_1 as follows:

- Given (ct, m) , feed ct to \mathcal{A} .
- If \mathcal{A} outputs m, output 0. Else, output 1.

Clearly, if \mathcal{A}' was interacting with H_0 , then

$$\Pr[\mathcal{A}'(1^{\lambda}, (\mathsf{ct}, m))] = \Pr[\mathcal{A}(\mathsf{WE}.\mathsf{Enc}(1^{\lambda}, m)) = m] = \frac{1}{2} + \rho(\lambda).$$

If \mathcal{A}' was interacting with H_1 , then ct is indepedent of m, so $\Pr[\mathcal{A}'(1^{\lambda}, (ct, m))] = 1/2$. Therefore, \mathcal{A}' distinguishes H_0 and H_1 with non-negligible probability, leading to a contradiction.

Putting together Lemma 6.16 and Lemma 6.17, we have Theorem 6.1.

6.4 SNARGs for QR and DCR from LWE.

Finally, we consider *unconditional, statistically secure* witness encryption schemes for specific NP languages. By observing that some such schemes have PV (indeed, PV) proofs of correctness, we obtain SNARGs for these restricted languages assuming only LWE. We remark (as discussed in Section 2.4.2) that we depart from standard definitions here by considering witness encryption schemes for *randomized*, AM verifiers, in order to obtain PV proofs of correctness.

Specifically, we consider particular (randomized) witness relations for quadratic (non-)residuosity ($\overline{\mathsf{QR}}$ and QR) and Nth non-residuosity ($\overline{\mathsf{DCR}}$). These are witness relations corresponding to the Goldwasser-Micali and Paillier public-key encryption schemes, respectively.

Definition 6.19 (QR and with specific witness relations). *Define the promise language* $QR_{YES}, QR_{NO} \subset \{(N, y \in \mathbb{Z}_N^{\times})\}$ as follows:

- Promise: $N = pq \leq 2^n$ is a product of two distinct primes that are 3 (mod 4) and the Jacobi symbol $(\frac{y}{N}) = 1.^{27}$
- Yes instances: y is a perfect square modulo N.
- No instances: y is not a perfect square modulo N.

We define an NP witness relation R_{QR} for this promise language as follows:

- Witness: (p,q).
- Verification: p, q are primes, $p \cdot q = N$, and $y^{\frac{p-1}{2}} \equiv 1 \pmod{p}$.

Finally, we define an additional AM verifier \tilde{V}_{QR} for R_{QR} as follows:

- Witness: (p,q).
- Random coins: $g_1, \ldots, g_n \leftarrow \{0, 1\}^{2n}$, interpreted to represent (nearly uniform) elements of \mathbb{Z}_N^{\times} .
- Verification: p, q are primes, $p \cdot q = N$, $gcd(g_i, N) \neq 1$ or $g_i^{p-1} \equiv 1 \pmod{p}$ for all i, and $y^{\frac{p-1}{2}} \equiv 1 \pmod{p}$.

²⁷The latter Jacobi symbol restriction can be removed; we include the restriction for simplicity.

This AM verifier decides the same relation R_{QR} as the above NP verifier.

Next, we define a witness relation $R_{\overline{\text{OR}}}$ for the complement of this promise language as follows:

- Witness: (p,q).
- Verification: p, q are primes, $p \cdot q = N$, and $y^{\frac{p-1}{2}} \equiv -1 \pmod{p}$.

As before, we define an additional AM verifier $\tilde{V}_{\overline{\text{OR}}}$ for $R_{\overline{\text{OR}}}$:

- Witness: (p,q).
- Random coins: $g_1, \ldots, g_n \leftarrow \{0, 1\}^{2n}$, interpreted to represent (nearly uniform) elements of \mathbb{Z}_N^{\times} .
- Verification: p, q are primes, $p \cdot q = N$, either $gcd(g_i, N) \neq 1$ or $g_i^{p-1} \equiv 1 \pmod{p}$ for all i, and $y^{\frac{p-1}{2}} \equiv -1 \pmod{p}$.

Definition 6.20 ($\overline{\text{DCR}}$ with specific witness relation). *Define the promise language* DCR_{YES} , $\text{DCR}_{\text{NO}} \subset \{(N, y \in \mathbb{Z}_{N^2}^{\times})\}$ as follows:

- Promise: N = pq is a product of two distinct odd primes.
- Yes instances: y is an Nth power modulo N^2 .
- No instances: y is not an Nth power modulo N^2 .

We define a witness relation $R_{\overline{\text{DCR}}}$ for the complement of this promise language as follows:

- Witness: (p,q).
- Verification: p, q are primes, $p \cdot q = N$, and $y^{(p-1)(q-1)} \not\equiv 1 \pmod{N^2}$.

Finally, we define an additional AM verifier \tilde{V}_{DCR} for R_{DCR} as follows:

- Witness: (p,q).
- Random coins: $g_1, \ldots, g_n \leftarrow \{0, 1\}^{2n}$, interpreted to represent (nearly uniform) elements of $\mathbb{Z}_{N^2}^{\times}$.
- Verification: p, q are primes, $p \cdot q = N$, either $gcd(g_i, N) \neq 1$ or $g_i^{N(p-1)(q-1)} \equiv 1 \pmod{N^2}$ for all i, and $y^{(p-1)(q-1)} \not\equiv 1 \pmod{N^2}$.

We now give witness encryption schemes for $\tilde{V}_{\bar{R}_{QR}}, \tilde{V}_{R_{QR}}$, and $\tilde{V}_{\bar{R}_{DCR}}$ that have *PV* proofs of decryption correctness. We define a witness encryption scheme for an AM verifier (as opposed a NP verifier) to allow the encryption algorithm access to the AM verifier's random coins.

Lemma 6.21. The AM verifiers $\tilde{V}_{\bar{R}_{QR}}$, $\tilde{V}_{R_{QR}}$, and $\tilde{V}_{\bar{R}_{DCR}}$ admit statistically secure witness encryption schemes with PV proofs of decryption correctness.

Proof. We first define the witness encryption scheme (Goldwasser-Micali encryption) for $\bar{V}_{\bar{R}_{QR}}$:

- Enc $(N, y, g_1, g_2, \ldots, g_n, m)$: if some $g_i \notin \mathbb{Z}_N^{\times}$, output (\perp, m) . Otherwise, for message $m \in \{0, 1\}$, sample $r \equiv \prod_{i=1}^n g_i^{s_i} \pmod{N}$ for uniformly random exponents s_1, \ldots, s_n and output $r^2 \cdot y^m$.
- Dec(p,q,c): if $c = (\perp,m)$ output m. Otherwise, output m = 0 if and only if $c^{\frac{p-1}{2}} \equiv 1 \pmod{p}$.

Proof of Correctness: The correctness property we would like to prove is described as follows:

$$\forall N, y, g_1, g_2, \dots, g_n, p, q, m, s: \text{ if } \widetilde{V}_{\overline{\mathsf{QR}}}(N, y, g_1, g_2, \dots, g_n, p, q) = 1$$

then $\mathsf{Dec}(p, q, \mathsf{Enc}(N, y, g_1, \dots, g_n, m; s)) = m.$

Whenever y is not a square modulo N (and the promise holds), it is the case that $y^{\frac{p-1}{2}} \equiv -1 \mod p$. Therefore, for every $r \in \mathbb{Z}_N^{\times}$, if $y^{\frac{p-1}{2}} \not\equiv 1 \pmod{p}$ (guaranteed by the assumption that $(N, y, p, q) \in \overline{R}_{QR}$), we have

$$(r^2 y^m)^{\frac{p-1}{2}} \equiv r^{p-1} \cdot (y^{\frac{p-1}{2}})^m \equiv r^{p-1} \cdot (-1)^m \pmod{p}.$$

As this only makes use of basic properties of arithmetic modulo p, this fact can be proved in PV (and therefore \mathcal{EF}). In addition, we are promised that

$$r \equiv \prod_{i=1}^{n} g_i^{s_i} \pmod{N},$$

and, by assumption on the validity of the witness, we know that $g_i^{p-1} \equiv 1 \pmod{p}$ for every *i*. Therefore, since $p \mid N$, we conclude that

$$r^{p-1} \equiv \prod_{i=1}^{n} (g_i^{s_i})^{p-1} \equiv \prod_{i=1}^{n} (g_i^{p-1})^{s_i} \equiv 1 \pmod{p}.$$

Thus, we can prove in PV that $Dec(p, Enc(N, y, g_1, g_2, \ldots, g_n, m; s)) = m$ for all valid witnesses (p, q).

Proof of Security. Security holds provided that the sampled group element r is uniform on the set \mathbb{Z}_N^{\times} . By [Pom02], we know that with all but negligible probability over the choice of g_1, \ldots, g_n , the set $\{g_1, \ldots, g_n\}$ generates \mathbb{Z}_N^{\times} ; under this condition, we conclude that r is indeed uniformly random from \mathbb{Z}_N^{\times} and we are done.

Extension to R_{QR} . Next, we observe that the same result holds for $\tilde{V}_{R_{QR}}$: since we are promised that $p, q \equiv 3 \pmod{4}$, we will make use of the fact that multiplication of y by -1 preserves the promise and switches QR_{YES} and QR_{NO} . We can therefore define the encryption scheme $Enc'(N, y, g_1, \ldots, g_n, m; s) = Enc(N, -y, g_1, \ldots, g_n, m; s)$ and obtain a similar PV proof of decryption correctness, additionally making use of the fact that $(-1)^{\frac{p-1}{2}} \equiv 1 \pmod{p}$.

Witness Encryption for $\tilde{V}_{R_{\overline{\text{DCR}}}}$. Finally, for $\tilde{V}_{R_{\overline{\text{DCR}}}}$, we will use a modification of the Paillier encryption scheme:

- $\operatorname{Enc}(N, y, g_1, \dots, g_n, m)$: if some $g_i \notin \mathbb{Z}_N^{\times}$, output (\perp, m) . Otherwise, for message $m \in \{0, 1\}$, sample $r \equiv \prod_{i=1}^n g_i^{s_i} \pmod{N^2}$ and output $r^N \cdot (1+N)^m$.
- Dec(p,q,c): if $c = (\perp, m)$, output m. Otherwise, output m = 0 if and only if $c^{(p-1)(q-1)} \equiv 1 \pmod{N^2}$.

Similarly to the case of QR, correctness of decryption (in the case that $y^{(p-1)(q-1)} \not\equiv 1 \pmod{N^2}$) follows from the fact that

$$c^{(p-1)(q-1)} \equiv r^{N(p-1)(q-1)}(1+N)^{m(p-1)(q-1)} \equiv r^{N(p-1)(q-1)}(1+(m(p-1)(q-1))N) \pmod{N^2}.$$

As before, we can conclude in PV that $r^{N(p-1)(q-1)} \equiv 1 \pmod{N^2}$ from our hypothesis on (p, q, g_1, \ldots, g_n) . Finally, $m \cdot (p-1)(q-1)$ is in the open interval (1, N), and therefore $c^{(p-1)(q-1)} \pmod{N^2}$ is equal to 1 if and only if $m \cdot (p-1)(q-1) = 0$, which is true if and only if m = 0.

Security holds, again as before, because with overwhelming probability over $g_1, \ldots, g_n, \{g_1, \ldots, g_n\}$ generates $\mathbb{Z}_{N^2}^{\times}$ [Pom02], and therefore the sampled string r will be uniformly random over $\mathbb{Z}_{N^2}^{\times}$. \Box

Finally, we briefly remark that Theorem 6.4 can be quickly modified to handle our "AM witness encryption schemes:" simply include the AM random coins as part of the (unencrypted) crs for the SNARG. The resulting SNARG is defined for the NP witness relations $R_{\overline{QR}}$, R_{QR} , $R_{\overline{DCR}}$ because the NP and AM verifiers define the same set of true/false statements.

7 Application II: Transparent and Adaptive SNARGs from iO and LWE

In this section, we show how to construct a transparent non-adaptive SNARG for NP as well as a adaptive SNARG for NP from subexponential iO and subexponential LWE.

Theorem 7.1. Assuming the existence of subexponentially secure iO and LWE, there exists a transparent, non-adaptively sound SNARG for all of NP.

Theorem 7.2. Assuming the existence of subexponentially secure iO and LWE, there exists an adaptively sound SNARG for all of NP.

We remark that the above theorems do *not* explicitly assume the existence of an iO scheme that has a PV proof of correctness. Instead, we show how to generically upgrade an subexponentially secure iO scheme into one that has a PV proof of correctness.

Theorem 7.3. Assuming subexponentially-secure iO and subexponential LWE, there exists an iO scheme with a PV proof of correctness.

Organization of this section. We begin with definitions of indistinguishability obfuscation and functional encryption, which are the main tools used in this section. Next, we construct an *inefficient* iO scheme (called XiO [LPST16a]) with a PV proof of correctness using any iO scheme in Section 7.3.1. Then, we use the bootstrapping techniques of [AJ15, BV15, LPST16b, LPST16a] as formalized by [Vai24] to compile this construction into an *efficient* iO scheme. We show that our variant of this transformation preserves PV proofs of correctness.

7.1 Indistinguishability Obfuscation for Circuits

Syntax. An indistinguishability obfuscator iO = (iO.Obf, iO.Eval) for a family of circuits $C = \{C_{\lambda}\}_{\lambda}$ is a pair of algorithms:

- $\mathsf{iO.Obf}(1^{\lambda}, C) \to \Pi$. This is a probabilistic algorithm that takes as input a circuit C and outputs a program Π .
- $\mathsf{iO}.\mathsf{Eval}(\Pi, x) \to y$ This is a deterministic algorithm that takes as input an obfuscated program Π , and input x and outputs y.

Definition 7.4 (Indistinguishability obfuscation (iO), [BGI⁺01]). A tuple of PPT algorithms (iO.Obf, iO.Eval) is an indistinguishability obfuscator iO for a circuit class $C = \{C_{\lambda}\}_{\lambda}$ if it satisfies the following conditions:

• Statistical Correctness: For all security parameters $\lambda \in \mathbb{N}$, for all $C \in \mathcal{C}_{\lambda}$, for all inputs x, we have that

 $\Pr[\Pi \leftarrow \mathsf{iO.Obf}(1^{\lambda}, C) : \mathsf{iO.Eval}(\Pi, x) = C(x)] = 1 - \mathsf{negl}(\lambda).$

We say that iO satisfies sub-exponential statistical correctness if the correctness error is $O(2^{-\lambda^{\alpha}})$ for some universal constant $\alpha > 0$.

• Polynomial Security: For all pairs of circuits $C_0, C_1 \in C_\lambda$ such that $C_0(x) = C_1(x)$ for all x, we have that for all PPT adversaries \mathcal{A} ,

 $\left| \Pr[\mathcal{A}(\mathsf{iO.Obf}(1^{\lambda}, C_0) = 1] - \Pr[\mathcal{A}(\mathsf{iO.Obf}(1^{\lambda}, C_1) = 1] \right| = \mathsf{negl}(\lambda) \;.$

We say that the iO is sub-exponentially secure if, for some $\alpha > 0$, all $2^{\lambda^{\alpha}}$ -time adversaries have distinguishing advantage at most $2^{-\lambda^{\alpha}}$ for some $\alpha > 0$.

7.2 Single-Key Functional Encryption

Syntax. A functional encryption scheme FE for a class of functions $\mathcal{F} = \{\mathcal{F}_n\}_{n \in \mathbb{N}}$ represented as boolean circuits with an *n*-bit input and *m*-bit output, is a tuple of four algorithms (FE.Setup, FE.KeyGen, FE.Enc, FE.Dec) such that:

- $\mathsf{FE.Setup}(1^{\lambda}) \to (\mathsf{mpk}, \mathsf{msk})$. Takes as input a security parameter 1^{λ} and outputs a master public key mpk and a master secret key msk .
- $\mathsf{FE}.\mathsf{KeyGen}(\mathsf{msk}, f) \to \mathsf{fsk}_f$. Takes as input the master secret key msk and a function $f \in \mathcal{F}$ and outputs a key fsk_f .
- $\mathsf{FE}.\mathsf{Enc}(\mathsf{mpk}, x) \to c$. Takes as input the master public key mpk and an input $x \in \{0, 1\}^*$ and outputs a ciphertext c.

 $\mathsf{FE}.\mathsf{Dec}(\mathsf{fsk}_f, c) \to y$. Takes as input a key fsk_f and a ciphertext c and outputs a value y.

Definition 7.5 (Functional encryption). A tuple of PPT algorithms (FE.Setup, FE.KeyGen, FE.Enc, FE.Dec) is a functional encryption scheme if it satisfies:

Perfect Correctness. For any polynomial $n(\cdot)$, and for sufficiently large security parameters λ , for $n = n(\lambda)$, for all $f \in \mathcal{F}_n$, and all $x \in \{0, 1\}^n$:

$$\Pr\left[\begin{array}{cc} (\mathsf{mpk},\mathsf{msk}) \leftarrow \mathsf{FE}.\mathsf{Setup}(1^{\lambda}) \\ \mathsf{FE}.\mathsf{Dec}(\mathsf{fsk}_f,c) = f(x) & : & \mathsf{fsk}_f \leftarrow \mathsf{FE}.\mathsf{KeyGen}(\mathsf{msk},f) \\ & c \leftarrow \mathsf{FE}.\mathsf{Enc}(\mathsf{mpk},x) \end{array}\right] = 1.$$

and one of the two following security properties.

Single-key selective IND-security. For every adversary \mathcal{A} , there exists a negligible function μ such that for every $\lambda \in \mathbb{N}$, for every $f \in \mathcal{F}_{\lambda}$, and every pair of inputs x_0, x_1 , we have

$$|\mathcal{D}_0(\mathcal{A}) - \mathcal{D}_1(\mathcal{A})]| \le \mu(\lambda)$$

where \mathcal{D}_b is the following probability:

$$\mathcal{D}_b(\mathcal{A}) = \Pr\left[\begin{array}{cc} (\mathsf{mpk},\mathsf{msk}) \leftarrow \mathsf{FE}.\mathsf{Setup}(1^\lambda) \\ \mathcal{A}(\mathsf{pk},\mathsf{fsk}_f,x_b) = 1 & : & \mathsf{fsk}_f \leftarrow \mathsf{FE}.\mathsf{KeyGen}(\mathsf{msk},f) \\ & c_b \leftarrow \mathsf{FE}.\mathsf{Enc}(\mathsf{mpk},x_b) \end{array}\right]$$

Full simulation security. For all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a simulator Sim such that the outcomes of Experiment 1 and Experiment 2 are indistinguishable.

	$\textbf{Experiment 2 EXP}_{FE,\mathcal{A}}^{\texttt{Real}}(1^{\lambda})$
$(mpk,msk) \leftarrow FE.Setup(1^{\lambda}).$	$(mpk,msk) \leftarrow FE.Setup(1^{\lambda}).$
$(f, st_{\mathcal{A}}) \leftarrow \mathcal{A}_1(mpk)$	$(f, st_\mathcal{A}) \leftarrow \mathcal{A}_1(mpk)$
$fsk_f \leftarrow FE.KeyGen(fmsk, f)$	$fsk_f \leftarrow FE.KeyGen(fmsk,f)$
$x, st'_{\mathcal{A}} \leftarrow \mathcal{A}_2(st_{\mathcal{A}}, fsk_f)$	$x, st'_{\mathcal{A}} \leftarrow \mathcal{A}_2(st_{\mathcal{A}}, fsk_f)$
$ct \leftarrow FE.Enc(fmpk, x)$	$ct \leftarrow Sim(mpk,fsk_f,f,f(x),1^{ x })$
Output (st'_A, ct) .	Output (st'_A, ct) .

In this work, we make use of the following succinct functional encryption scheme $[GKP^+13]$.

Theorem 7.6 ([GKP⁺13]). Assuming subexponential LWE, for any depth parameter $d = d(\lambda)$, there is a single-key functional encryption scheme sFE for general functions computable by circuits of depth d with output length m = 1 and the following efficiency parameters: assuming the circuit class has size bounded by s and depth bounded by d,

- *Runtime of* sFE.Setup, sFE.KeyGen, sFE.Dec: $poly(\lambda, s, d, n)$.
- Runtime of sFE.Enc: $poly(\lambda, d, n)$.
- Size of ciphertexts: $poly(\lambda, d, n)$.

The scheme satisfies full simulation security.

7.3 Upgrading iO to have a PV proof of correctness

7.3.1 Slow XiO with PV proof of correctness

Given an indistinguishability obfuscation scheme iO, we first show how to give it a PV proof of correctness, at the cost of an inefficient obfuscation procedure (that still produces a small obfuscated circuit).

Algorithm 3 Algorithm slowXiO which has PV proof of correctness for circuits.

slowXiO.Obf $(1^{\lambda}, 1^{TT(C)}, C; r)$: Security parameter 1^{λ} , truth-table size $1^{TT(C)}$ and circuit C.

- Compute $\widetilde{C} \leftarrow iO.Obf(1^{\rho}, C; r)$. Here, we pick parameter ρ here so that $2^{\rho^{\alpha}} > poly(\lambda, TT(C))$. Note that we can pick $\rho = poly(\lambda, |C|)$.
- For all inputs x, check if iO.Eval(C̃, x) = C(x) (note that this can be done in parallel, so the resulting slowXiO circuit has low depth poly(λ, |C|)).
- If any test fails, output (\perp, C) . Else, output C.

slowXiO.Eval (Γ, x) :

- If Γ is of the form (\bot, C) , compute and output C(x).
- Else, output $iO.Eval(\Gamma, x)$.

Lemma 7.7. Suppose that iO is a statistically correct obfuscation scheme. Then, slowXiO (Algorithm 3) is an iO scheme with the following efficiency:

- Running time of slowXiO: $poly(\lambda, |C|, TT(C))$.
- Randomness complexity of slowXiO: $poly(\lambda, |C|)$.
- Depth of slowXiO: poly(λ , |C|) (by the obfuscation efficiency of the underlying iO).
- Output size: $poly(\lambda, |C|)$.

Moreover, slowXiO has sub-exponential security provided that iO is sub-exponentially secure and sub-exponentially statistically correct.

Finally, slowXiO has a PV proof of the fact that:

for all randomness r, all circuits C, if $\Gamma = \text{slowXiO}(1^{\lambda}, 1^{TT(C)}, C; r)$, then for all x, slowXiO.Eval $(\Gamma, x) = C(x)$.

Proof. We show that Algorithm 3 is secure and correct.

Security. First, we show that the resulting scheme is (sub-exponentially) secure if the underlying iO scheme is (sub-exponentially) secure and (sub-exponentially) statistically correct. Note that since iO is statistically correct, the output distribution of slowXiO is statistically close to the output distribution of iO. Therefore, by the correctness and security properties of iO, we have that for all circuits with $C_1 \equiv C_2$:

$$\mathsf{slowXiO}(C_1) \approx_s \mathsf{iO}(C_1) \approx_c \mathsf{iO}(C_2) \approx_s \mathsf{slowXiO}(C_2)$$

as desired. Moreover, $slowXiO(C_1)$ and $slowXiO(C_2)$ are sub-exponentially computationally indistinguishable provided that iO is both sub-exponentially secure and sub-exponentially correct. **PV Proof of Correctness.** We give a sketch of the PV correctness proof below.

- For any security parameter λ , circuit C, and randomness r, let $\widetilde{C} = iO(1^{\lambda}, C; r)$ and let $\Gamma = slowXiO(1^{\lambda}, C; r)$.
- Let $b \in \{0, 1\}$ denote the bit indicating whether $\Gamma = (\bot, C)$.
- Then, by the definition of slowXiO, we know that:

$$\forall C, x, r, \lambda : C(x) \neq C(x) \rightarrow b = 1.$$

The contrapositive of this is that

$$\forall C, x, r, \lambda : b = 0 \to \widetilde{C}(x) = C(x).$$

- We also know (by the definition of slowXiO) that if b = 0, $\Gamma = \widetilde{C}$.
- Therefore, we have that

$$\forall C, x, r, \lambda : \Gamma(x) = b \cdot C(x) + (1-b) \cdot \widetilde{C}(x) = b \cdot C(x) + (1-b) \cdot C(x) = C(x),$$

as desired.

7.3.2 Slow XiO to Fast XiO with CRS

Our next transformation converts the Slow-XiO scheme from Lemma 7.7 into an obfuscation scheme with the following properties:

- The scheme is defined in the common reference string model. Specifically, there is a *slow* setup algorithm $\text{Gen}(1^{\lambda}, 1^{T}, 1^{s})$ producing $(\text{crs}_{\mathcal{O}}, \text{crs}_{\text{Eval}})$ to be used for obfuscation and evaluation, respectively, of circuits with size at most *s* and truth table size at most *T*.
- The running time of the obfuscation procedure (and the length of $\operatorname{crs}_{\mathcal{O}}$) is $\operatorname{poly}(\lambda, |C|)$.
- The length of crs_{Eval} and the running time of Eval is $poly(\lambda, T)$.
- iO security holds for C_0, C_1 chosen *adaptively* by the adversary as a function of the crs.

To do this, we rely on techniques from [GKP⁺13, AJ15, BV15, LPST16a, LPST16b], as formalized by [Vai24]. We include a full exposition and security proof here for completeness.

The transformation requires one new building block: a succinct functional encryption scheme. In Appendix C, we show that the scheme from Theorem 7.6 can be instantiated with a PV proof of correctness, i.e.

for all randomness, for all circuits f and inputs x, if $(mpk, msk) \leftarrow sFE.Setup(1^{\lambda})$, fsk_f $\leftarrow sFE.KeyGen(msk, f)$, $c \leftarrow sFE.Enc(mpk, x)$, then $y \leftarrow FE.Dec(fsk_f, c)$ satisfies y = f(x).

Algorithm 4 (sFE is the succinct functional encryption scheme as guaranteed by Theorem 7.6)

fastXiO.Gen $(1^{\lambda}, 1^{T}, 1^{s})$:

- 1. Construct the low-depth circuit O representing $slowXiO(1^{\lambda}, 1^{T}, \cdot; \cdot)$ from Lemma 7.7.
- 2. Suppose the total input length of O (circuit plus randomness) is $\ell = poly(\lambda, |C|, \log T)$ bits, and the depth of O is $d = poly(\lambda, |C|, \log T)$. Moreover, suppose the output of slowXiO(C) has length $m = poly(\lambda, |C|)$.
- 3. We use the notation O_i to denote the circuit that outputs the *i*th bit of the output of O.
- 4. For each $i \in [m]$, sample master public and secret keys $(\mathsf{mpk}_i, \mathsf{msk}_i) \leftarrow \mathsf{sFE}.\mathsf{Gen}(1^{\lambda}, 1^d, 1^\ell, 1^T).$
- 5. Compute corresponding function keys $\mathsf{fsk}_i \leftarrow \mathsf{sFE}.\mathsf{KeyGen}(\mathsf{msk}_i, O_i)$.
- 6. Output $\operatorname{crs}_{\mathcal{O}} = {\operatorname{mpk}_i}_i \operatorname{in[m]}, \operatorname{crs}_{\mathsf{Eval}} = {\operatorname{fsk}_i}_{i \in [m]}$.

 $\mathsf{fastXiO.Obf}(\mathsf{crs}_{\mathcal{O}} = \{\mathsf{mpk}_i\}_{i \in [m]}, C):$

- 1. Sample randomness $r \leftarrow \{0,1\}^{\ell}$ for slowXiO.
- 2. For each $i \in [m]$, compute $\Gamma_i \leftarrow \mathsf{sFE}.\mathsf{Enc}(\mathsf{mpk}_i, (C, r))$.
- 3. Output $\Gamma = {\Gamma_i}_{i \in [m]}$.

 $\mathsf{fastXiO}.\mathsf{Eval}(\mathsf{crs}_{\mathsf{Eval}} = \{\mathsf{fsk}_i\}_{i \in [m]}, \Gamma = \{\Gamma_i\}_{i \in [m]}, x)$

- 1. For each $i \in M$, compute $c_i \leftarrow \mathsf{sFE}.\mathsf{Dec}(\mathsf{fsk}_i, \Gamma_i)$.
- 2. Compute $\widetilde{C} = (c_1, c_2, \dots c_m)$.
- 3. Output slowXiO.Eval(\widetilde{C}, x).

Lemma 7.8. Algorithm 4 is an indistinguishability obfuscation scheme in the CRS model assuming that slowXiO is a indistinguishability obfuscation scheme with parameters as in Lemma 7.7 and sFE is a single-key succinct functional encryption scheme as in Theorem 7.6.

Moreover, assuming the sFE has a proof of correctness and that slowXiO has a proof of correctness (as in Lemma 7.7), then fastXiO has a PV proof of the following correctness property:

for all randomness r_1, r_2 , then for all circuits C, if $(crs_{\mathcal{O}}, crs_{\mathsf{Eval}}) \leftarrow \mathsf{fastXiO.Gen}(1^{\lambda}, 1^{TT(C)}, 1^{|C|}; r_1)$ and $\Gamma \leftarrow \mathsf{fastXiO.Obf}(crs_{\mathcal{O}}, C; r_2)$, then for all inputs x, $\mathsf{fastXiO.Eval}(crs_{\mathsf{Eval}}, \Gamma, x) = C(x)$.

Proof. We first note that $|crs_{\mathcal{O}}|$ and the running time of fastXiO.Obf are indeed $poly(\lambda, |C|)$ by the underlying efficiency properties of slowXiO and sFE. In particular, the running time of sFE.Enc is $poly(\lambda, d, \ell) = poly(\lambda, |C|)$.

To prove security, we observe that by the simulation-security of sFE, there exists a simulator $Sim(\{mpk_i\}, \{fsk_{O_i}\}, O, \tilde{C}, 1^{poly(\lambda, |C|)})$ such that for all efficient adversaries A and $(C, r) = A(mpk, \{fsk_{O_i}\})$, we have:

{sFE.Enc(mpk_i, (C, r))}_i \approx_c Sim({mpk_i}, {fsk_{O_i}}, O, slowXiO(C; r), 1^{poly(\lambda, |C|)}).

Thus, an adaptive adversary breaking the security of fastXiO implies the existence of an adversary violating the security of slowXiO.

PV Proof. We now sketch the PV proof of correctness for the above proof.

• For any security parameter λ , and randomness r_1 , let

$$\mathsf{crs} := (\{\mathsf{mpk}_i, \mathsf{fsk}_i\}) \leftarrow \mathsf{fastXiO}.\mathsf{Gen}(1^{\lambda}; r_1).$$

- For any randomness r_2 , let $\Gamma := \{\Gamma_i\}_{i \in [m]} \leftarrow \mathsf{fastXiO.Obf}(\mathsf{crs}, C; r_2).$
- By definition of fastXiO.Gen, we have (mpk_i, msk_i) ← sFE.Gen(1^λ), and fsk_i ← sFE.KeyGen(msk_i, U_i) using randomness r₁.
- By definition of fastXiO.Obf, we have $\Gamma_i = sFE.Enc(mpk_i, (C, r_2))$.
- By correctness of sFE, we have that for $y \leftarrow sFE.Dec(fsk_i, sFE.Enc(mpk_i, (C, r_2))$ satisfies $y = O_i(C, r_2) = slowXiO.Obf(C; r_2)_i$.
- Define $\tilde{C} = \mathsf{slowXiO.Obf}(C; r_2)$.
- By substitution, $y = {sFE.Dec(fsk_i, \Gamma_i)}_i$ satisfies $y = \tilde{C}$.
- By substitution, slowXiO.Eval(\tilde{C}, x) = slowXiO.Eval(slowXiO($C; r_2), x$).
- By correctness of slowXiO, we have $slowXiO(\tilde{C}, x) = C(x)$.
- By combining the equalities, we have that for all x, r_1, r_2 , fastXiO.Eval(crs_{Eval}, Γ, x) = C(x).

This completes the proof.

From this point onward, to obtain iO for circuits with a PV proof of correctness, we analyze a sequence of four transformations from the literature [LPST16a, ABSV15, LPST16b] converting any fastXiO into a full-fledged obfuscation scheme, additionally assuming a succinct function encryption scheme sFE and a puncturable pseudorandom function family. In the following sections, we sketch the transformations and observe that under suitable instantiations of the building blocks in these transformations (specifically the sFE and Yao's garbling scheme), the transformations maintain PV proofs of correctness.

7.3.3 Single-Key Compact FE for Bounded-Depth Circuits

First, we construct a single-key FE scheme satisfying IND-security for circuits with bounded depth d, such that FE encryption runs in time that does now grow with m, the output length of the functions supported by the FE. This transformation is due to [LPST16a] and makes use of a succinct functional encryption scheme sFE along with a puncturable PRF family $\mathcal{F} = \{F_k\}$. We will make use of an sFE that has a PV proof of correctness (see Appendix C.3).

Theorem 7.9 ([LPST16a]). Assuming the security of fastXiO and the full simulation security of sFE, the scheme dFE in Algorithm 5 is a selectively IND-secure compact FE with |mpk| and encryption time bounded by $poly(\lambda, d, n, \log m)$.

What remains is to observe the following:

Claim 7.10. If sFE and fastXiO have PV proofs of correctness, then so does dFE.

Algorithm 5 Single-Key Bounded-Depth Compact FE

dFE.Setup $(1^{\lambda}, 1^{n}, 1^{m}, 1^{s}, 1^{d})$:

- 1. Call sFE.Setup $(1^{\lambda}, 1^{n+\log m}, 1^{s+\tilde{O}(m)}, 1^d) \to (\mathsf{mpk}, \mathsf{msk}).$
- 2. Call fastXiO.Gen $(1^{\lambda}, 1^{\log m}, 1^{S}) \rightarrow (\operatorname{crs}_{\mathcal{O}}, \operatorname{crs}_{\mathsf{Eval}})$, where $S = \operatorname{poly}(\lambda, d)$ denotes the size of the circuit family G defined below.
- 3. Output $((\mathsf{mpk}, \mathsf{crs}_{\mathcal{O}}), (\mathsf{msk}, \mathsf{crs}_{\mathsf{Eval}}))$.

 $dFE.KeyGen(msk, crs_{Eval}, C):$

- 1. Call sFE.KeyGen(msk, C') for $C'(x, i) = C(x)_i$.
- 2. Output $\mathsf{sk}_{C'}, \mathsf{crs}_{\mathsf{Eval}}$.

dFE.Enc(mpk, $crs_{\mathcal{O}}, x$):

- 1. Sample a puncturable PRF key k.
- 2. Define the circuit $G[\mathsf{mpk}, k, x]$ to have input $i \in [m]$ and compute $\mathsf{sFE}.\mathsf{Enc}(\mathsf{mpk}, (x, i); F_k(i))$.
- 3. Output fastXiO.Obf($crs_{\mathcal{O}}, G[mpk, k, x]$).

$dFE.Dec(sk_{C'}, crs_{Eval}, ct):$

- 1. Compute $ct_i = fastXiO.Eval(crs_{Eval}, ct, i)$ for all $1 \le i \le m$.
- 2. Compute $y_i = \mathsf{sFE}.\mathsf{Dec}(\mathsf{sk}_{C'}, \mathsf{ct}_i)$ for all $1 \le i \le m$.
- 3. Output y.

This is true by the following (sketched) straightline proof of correctness:

- dFE.Dec(sk_{C'}, crs_{Eval}, ct) = (sFE.Dec(sk_{C'}, ct₁), ..., sFE.Dec(sk_{C'}, ct_m)).
- For every i, ct_i = fastXiO.Eval(crs_{Eval}(fastXiO.Obf(crs_O, G[mpk, k, x]), i) = G[mpk, k, x](i) by the correctness of fastXiO.
- For every *i*, sFE.Dec(sk_{C'}, G[mpk, k, x](*i*)) = C'(x, *i*) = C(x)_i by substitution and the correctness of sFE.
- Thus, y = C(x), as claimed.

7.3.4 Single-Key Compact FE for all circuits

Next, we remove the depth restriction from dFE. This transformation is due to [ABSV15] and makes use of a pseudorandom function family²⁸ as well as Yao's garbling scheme. We will make use of an instantiation of the following building blocks:

• Garbled circuits, instantiated with a PV proof of correctness (see Appendix C.1). For this garbling scheme, define the algorithm $\mathsf{RE}(C, x; r \in \{0, 1\}^{2 \cdot s \cdot \lambda})$ to output the garbled circuit \tilde{C} along with the garbled input keys corresponding to x.

 $^{^{28}}$ In [ABSV15], the goal was to start with FE for NC¹, but here, we are starting with a stronger dFE and can therefore avoid dealing with PRFs in NC¹. In addition, [ABSV15] also wishes to preserve multi-key security, while we only care about single-key security for long-output circuits.

- A pseudorandom function family $\mathcal{F} = \{F_k\}$.
- A secret-key encryption scheme with keys, messages, and pseudorandom ciphertexts (even in the multi-message setting) of length $poly(\lambda)$ (which can be instantiated from PRFs as well). The encryption scheme is then defined on longer messages via block-by-block encryption, so that decryption is computable by a circuit of depth $poly(\lambda)$.

Algorithm 6 Single-Key Compact FE

 $\begin{aligned} \mathsf{FE.Setup}(1^{\lambda}, 1^n, 1^m, 1^s) \text{ calls } \mathsf{dFE.Setup}(1^{\lambda}, 1^{n+2\lambda+1}, 1^{s'=s \cdot \mathsf{poly}(\lambda)}, 1^m) \to (\mathsf{mpk}, \mathsf{msk}). \\ \mathsf{FE.KeyGen}(\mathsf{msk}, C): \end{aligned}$

- 1. Sample a ciphertext $\mathsf{ct}_{\mathsf{SKE}} \leftarrow \{0,1\}^{s \cdot \mathsf{poly}(\lambda)}$ with corresponding plaintext length s' (of a dummy message).
- 2. Derive key $\mathsf{sk}_{C'} = \mathsf{dFE}.\mathsf{KeyGen}(\mathsf{msk}, C')$ for the circuit

$$C'[\mathsf{ct}_{\mathsf{SKE}}](x, k_{\mathsf{PRF}}, k_{\mathsf{SKE}}, \beta) = \begin{cases} \mathsf{Dec}(k_{\mathsf{SKE}}, \mathsf{ct}_{\mathsf{SKE}}) & \text{if } \beta = 1\\ \mathsf{RE}(C, x; F_{k_{\mathsf{PRF}}}(1), F_{k_{\mathsf{PRF}}}(2), \dots, F_{k_{\mathsf{PRF}}}(s')) & \text{otherwise} \end{cases}$$

FE.Enc(mpk, x) samples k_{PRF} and calls dFE.Enc(mpk, $(x, k_{\mathsf{PRF}}, 0^{\lambda}, 0))$. FE.Dec(sk_{C'}, ct):

- 1. Compute $\hat{y} = \mathsf{dFE}.\mathsf{Dec}(\mathsf{sk}_{C'},\mathsf{ct}).$
- 2. Compute and output y to be the garbling scheme decoding of \hat{y} .

Theorem 7.11 ([ABSV15], slightly modified). Assuming the single-key security of dFE, SKE, \mathcal{F} , and the garbling scheme, the scheme FE in Algorithm 6 is a selectively IND-secure single-key compact FE with $|\mathsf{mpk}|$ and encryption time bounded by $\mathsf{poly}(\lambda, n, \log m)$.

The main difference from [ABSV15] is that since dFE supports arbitrary depth d circuits (with an associated encryption cost), it suffices that the circuit C' above has depth $poly(\lambda, n, \log m)$.

What remains is to observe the following:

Claim 7.12. If dFE and the garbling scheme have PV proofs of correctness, then so does FE.

This is true by the following (sketched) straightline proof of correctness:

- FE.Dec($sk_{C'}$, ct) = Eval(dFE.Dec($sk_{C'}$, ct)).
- By the correctness of dFE, we have that dFE.Dec($\mathsf{sk}_{C'}, \mathsf{ct}$) = $C'[\mathsf{ct}_{\mathsf{SKE}}](x, k_{\mathsf{PRF}}, 0^{\lambda}, 0) = \mathsf{RE}(C, x; \rho)$ for $\rho = (F_{k_{\mathsf{PRF}}}(0), \ldots, F_{k_{\mathsf{PRF}}}(s'))$.
- By the correctness of the garbling scheme, we have that $\mathsf{Eval}(\mathsf{RE}(C, x; \rho)) = C(x)$ for all ρ .
- Therefore, $FE.Dec(sk_{C'}, ct) = C(x)$, as claimed.

7.3.5 Output-Compressing Randomized Encodings for Turing Machines

Next, following [LPST16b] Section 6 (Definitions 20, 22, 23), we convert FE into what is called an "output-compressing randomized encoding scheme for Turing machines" (with common reference string). The transformation makes use of a pseudorandom generator G with output length m.

Algorithm 7 Output-Compressing RE for TMs

RE.Setup $(1^{\lambda}, 1^{L}, 1^{n}, 1^{m}, 1^{T})$:

- 1. Call FE.Setup $(1^{\lambda}, 1^{L+\log T + n + \lambda + 1}, 1^T, 1^{\mathsf{poly}(\lambda,T)}) \to (\mathsf{mpk}, \mathsf{msk})$. Note that we have used T as an upper bound for the output length m for simplicity.
- 2. Sample a uniformly random seed $s \leftarrow \{0,1\}^{\lambda}$ and compute c = G(s).
- 3. Let U denote a universal circuit implementing the map $(M, T, x) \mapsto M(x, 1^T)$. In particular, we let U denote a canonical circuit associated with a universal Turing machine with a PV proof of correctness (e.g. via the Cook-Levin reduction in [Pic15]).
- 4. Let C denote the circuit that on input (M, T, x, s', b) outputs $c \oplus G(s')$ if b = 1 and U(M, T, x) if b = 0.
- 5. Compute $\mathsf{sk}_C = \mathsf{FE}.\mathsf{KeyGen}(\mathsf{msk}, C)$.
- 6. Output $(crs_{Enc} = mpk, crs_{Eval} = sk_C)$.

$$\begin{split} &\mathsf{RE}.\mathsf{Enc}(\mathsf{crs}_{\mathsf{Enc}}=\mathsf{mpk},M,x,T): \text{calls and outputs }\mathsf{FE}.\mathsf{Enc}(\mathsf{mpk},(M,x,0^\lambda,0)).\\ &\mathsf{RE}.\mathsf{Eval}(\mathsf{crs}_{\mathsf{Eval}}=\mathsf{sk}_C,\hat{\Pi}_x) \text{ calls and outputs }\mathsf{FE}.\mathsf{Dec}(\mathsf{sk}_C,\hat{\Pi}_x). \end{split}$$

Theorem 7.13 ([LPST16b] Theorem 12). Assuming the single-key security of FE and the security of G, the scheme in Algorithm γ is a simulation secure output-compressing randomized encoding in the crs model such that RE.Enc(crs_{Enc} = mpk, M, x, T) : runs in time poly($|M|, |x|, \lambda, \log m, \log T$).

What remains is to observe that:

Claim 7.14. *If* FE *has a PV proof of correctness, then* RE *has a PV proof of the following correctness property:*

for all $\lambda, L, n, m, T, r_1, r_2$, if $(crs_{Enc}, crs_{Eval}) = \mathsf{RE.Gen}(1^{\lambda}, 1^L, 1^n, 1^m, 1^T; r_1)$, for all M such that $|M| \leq L$, if $\hat{\Pi}_x = \mathsf{RE.Enc}(crs_{Enc}, M, x, T; r_2)$ and $y = \mathsf{RE.Eval}(crs_{Eval}, \Pi_x)$, we have that $y = M(x, 1^T)$.

This claim holds by the following (sketched) proof of correctness:

- By the correctness of FE, we have that $y = C(M, T, x, 0^{\lambda}, 0)$.
- By the definition of C, we have y = U(M, T, x).
- By the correctness of the circuit U, we have that $y = M(x, 1^T)$.
Algorithm 8 iO based on Output-Compressing RE [LPST16b]

 $iO.Obf(1^{\lambda}, C)$:

- 1. Let n denote the input length to C.
- 2. Set the security parameter $\kappa = \lambda \cdot \mathsf{poly}(n)$ so that the RE and PRG are both $2^{-n} \cdot \mathsf{negl}(\lambda)$ -secure.
- 3. Set $L = |C| + (n+1)(\kappa + 1)$.
- 4. Set $T = \mathsf{poly}(|C|, n, \lambda)$ such that for all Turing machines M of size at most L, all $\mathsf{crs}_{\mathsf{Enc}}$ using security parameter κ , all output lengths $m \leq 2^{\lambda}$, and all running times $T' \leq 2^{\lambda}$, we have that the running time of $\mathsf{RE.Enc}(\mathsf{crs}_{\mathsf{Enc}}, M, T', \bot)$ (empty input) is at most T/2.
- 5. Sample *n* common reference string pairs $(\mathsf{crs}_{\mathsf{Enc}}^{(i)}, \mathsf{crs}_{\mathsf{Eval}}^{(i)}) \leftarrow \mathsf{RE}.\mathsf{Setup}(1^{\lambda}, 1^{L}, 1^{0}, 1^{T}, 1^{T})$
- 6. Sample a PRG seed r_{ϵ} . For all r, define $G(r) = (r_0, r_1, s_0, s_1)$.
- 7. Define the collection of machines $M_{C,z,\operatorname{crs}_{i+1},\ldots,\operatorname{crs}_n,r}$ for $z \in \{0,1\}^{\leq n}$, i = |z| and $r \in \{0,1\}^{\kappa}$ as follows:
 - $M_{C,z,r}$ only runs on an empty input.
 - On empty input, first compute $(r_0, r_1, s_0, s_1) = G(r)$.
 - If i = n, output C(z). Otherwise:
 - Compute and output $\mathsf{RE}.\mathsf{Enc}(\mathsf{crs}_{\mathsf{Enc}}^{(i+1)}, M_{C,z||0,\mathsf{crs}_{i+2},\ldots,\mathsf{crs}_n,r_0}; s_0)$ as well as $\mathsf{RE}.\mathsf{Enc}(M_{C,z||1,\mathsf{crs}_{i+2},\ldots,\mathsf{crs}_n,r_1}; s_1).$

8. Output
$$M_{C,\epsilon,\operatorname{crs}_{\operatorname{Enc}}^{(1)},\ldots,\operatorname{crs}_{\operatorname{Enc}}^{(n)},r_{\epsilon}},\operatorname{crs}_{\operatorname{Eval}}^{(1)},\ldots,\operatorname{crs}_{\operatorname{Eval}}^{(n)}$$
.

 $\mathsf{iO}.\mathsf{Eval}(\tilde{C}, x)$:

- 1. Let $\tilde{C} = M_{C,\epsilon}, \operatorname{crs}_{\mathsf{Eval}}^{(1)}, \dots, \operatorname{crs}_{\mathsf{Eval}}^{(n)}$
- 2. For i from 1 to n:
 - Define $M_{C,x_1,...,x_i}$ to be the x_i th half of the output of RE.Eval $(crs_{Eval}^{(i-1)}, M_{C,x_1,...,x_{i-1}})$.
- 3. Compute and output $M_{C,x}(1^T)$.

7.3.6 iO from Output-Compressing RE

Finally, following [LPST16b], we construct iO from the output-compressing RE in the previous section, assuming that the RE is sub-exponentially secure and additionally making use of a sub-exponentially secure length-quadrupling PRG.

Theorem 7.15 ([LPST16b] Theorem 12). If RE is sub-exponentially simulation secure (in the crs model) and G is a sub-exponentially secure PRG, then the iO scheme in Algorithm 8 is secure.

Finally, it remains to demonstrate that:

Claim 7.16. If RE has a PV proof of correctness (as in Claim 7.14), then so does iO.

To see this, we prove the correctness of iO by proving the following main claim:

Claim 7.17. Let $\tilde{C} = iO.Obf(1^{\lambda}, C)$. For all z, let $G(r_z) = (r_{z||0}, r_{z||1}, s_{z||0}, s_{z||1})$. Then, for every i, the program $M_{C,x_1,...,x_i}$ defined in the evaluation procedure $iO.Eval(\tilde{C}, x)$ is equal to the program $M_{C,x_1,...,x_i}$, $crs_{i+1},...,crs_{n,r_{x_1},...,x_i}$ defined in $iO.Obf(1^{\lambda}, C)$.

This claim is proved by induction on i. The claim holds for i = 0 by the definition of iO.Obf and iO.Eval. For the inductive step, we observe that:

- $M_{C,x_1,...,x_{i+1}} = \left(\mathsf{RE}_{\mathsf{Eval}}(\mathsf{crs}_{\mathsf{Eval}}^{(i)}, M_{C,x_1,...,x_i})\right)[x_i].$
- By the inductive hypothesis on M_{C,x_1,\ldots,x_i} , this is equal to

$$\Big(\mathsf{RE}_{\mathsf{Eval}}(\mathsf{crs}_{\mathsf{Eval}}^{(i)}, M_{C, x_1, \dots, x_i, \mathsf{crs}_{\mathsf{Enc}}^{(i+1)}, \dots, \mathsf{crs}_{\mathsf{Enc}}^{(n)}, r_{x_1, \dots, x_i}})\Big)[x_i].$$

• By the correctness of RE, the definition of $r_{x_1,...,x_{i+1}}$, and the definition of $M_{C,x_1,...,x_i,crs_{Enc}^{(i+1)},...,crs_{Enc}^{(n)},r_{x_1,...,x_i}}$ this is equal to $M_{C,x_1,...,x_{i+1},crs}$.

By induction, we conclude that $M_{C,x} = M_{C,x,r_x}$. Finally, by the definition of M_{C,x,r_x} , we know that the output of M_{C,x,r_x} is equal to C(x), so we conclude that $\mathsf{iO}.\mathsf{Eval}(\tilde{C},x) = C(x)$ as claimed.

Provided that the correctness of RE is proved in PV, this proof can also be formalized in PV because the only invocation of induction is with respect to the length $i \leq n$ corresponding to the bit-string z.

7.4 Non-adaptive Transparent SNARGs for NP from iO and LWE

In this section, we recall the SNARG for NP construction of Sahai and Waters [SW14] in Algorithm 9, and show that it does indeed have a proof of correctness if the underlying iO has a proof of correctness.

Claim 7.18. Algorithm 9 is a SNARG scheme with a PV proof of completeness.

Proof sketch. We outline the proof of completeness below.

1. For any security parameter λ , $k \leftarrow \mathsf{PRF.Gen}(1^{\lambda})$ and randomness r_P, r_V , let $\tilde{P} \leftarrow \mathsf{iO}(P_k; r_P)$ and $\tilde{V} \leftarrow \mathsf{iO}(V_k; r_V)$

Algorithm 9 SNARG construction of [SW14, MPV24]

SNARG.Gen $(1^{\lambda}, R)$:

- Compute $k \leftarrow \mathsf{PRF}.\mathsf{Gen}(1^{\lambda})$, where PRF is a puncturable PRF family.
- Compute obfuscations of the following programs:
 - P_k : On input instance x, witness w, if R(x, w) = 1, output $\pi = \mathsf{PRF}.\mathsf{Eval}(k, x)$. Else, output \perp .
 - V_k : It takes as input instance x and proof π : if $f(\pi) = f(\mathsf{PRF}.\mathsf{Eval}(k, x))$, then output 1. Else, output 0.
- Output $\operatorname{crs} = (\widetilde{P} = \operatorname{iO}(P_k), \widetilde{V} = \operatorname{iO}(V_k)).$

SNARG.Prove(crs = $(\tilde{P}, \tilde{V}), x, w$): Compute and output $\pi \leftarrow \tilde{P}(x, w)$. SNARG.Ver(crs = $(\tilde{P}, \tilde{V}), x, \pi$): Accept if $1 \leftarrow \tilde{V}(x, \pi)$, and reject otherwise.

- 2. By correctness of iO, $P_k(x, w) \to \pi \to \tilde{P}(x, w) = \pi$.
- 3. By definition of $P_k(x, w)$, $R(x, w) = 1 \rightarrow P_k(x, w) = \mathsf{PRF}.\mathsf{Eval}(k, x)$.
- 4. Combining (2) and (3), we have $R(x, w) = 1 \rightarrow \tilde{P}(x, w) = \mathsf{PRF}.\mathsf{Eval}(k, x)$.
- 5. By correctness of iO, $V_k(x,\pi) = b \rightarrow \tilde{V}(x,\pi) = b$.
- 6. By definition, for all x, $V_k(x, \mathsf{PRF}.\mathsf{Eval}(k, x)) = 1$.
- 7. By (5), we have that for all x, $\tilde{V}(x, \mathsf{PRF}.\mathsf{Eval}(k, x)) = 1$.
- 8. Combining (4) and (7), we have $R(x, w) = 1 \rightarrow \tilde{V}(x, \tilde{P}(x, w)) = 1$, as desired.

7.5 Adaptive SNARGs for NP from iO and LWE

In this section, we show that the adaptively sound dv-SNARG from [MPV24] does have a proof of completeness. Recall this construction is identical to the construction of Algorithm 9, except that only the prover algorithm P_k is published.

Claim 7.19. Algorithm 10 has a PV proof of completeness.

Proof. This proof follows very closely to the proof of the previous claim.

- 1. For any security parameter $\lambda, k \leftarrow \mathsf{PRF}.\mathsf{Gen}(1^{\lambda})$ and randomness r, let $\tilde{P} \leftarrow \mathsf{iO}(P_k; r)$.
- 2. By correctness of iO, $P_k(x, w) \to \pi \to \tilde{P}(x, w) = \pi$.
- 3. By definition of $P_k(x, w)$, $R(x, w) = 1 \rightarrow P_k(x, w) = \mathsf{PRF}.\mathsf{Eval}(k, x)$.
- 4. Combining (2) and (3), we have $R(x, w) = 1 \rightarrow \tilde{P}(x, w) = \mathsf{PRF}.\mathsf{Eval}(k, x)$.
- 5. By definition, for all x, dv-SNARG.Ver(k, x, PRF.Eval(k, x)) = 1.
- 6. Combining (4) and (5), we have $R(x, w) = 1 \rightarrow \mathsf{dv}\text{-}\mathsf{SNARG}$. $\mathsf{Ver}(x, P(x, w)) = 1$, as desired.

Algorithm 10 dv-SNARG construction of [SW14]

dv-SNARG.Gen $(1^{\lambda}, R)$:

- Compute $k \leftarrow \mathsf{PRF}.\mathsf{Gen}(1^{\lambda})$, where PRF is a puncturable PRF family.
- Compute an obfuscation of the following program:
 - P_k : On input instance x, witness w, if R(x, w) = 1, output $\pi = \mathsf{PRF}.\mathsf{Eval}(k, x)$. Else, output \perp .
- Output $\operatorname{crs} = (\widetilde{P} = \operatorname{iO}(P_k))$, and verifier key $\mathsf{vk} = k$.

dv-SNARG.Prove(crs = \tilde{P}, x, w): Compute and output $\pi \leftarrow \tilde{P}(x, w)$. dv-SNARG.Ver(vk, x, π): Accept if $\pi = \mathsf{PRF.Eval}(k, x)$. Else, reject.

8 Application III: Adaptive Transparent SNARGs for UP from Evasive LWE

In this section, we prove the following theorem via Theorem 5.1.

Theorem 8.1. Assuming subexponential LWE and evasive LWE, there exists an adaptively sound SNARG for $\mathcal{L} \in UP$ with a relation circuit of size T with the following parameters:

- The crs is of length $poly(n, T, \lambda)$ with transparent set-up.
- The proof is of size $poly(\lambda, \log T)$.
- The prover runtime is $poly(T + L, \lambda)$.
- The verifier runtime is $poly(n, \lambda, \log T)$.

To obtain Theorem 8.1, we first recap the adaptively sound designated-verifier SNARG for UP from Mathialagan, Peters and Vaikuntanathan [MPV24].

Theorem 8.2 ([MPV24, Corollary 6.2]). Assuming subexponential LWE and evasive LWE, there exists an adaptively sound dv-SNARG for UP with a relation circuit of size T with the following parameters:

- The crs has length $poly(n, T, \lambda)$.
- The proof is of size $\lambda + \omega(\log(T, \lambda))$.
- The prover runtime is $poly(T, \lambda)$.
- The verifier runtime is $poly(n, \lambda, \log T)$.

As is, their construction does not have an \mathcal{EF} proof of completeness. We minor modifications to achieve *perfect completeness* and argue that it has an \mathcal{EF} proof of completness, and argue that the modified scheme is still sound. Additionally, we argue that the common reference string of their scheme is indistinguishable from random, even against circuits of size $2^{|x|}$.

Theorem 8.3. Assuming LWE and evasive LWE, there exists an adaptively sound dv-SNARG for UP with a polynomial-sized \mathcal{EF} proof of completeness.

- The crs has length $poly(n, T, \lambda)$. Moreover, it is 2^{-n} indistinguishable from a transparent distribution.
- The proof is of size $\lambda + \omega(\log(T, \lambda))$.
- The prover runtime is $poly(T, \lambda)$.
- The verifier runtime is $poly(n, \lambda, \log T)$.
- The \mathcal{EF} proof of completeness has size $poly(n, \lambda, T)$.

Therefore, combining the above result with Theorem 5.1, we obtain Theorem 8.1.

Organization. We first recap some additional tools from [MPV24] involving read-c MBPs (rather than read-once, as was sufficient for the witness encryption construction) in Appendix A.2. We defer the construction and proof of Theorem 8.3 to Appendix A.3. We then prove that for instances of size h and relation circuit size T, the new scheme:

- is complete with a polynomial-sized \mathcal{EF} proof of size $poly(h, T, \lambda)$ (Claim A.6).
- is adaptively sound with proof size $\lambda + \omega(\log(h+T+\lambda))$ and crs size $poly(h, T, \lambda)$ (Claim A.7).
- has a common reference string which is 2^{-h} indistinguishable from a string sampled from a transparent distribution (Claim A.10).

The above claims prove Theorem 8.3. Combining this with Theorem 5.1, we obtain Theorem 8.1.

9 Application IV: Universal Micali SNARGs

In this section, as an application of our main theorem, we show a universal SNARG construction for NP with non-adaptive soundness, assuming the (non-explicit) existence of a hash function that securely instantiates Micali's SNARG. To state our result, we need to define probabilistically checkable proofs with an additional property, which requires that the completeness PV proofs.

Definition 9.1 (PCP with PV proofs of completeness). Let $\mathcal{L} \subseteq \{0,1\}^*$ be an NP language with the relation $\mathcal{R}_{\mathcal{L}}$. We say a probabilistically checkable proof $(\mathcal{P}, \mathcal{Q}, \mathcal{V})$ for \mathcal{L} has a PV proof of completeness, if there exists a polynomial p = p(n) such that, for every $n \in \mathbb{N}$, there exists an uniformly computable polynomial-sized \mathcal{EF} proof of length p(n) for the following statement:

 $(\mathcal{R}_{\mathcal{L}}(x,w) = 1 \land \pi = \mathcal{P}(x,w) \land Q = \mathcal{Q}(1^n,r)) \to \mathcal{V}(\mathsf{Proj}(\pi,Q),r) = 1),$

where $\operatorname{Proj}(\pi, Q)$ is a circuit that takes a string π and a subset of indices Q as input, and it outputs the projection of π onto the coordinates specified by the subset Q, i.e. $\pi|_Q$.

Intuitively, the formula in Definition 9.1 states that, for any x, w such that $\mathcal{R}_{\mathcal{L}}(x, w) = 1$, let $\pi = \mathcal{P}(x, w)$, then for any r and $Q = \mathcal{Q}(1^n, r)$, $\mathcal{V}(\pi|_Q, r) = 1$.

We are ready to state our main theorem in this section.

Theorem 9.2. For any NP language $\mathcal{L} \subseteq \{0,1\}^*$ with a relation $\mathcal{R}_{\mathcal{L}}$ that can be computed by a circuit of size T = T(n), there exists a SNARG construction for \mathcal{L} , with the following parameters.

- The crs is of length $poly(\lambda)$.
- The proof length is $poly(\log n, \lambda)$.
- The prove runtime is $poly(T, \lambda)$.
- The verifier runtime is $\tilde{O}(n) + \mathsf{poly}(\log n, \lambda)$.

It is non-adaptively sound assuming LWE, FHE, and the existence of a (non-explicit) instantiation of the hash function such that Micali's [Mic94] SNARG for \mathcal{L} is non-adaptively sound, where the underlying PCP of Micali's SNARG is instantiated with a PCP with PV proof of completeness.

We stress that PCP with PV proof of completeness is a property satisfied by most of the PCP constructions. For example, [Pic15] proved that Dinur's PCP [Din07] has polynomial-size \mathcal{EF} proofs of completeness (and soundness).

The proof of the theorem relies on the following lemma.

Lemma 9.3 (Informal). There exists a PV proof for the opening completeness property of Merkle hash.

We remark that Lemma 9.3 works for any hash function underlying the Merkle tree. We will formalize the above theorem in PV and provide a formal proof in Section 9.3.

To prove Theorem 9.2, we first prove the following lemma about Micali's SNARG. Then Theorem 9.2 follows from the combination of the following lemma and Theorem 4.1.

Lemma 9.4 (*PV* Proof of Completeness for Micali's SNARG). For any NP language $\mathcal{L} \subseteq \{0, 1\}^*$, and Micali's SNARGs (Gen, \mathcal{P}, \mathcal{V}) instantiated by any hash function, and any *PCP* with a *PV* proof of completeness, there exists a *PV* for the following statement:

$$(\operatorname{crs} = \operatorname{Gen}(1^{\lambda}; r) \land \pi = \mathcal{P}(\operatorname{crs}, x, w)) \to \mathcal{V}(\operatorname{crs}, x, \pi) = 1.$$

Proof Sketch. We only provide a proof sketch here. Recall that, Micali's SNARG construction is as follows. The CRS generation algorithm outputs a hash key for Merkle hash as the CRS. The honest prover algorithm generates a probabilistically checkable proof (PCP) π , and Merkle hashes the PCP and obtains a Merkle root rt. It applies the hash function again to rt, and parses the hash output as PCP queries Q. Then it answers the PCP queries $\pi|_Q$ and the local opening $\{\rho_q\}_{q\in Q}$ for the locations in Q. The SNARG proof consists of $(\mathsf{rt}, \pi|_Q, \{\rho_q\}_{q\in Q})$. Given the SNARG proof, the verifier first verifies the PCP answers, and then it verifies the opening of the Merkle hash.

The PV proof of completeness consists of the PV proof of the PCP, and the PV proof of local opening completeness of the Merkle hash. From Lemma 9.3, there exists a PV proof of the local opening completeness of Merkle hash.

Concatenating the two PV proofs, we obtain an PV proof of completeness for Micali's SNARG. This finishes the proof of the lemma.

9.1 Probabilistically Checkable Proofs

A probabilistically checkable proof for an NP language \mathcal{L} is a tuple of algorithms $(\mathcal{P}, \mathcal{Q}, \mathcal{V})$, with the following properties.

Syntax. $(\mathcal{P}, \mathcal{Q}, \mathcal{V})$ have the following syntax.

- $\mathcal{P}(x, w)$: This algorithm takes as input an instance x and an witness w, and it outputs a proof π of length L(|x|), where L = L(n) is a polynomial.
- $\mathcal{Q}(1^n, r)$: This algorithm takes as input the input length n, and a uniformly random string r, and output a subset of indices $Q \subseteq [L(n)]$.
- $\mathcal{V}(\pi', r)$: The verification algorithm takes as input a string π' of length |Q|, and the random coin r to verify the proof. It either accepts the proof (outputs 1), or rejects the proof (outputs 0).

Moreover, we require them to satisfy the following properties.

• Completeness. For any $(x, w) \in \mathcal{R}_{\mathcal{L}}$, where $\mathcal{R}_{\mathcal{L}}$ is the NP-relation of \mathcal{L} , we have

$$\Pr_{r}[\mathcal{V}(\pi|_{Q}, r) = 1 : \pi \leftarrow \mathcal{P}(x, w), Q \leftarrow \mathcal{Q}(1^{|x|}, r)] = 1.$$

• ϵ -Soundness. For any $x \notin \mathcal{L}$,

$$\Pr[\exists \pi^* \in \{0, 1\}^{L(|x|)} \text{ s.t. } \mathcal{V}(\pi^*|_Q, r) = 1 : Q \leftarrow \mathcal{Q}(1^{|x|}, r)] \le \epsilon.$$

9.2 Merkle Tree Hash

In this section we recall the definition of a hash family with local opening [Mer88]. This section is [BBK⁺23].

- Syntax. A hash family (HT) with succinct local opening consists of the following algorithms:
- $Gen(1^{\lambda}) \rightarrow hk$. This is a PPT algorithm that takes as input the security parameter λ in unary and outputs a hash key hk.
- $\mathsf{Hash}(\mathsf{hk}, x) \to \mathsf{rt.}$ This is a deterministic poly-time algorithm that takes as input a hash key hk and an input $x \in \{0, 1\}^N$ for $N \leq 2^{\lambda}$, and outputs a hash value $\mathsf{rt.}$
- Open(hk, x, j) $\rightarrow \rho$. This is a deterministic poly-time algorithm that takes as input a hash key hk, an input $x \in \{0, 1\}^N$ for $N \leq 2^{\lambda}$, and an index $j \in [N]$, and outputs an opening ρ .
- Verify(hk, rt, j, b, ρ) $\rightarrow 0/1$. This is a deterministic poly-time algorithm that takes as input a hash key hk, a hash value rt, an index $j \in [N]$, a bit $b \in \{0, 1\}$ and an opening ρ . It outputs 1 (accept) or 0 (reject).

Definition 9.5. (Properties of HT) A HT family (Gen, Hash, Open, Verify) is required to satisfy the following properties.

Opening completeness. For any $\lambda \in \mathbb{N}$, any $N \leq 2^{\lambda}$, any $x \in \{0,1\}^N$, and any index $j \in [N]$,

$$\Pr\left[\begin{array}{cc}\mathsf{hk} \leftarrow \mathsf{Gen}(1^{\lambda}),\\\mathsf{Verify}(\mathsf{hk},\mathsf{rt},j,x_j,\rho) = 1 &: &\mathsf{rt} = \mathsf{Hash}(\mathsf{hk},x),\\ &\rho = \mathsf{Open}(\mathsf{hk},x,j)\end{array}\right] = 1.$$

Succinctness. In the completeness experiment above, we have that $|h\mathbf{k}| + |\mathbf{rt}| + |\rho| = \text{poly}(\lambda)$.

Collision resistance w.r.t. opening. For any poly-size adversary \mathcal{A} there exists a negligible function negl(·) such that for every $\lambda \in \mathbb{N}$,

$$\Pr\left[\begin{array}{cc} \mathsf{Verify}(\mathsf{hk},\mathsf{rt},j,0,\rho_0) = 1 \\ \wedge \mathsf{Verify}(\mathsf{hk},\mathsf{rt},j,1,\rho_1) = 1 \end{array} : \begin{array}{c} \mathsf{hk} \leftarrow \mathsf{Gen}(1^n), \\ (\mathsf{rt},j,\rho_0,\rho_1) \leftarrow \mathcal{A}(\mathsf{hk}) \end{array}\right] = \mathsf{negl}(n).$$

Remark 9.6. We say that a hash family with local opening is T-secure, for T = T(n), if the collision resistance w.r.t. opening property holds against any poly(T)-size adversary (as opposed to poly(n)-size) and the probability that the adversary finds a collision is negl(T) (as opposed to negl(n)). We refer to this property as T-collision-resistance w.r.t. opening.

Remark 9.7. One can naturally extend the definition of a hash family with local opening to allow the Open algorithm to take as input (hk, x, J) where $J \subseteq [N]$ consists of a set of indices, as opposed to a single index. Open(hk, x, J) will simply run Open(hk, x, j) for every $j \in J$. Verify can be extended in a similar way to take as input (hk, rt, J, b_J, ρ_J) , and accept if and only if Verify $(hk, rt, j, b_j, \rho_j) = 1$ for every $j \in J$.

Theorem 9.8 ([Mer88]). Assuming the existence of a collision resistant hash family there exists a hash family with local opening (according to Definition 9.5).

9.3 PV Proof of Completeness for Merkle Hash

In this section, we will prove that the opening completeness of any Merkle hash can be proven in PV, for a Merkle hash tree instantiated from any underlying hash function.

Indeed, one can formalize the proof of completeness in Cook's theory PV and then use Cook's propositional translation to obtain a polynomial-size proof in \mathcal{EF} .

We first formalize the opening completeness of Merkle hash in Extended Frege system. To do this, we need to express the statement

"for any hk and x_1, \ldots, x_N , rt = Hash(hk, x_1, \ldots, x_N), ρ = Open(hk, $(x_1, \ldots, x_N), j$), then Verify(hk, rt, j, x_j, ρ) = 1."

in *PV*. To achieve this, we first need to formalize the notation x_j as a function of x_1, \ldots, x_N and j. Namely, we define the following functions $\operatorname{Proj}(\mathbf{x}, j) = x_j$, where $\mathbf{x} = (x_1, \ldots, x_N)$.

Without loss of generality, in the rest of this section, we assume that N is a power of 2. Namely, $N = 2^d$ for an integer d. Then we construct Proj circuit, as well as the circuits Hash, Open, Verify recursively.

Definition 9.9 (Merkle Hash). Let $H = {(Gen_{\lambda}, H_{\lambda})}_{\lambda \in \mathbb{N}}$ be any hash function family with the following syntax.

- Gen_{λ}(r) : This is a key generation circuit that takes $R = R(\lambda)$ bits of random coins, and outputs a hash key hk.
- H_λ: The hash function takes as input a hash key hk and a string of 2λ bits, and it outputs a hash value of λ bits.

We recursively construct the following circuits of the Merkle tree hash family $HT = (Gen, Hash, Open, Verify)^{29}$:

• $Proj(\mathbf{x}, j)$: For any $b \in \{0, 1\}$ and $x_0, x_1 \in \{0, 1\}$, we construct

 $\mathsf{Proj}(x_0||x_1,b) = \mathbf{x}_b.$

For any $\mathbf{x}_0, \mathbf{x}_1 \in \{0,1\}^{2^{d-1}}$ and $\overline{j} \in \{0,1\}^{d-1}$ where d > 1, and $b \in \{0,1\}$, we construct

$$\operatorname{Proj}(\mathbf{x}_0||\mathbf{x}_1,b||\overline{j}) = \operatorname{Proj}(\mathbf{x}_b,\overline{j}).$$

- Gen(1^λ; r): The key generation algorithm takes R(λ) random bits as input and runs hk ← Gen_λ(r). It outputs hk.
- $\mathsf{Hash}(\mathsf{hk}, \mathbf{x})$: For any $x_0, x_1 \in \{0, 1\}$, we construct

$$Hash(hk, (x_0, x_1)) = H(hk, (x_0, x_1)).$$

For any $\mathbf{x}_0, \mathbf{x}_1 \in \{0, 1\}^{2^{d-1}}$ where d > 1, we construct

 $\mathsf{Hash}(\mathsf{hk},\mathbf{x}_0||\mathbf{x}_1) = \mathsf{H}(\mathsf{hk},\mathsf{Hash}(\mathsf{hk},\mathbf{x}_0)||\mathsf{Hash}(\mathsf{hk},\mathbf{x}_1)).$

• Open(hk, \mathbf{x}, j): For $j \in \{0, 1\}, x_0, x_1 \in \{0, 1\}$, we construct

Open(hk,
$$\mathbf{x}_0 || \mathbf{x}_1, j) = (x_0, x_1).$$

For any $\mathbf{x}_0, \mathbf{x}_1 \in \{0,1\}^{2^{d-1}}$ where d > 1, $\overline{j} \in \{0,1\}^{d-1}$, and $b \in \{0,1\}$, we construct

 $\mathsf{Open}(\mathsf{hk}, \mathbf{x}_0 || \mathbf{x}_1, b || \overline{j}) = (\mathsf{Hash}(\mathsf{hk}, \mathbf{x}_0), \mathsf{Hash}(\mathsf{hk}, \mathbf{x}_1), \mathsf{Open}(\mathsf{hk}, \mathbf{x}_b, \overline{j})).$

• Verify(hk, rt, j, v, ρ): $j \in \{0, 1\}^d$, $v \in \{0, 1\}$ where $d \ge 1$, and ρ be a tuple of length 2d, where each entry is a binary string of length λ .

For d = 1, we construct Verify(hk, rt, j, b, ρ) as the following circuit. It parses $\rho = (x_0, x_1)$ and checks whether

$$\mathsf{rt} = \mathsf{H}(\mathsf{hk}, (x_0, x_1)) \land v = x_j.$$

For $d \ge 2$, we recursively construct Verify(hk, rt, j, b, ρ) as the following circuit. We first parse $\rho = (h_0, h_1, \rho')$ and construct

$$\mathsf{Verify}(\mathsf{hk},\mathsf{rt},b||j,v,\rho) = \mathsf{Verify}(\mathsf{hk},h_b,j,v,\rho') \land \mathsf{rt} = \mathsf{H}(\mathsf{hk},h_0||h_1).$$

Then we formalize the opening completeness of the Merkle hash as the following theorem statement.

²⁹Indeed, Gen = {Gen_{λ}}_{$\lambda \in \mathbb{N}$}, Hash = {Hash_{$d,\lambda}}_{<math>d,\lambda \in \mathbb{N}$}, Open = {Open_{$d,\lambda}}_{<math>d,\lambda \in \mathbb{N}$}, Verify = {Verify_{$d,\lambda}}_{<math>d,\lambda \in \mathbb{N}$} are families of circuits. For the simplicity of representation, we suppress the index when it is clear from the context.</sub></sub></sub>

Theorem 9.10 (*PV*-Proofs of Opening Completeness). There exists a polynomial $L = L(N, \lambda)$ such that, for any integer $\lambda \in \mathbb{N}$, and $N = 2^d$, there exists an Extended Frege proof of length $L(N, \lambda)$ for the following statement:

$$(\mathsf{rt} = \mathsf{Hash}(\mathsf{hk}, \mathbf{x}) \land \rho = \mathsf{Open}(\mathsf{hk}, \mathbf{x}, j)) \rightarrow \mathsf{Verify}(\mathsf{hk}, \mathsf{rt}, j, \mathsf{Proj}(\mathbf{x}, j), \rho) = 1,$$

where $\mathbf{x} = (x_1, \dots, x_N)$, and the circuits Gen, Hash, Open, Verify are constructed in Definition 9.9.

Proof. We prove the theorem by constructing the PV proofs for the statement. The construction is recursive on the variable d. Namely, for any fixed $\lambda \in \mathbb{N}$, we first prove the theorem directly in PV for d = 1, and denote the PV proof as π_1 . For $d \geq 2$, we recursively build an PV proof π_d from π_{d-1} .

Base Case: d = 1. If d = 1, then $N = 2^d = 2$. From the construction in Definition 9.9, we have that, for any $x_0, x_1 \in \{0, 1\}, j \in \{0, 1\}, v \in \{0, 1\}$, the circuits used in Merkle hash can be represented as follows.

- $\mathsf{Hash}(\mathsf{hk}, (x_0, x_1)) = \mathsf{H}(\mathsf{hk}, (x_0, x_1)),$
- Open(hk, $(x_0, x_1), j) = (x_0, x_1)$.
- $\mathsf{Verify}(\mathsf{hk},\mathsf{rt},j,v,\rho=(x_0',x_1')\in\{0,1\}^2)=\mathsf{H}(\mathsf{hk},(x_0',x_1'))\wedge v=x_j'.$
- $Proj((x_0, x_1), j) = x_j$.

Now the statement we want to prove becomes

$$(\mathsf{rt} = \mathsf{H}(\mathsf{hk}, (x_0, x_1)) \land \rho = (x_0, x_1)) \to \mathsf{Verify}(\mathsf{hk}, \mathsf{rt}, j, \mathsf{Proj}((x_0, x_1), j), \rho) = 1.$$
(22)

We prove it in PV by considering two cases j = 0 and j = 1.

• If j = 0, we only need to further prove in PV that

$$(\mathsf{rt} = \mathsf{H}(\mathsf{hk}, (x_0, x_1)) \land \rho = (x_0, x_1)) \rightarrow \mathsf{Verify}(\mathsf{hk}, \mathsf{rt}, 0, x_0, \rho) = 1.$$

This is because we only need a constant size PV proof to show that $Proj((x_0, x_1), 0) = x_0$, from the construction of Proj. The above formula can be proven by an PV-proof of size $poly(\lambda)$ using the circuit construction of Verify.

• If j = 1, we need to further prove in PV the following formula:

$$(\mathsf{rt} = \mathsf{H}(\mathsf{hk}, (x_0, x_1)) \land \rho = (x_1, x_0)) \rightarrow \mathsf{Verify}(\mathsf{hk}, \mathsf{rt}, 1, x_1, \rho) = 1.$$

This formula has a $poly(\lambda)$ -size PV proof due to the same reason as the case of j = 0.

Combining the above two cases, we obtain a PV proof for Equation (22).

Induction Step: $d \ge 2$. In this case, we represent $\mathbf{x} = \mathbf{x}_0 ||\mathbf{x}_1$, where $\mathbf{x}_0, \mathbf{x}_1$ are the vectors of 2^{d-1} variables. We also represent $j = b ||\overline{j}$, where b is a variable, and \overline{j} is a vector of (d-1) variables. Then the formula we want to prove is

$$(\mathsf{rt} = \mathsf{Hash}(\mathsf{hk}, \mathbf{x}_0 || \mathbf{x}_1) \land \rho = \mathsf{Open}(\mathsf{hk}, \mathbf{x}_0 || \mathbf{x}_1, b || \overline{j})) \to \mathsf{Verify}(\mathsf{hk}, \mathsf{rt}, b || \overline{j}, \mathsf{Proj}(\mathbf{x}_0 || \mathbf{x}_1, b || \overline{j}), \rho) = 1.$$

$$(23)$$

From the recursive construction of $\mathsf{Hash}(\mathsf{hk}, \mathbf{x}_0 || \mathbf{x}_1)$, we have a *PV* proof of

$$\mathsf{Hash}(\mathsf{hk}, \mathbf{x}_0 || \mathbf{x}_1) = \mathsf{H}(\mathsf{hk}, \mathsf{Hash}(\mathsf{hk}, \mathbf{x}_0), \mathsf{Hash}(\mathsf{hk}, \mathbf{x}_1)).$$

Similarly, we have PV proofs of the following formulas.

$$\begin{split} \mathsf{Open}(\mathsf{hk},\mathbf{x}_0||\mathbf{x}_1,b||\overline{j}) &= (\mathsf{Hash}(\mathsf{hk},\mathbf{x}_0),\mathsf{H}(\mathsf{hk},\mathbf{x}_1),\mathsf{Open}(\mathsf{hk},\mathbf{x}_b,\overline{j})), \quad b = 0,1,\\ \mathsf{Proj}(\mathbf{x}_0||\mathbf{x}_1,b||\overline{j}) &= \mathsf{Proj}(\mathbf{x}_b,\overline{j}), \quad b = 0,1, \end{split}$$

and also,

10

$$\mathsf{Verify}(\mathsf{hk},\mathsf{rt},b||\overline{j},v,\rho=(h_0,h_1,\rho')) = \mathsf{Verify}(\mathsf{hk},h_b,\overline{j},v,\rho') \land \mathsf{rt} = \mathsf{H}(\mathsf{hk},h_0||h_1), \quad b = 0,1$$
(24)

Leveraging the above PV proofs, to prove Equation (23), it suffices to prove the following formula, for any fixed value $b \in \{0, 1\}$ "hardwired" in the formula,

$$(\mathsf{rt} = \mathsf{H}(\mathsf{hk}, \mathsf{Hash}(\mathsf{hk}, \mathbf{x}_0), \mathsf{Hash}(\mathsf{hk}, \mathbf{x}_1)) \land \rho = (\mathsf{Hash}(\mathsf{hk}, \mathbf{x}_0), \mathsf{Hash}(\mathsf{hk}, \mathbf{x}_1), \mathsf{Open}(\mathsf{hk}, \mathbf{x}_b, \overline{j}))) \rightarrow \mathsf{Verify}(\mathsf{hk}, \mathsf{rt}, b||\overline{j}, \mathsf{Proj}(\mathbf{x}_b, \overline{j}), \rho) = 1.$$
(25)

Levering Equation (24), we can use a PV proof to prove that the above formula is equivalent to

$$(\mathsf{rt} = \mathsf{H}(\mathsf{hk}, \mathsf{Hash}(\mathsf{hk}, \mathbf{x}_0), \mathsf{Hash}(\mathsf{hk}, \mathbf{x}_1)) \land \rho = (\mathsf{Hash}(\mathsf{hk}, \mathbf{x}_0), \mathsf{Hash}(\mathsf{hk}, \mathbf{x}_1), \mathsf{Open}(\mathsf{hk}, \mathbf{x}_b, \overline{j}))) \\ \rightarrow \mathsf{Verify}(\mathsf{hk}, \mathsf{Hash}(\mathsf{hk}, \mathbf{x}_b), \overline{j}, \mathsf{Proj}(\mathbf{x}_b, \overline{j}), \mathsf{Open}(\mathsf{hk}, \mathbf{x}_b, \overline{j})) = 1.$$
(26)

Let π_d be the *PV* proof of the Equation (23) for *d*-level Merkle hash tree. Then Equation (26) can be prove by two copies of π_{d-1} , one for b = 0, and the other for b = 1. Hence, we obtain an *PV* proof π_d for Equation (23).

Therefore, by the polynomial-time induction rule in PV to complete the proof.

Acknowledgements

We would like to thank Vinod Vaikuntanathan and Jiatu Li for helpful discussions.

Yael Tauman Kalai is supported by DARPA under Agreement No. HR00112020023. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Government or DARPA. Surya Mathialagan is supported in part by NSF CNS-2154149, a Simons Investigator Award, and Jane Street.

References

- [ABSV15] Prabhanjan Ananth, Zvika Brakerski, Gil Segev, and Vinod Vaikuntanathan. From selective to adaptive security in functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, CRYPTO 2015, Part II, volume 9216 of LNCS, pages 657–677. Springer, Berlin, Heidelberg, August 2015. 66, 67, 68
- [AJ15] Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, CRYPTO 2015, Part I, volume 9215 of LNCS, pages 308–326. Springer, Berlin, Heidelberg, August 2015. 1, 19, 60, 64
- [Ajt96] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In 28th ACM STOC, pages 99–108. ACM Press, May 1996. 50
- [AP11] Joël Alwen and Chris Peikert. Generating shorter bases for hard random lattices. Theory of Computing Systems, 48:535–553, 2011. 50
- [ARYY23] Shweta Agrawal, Mélissa Rossi, Anshu Yadav, and Shota Yamada. Constant input attribute based (and predicate) encryption from evasive and tensor LWE. In Helena Handschuh and Anna Lysyanskaya, editors, CRYPTO 2023, Part IV, volume 14084 of LNCS, pages 532–564. Springer, Cham, August 2023. 4
- [Bar86] David A. Mix Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC¹. In 18th ACM STOC, pages 1–5. ACM Press, May 1986. 90
- [BBK⁺23] Zvika Brakerski, Maya Farber Brodsky, Yael Tauman Kalai, Alex Lombardi, and Omer Paneth. SNARGs for monotone policy batch NP. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part II*, volume 14082 of *LNCS*, pages 252–283. Springer, Cham, August 2023. 2, 9, 24, 76
- [BCC⁺17] Nir Bitansky, Ran Canetti, Alessandro Chiesa, Shafi Goldwasser, Huijia Lin, Aviad Rubinstein, and Eran Tromer. The hunting of the snark. *Journal of Cryptology*, 30(4):989–1066, 2017. 1
- [BCCT13] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for SNARKS and proof-carrying data. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, 45th ACM STOC, pages 111–120. ACM Press, June 2013. 23
- [BCI⁺13] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct non-interactive arguments via linear interactive proofs. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 315–333. Springer, Berlin, Heidelberg, March 2013. 1
- [BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In Martin Hirt and Adam D. Smith, editors, TCC 2016-B, Part II, volume 9986 of LNCS, pages 31–60. Springer, Berlin, Heidelberg, October / November 2016. 1, 6

- [BGG⁺14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully keyhomomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 533–556. Springer, Berlin, Heidelberg, May 2014. 19
- [BGI⁺01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 1–18. Springer, Berlin, Heidelberg, August 2001. 61
- [BHK17] Zvika Brakerski, Justin Holmgren, and Yael Tauman Kalai. Non-interactive delegation and batch NP verification from standard computational assumptions. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, 49th ACM STOC, pages 474–482. ACM Press, June 2017. 2, 29, 30
- [BKK⁺18] Saikrishna Badrinarayanan, Yael Tauman Kalai, Dakshita Khurana, Amit Sahai, and Daniel Wichs. Succinct delegation for low-space non-deterministic computation. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, 50th ACM STOC, pages 709–721. ACM Press, June 2018. 2
- [BLMR13] Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key homomorphic PRFs and their applications. In Ran Canetti and Juan A. Garay, editors, CRYPTO 2013, Part I, volume 8042 of LNCS, pages 410–428. Springer, Berlin, Heidelberg, August 2013. 16, 21
- [BR94] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 232–249. Springer, Berlin, Heidelberg, August 1994. 1
- [Bus86] Samuel Buss. Bounded Arithmetic. Bibliopolis, Naples, Italy, 1986. 3, 16, 27, 96
- [BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, 52nd FOCS, pages 97–106. IEEE Computer Society Press, October 2011. 2, 19, 23, 107
- [BV15] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In Venkatesan Guruswami, editor, 56th FOCS, pages 171–190. IEEE Computer Society Press, October 2015. 1, 19, 60, 64
- [CCH⁺19] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. Fiat-Shamir: from practice to theory. In Moses Charikar and Edith Cohen, editors, 51st ACM STOC, pages 1082–1090. ACM Press, June 2019.
- [CGJ⁺23] Arka Rai Choudhuri, Sanjam Garg, Abhishek Jain, Zhengzhong Jin, and Jiaheng Zhang. Correlation intractability and SNARGs from sub-exponential DDH. In Helena Handschuh and Anna Lysyanskaya, editors, CRYPTO 2023, Part IV, volume 14084 of LNCS, pages 635–668. Springer, Cham, August 2023. 2, 8, 25

- [CHVW19] Yilei Chen, Minki Hhan, Vinod Vaikuntanathan, and Hoeteck Wee. Matrix PRFs: Constructions, attacks, and applications to obfuscation. In Dennis Hofheinz and Alon Rosen, editors, TCC 2019, Part I, volume 11891 of LNCS, pages 55–80. Springer, Cham, December 2019. 16
- [CJJ21] Arka Rai Choudhuri, Abhishek Jain, and Zhengzhong Jin. Non-interactive batch arguments for NP from standard assumptions. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part IV*, volume 12828 of *LNCS*, pages 394–423, Virtual Event, August 2021. Springer, Cham. 2
- [CJJ22] Arka Rai Choudhuri, Abhishek Jain, and Zhengzhong Jin. SNARGs for *P* from LWE. In 62nd FOCS, pages 68–79. IEEE Computer Society Press, February 2022. 2, 8, 24, 25, 31, 34
- [Cob65] Alan Cobham. The intrinsic computational difficulty of functions. In Yehoshua Bar-Hillel, editor, Logic, Methodology and Philosophy of Science: Proceedings of the 1964 International Congress (Studies in Logic and the Foundations of Mathematics), pages 24–30. North-Holland Publishing, 1965. 27, 28
- [Coo75] Stephen A. Cook. Feasibly constructive proofs and the propositional calculus (preliminary version). In *Proceedings of the Seventh Annual ACM Symposium on Theory* of Computing, STOC '75, page 83–97, New York, NY, USA, 1975. Association for Computing Machinery. 3, 15, 16, 27, 29
- [CR79] Stephen A. Cook and Robert A. Reckhow. The relative efficiency of propositional proof systems. Journal of Symbolic Logic, 44(1):36–50, 1979. 29
- [CVW18] Yilei Chen, Vinod Vaikuntanathan, and Hoeteck Wee. GGH15 beyond permutation branching programs: Proofs, attacks, and candidates. In Hovav Shacham and Alexandra Boldyreva, editors, CRYPTO 2018, Part II, volume 10992 of LNCS, pages 577–607. Springer, Cham, August 2018. 3, 4, 15, 21, 46, 48, 49, 50, 51, 88, 90
- [DCL08] Giovanni Di Crescenzo and Helger Lipmaa. Succinct np proofs from an extractability assumption. In Logic and Theory of Algorithms: 4th Conference on Computability in Europe, CiE 2008, Athens, Greece, June 15-20, 2008 Proceedings 4, pages 175–185. Springer, 2008. 1
- [DFS24] Thomas Debris-Alazard, Pouria Fallahpour, and Damien Stehlé. Quantum oblivious LWE sampling and insecurity of standard model lattice-based SNARKs. In Bojan Mohar, Igor Shinkar, and Ryan O'Donnell, editors, 56th ACM STOC, pages 423–434. ACM Press, June 2024. 6
- [DGKV22] Lalita Devadas, Rishab Goyal, Yael Kalai, and Vinod Vaikuntanathan. Rate-1 noninteractive arguments for batch-NP and applications. In 63rd FOCS, pages 1057–1068. IEEE Computer Society Press, October / November 2022. 2
- [Din07] Irit Dinur. The pcp theorem by gap amplification. J. ACM, 54(3):12–es, jun 2007. 5, 20, 75

- [FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Berlin, Heidelberg, August 1987. 19
- [Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, 41st ACM STOC, pages 169–178. ACM Press, May / June 2009. 2, 23
- [GGH15] Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, TCC 2015, Part II, volume 9015 of LNCS, pages 498–527. Springer, Berlin, Heidelberg, March 2015. 46, 48, 50
- [GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, 45th ACM STOC, pages 467–476. ACM Press, June 2013. 4, 45
- [GKP⁺13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, 45th ACM STOC, pages 555–564. ACM Press, June 2013. 19, 62, 64, 99, 103, 106, 107, 108
- [GM82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In 14th ACM STOC, pages 365–377. ACM Press, May 1982. 17
- [GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, ACM CCS 2006, pages 89–98. ACM Press, October / November 2006. Available as Cryptology ePrint Archive Report 2006/309. 103
- [GR07] Shafi Goldwasser and Guy N. Rothblum. On best-possible obfuscation. In Salil P. Vadhan, editor, TCC 2007, volume 4392 of LNCS, pages 194–213. Springer, Berlin, Heidelberg, February 2007. 4
- [Gro10] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In Masayuki Abe, editor, ASIACRYPT 2010, volume 6477 of LNCS, pages 321–340. Springer, Berlin, Heidelberg, December 2010. 1
- [Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, EUROCRYPT 2016, Part II, volume 9666 of LNCS, pages 305–326. Springer, Berlin, Heidelberg, May 2016. 1
- [GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, CRYPTO 2013, Part I, volume 8042 of LNCS, pages 75–92. Springer, Berlin, Heidelberg, August 2013. 19, 96, 107

- [GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, 43rd ACM STOC, pages 99–108. ACM Press, June 2011. 1, 23
- [HJKS22] James Hulett, Ruta Jawale, Dakshita Khurana, and Akshayaram Srinivasan. SNARGs for P from sub-exponential DDH and QR. In Orr Dunkelman and Stefan Dziembowski, editors, EUROCRYPT 2022, Part II, volume 13276 of LNCS, pages 520–549. Springer, Cham, May / June 2022. 25
- [HLL23] Yao-Ching Hsieh, Huijia Lin, and Ji Luo. Attribute-based encryption for circuits of unbounded depth from lattices. In *IEEE FOCS*, 2023. 4
- [HW15] Pavel Hubacek and Daniel Wichs. On the communication complexity of secure function evaluation with long output. In Tim Roughgarden, editor, *ITCS 2015*, pages 163–172. ACM, January 2015. 8
- [JJ22] Abhishek Jain and Zhengzhong Jin. Indistinguishability obfuscation via mathematical proofs of equivalence. In *63rd FOCS*, pages 1023–1034. IEEE Computer Society Press, October / November 2022. 2, 3, 7, 14, 26, 27
- [JKKZ21] Ruta Jawale, Yael Tauman Kalai, Dakshita Khurana, and Rachel Yun Zhang. SNARGs for bounded depth computations and PPAD hardness from sub-exponential LWE. In Samir Khuller and Virginia Vassilevska Williams, editors, 53rd ACM STOC, pages 708–721. ACM Press, June 2021. 2
- [JKLV24] Zhengzhong Jin, Yael Kalai, Alex Lombardi, and Vinod Vaikuntanathan. SNARGs under LWE via propositional proofs. In Bojan Mohar, Igor Shinkar, and Ryan O'Donnell, editors, 56th ACM STOC, pages 1750–1757. ACM Press, June 2024., 2, 5, 7, 8, 9, 10, 13, 14, 17, 31, 32, 33, 38
- [JLS21] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from wellfounded assumptions. In Samir Khuller and Virginia Vassilevska Williams, editors, 53rd ACM STOC, pages 60–73. ACM Press, June 2021. 1
- [JLS22] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from LPN over \mathbb{F}_p , DLIN, and PRGs in NC^0 . In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part I*, volume 13275 of *LNCS*, pages 670–699. Springer, Cham, May / June 2022. 1
- [Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In 24th ACM STOC, pages 723–732. ACM Press, May 1992. 19
- [KLV23] Yael Tauman Kalai, Alex Lombardi, and Vinod Vaikuntanathan. SNARGs and PPAD hardness from the decisional Diffie-Hellman assumption. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part II*, volume 14005 of *LNCS*, pages 470–498. Springer, Cham, April 2023. 2
- [KLVW23] Yael Kalai, Alex Lombardi, Vinod Vaikuntanathan, and Daniel Wichs. Boosting batch arguments and RAM delegation. In Barna Saha and Rocco A. Servedio, editors, 55th ACM STOC, pages 1545–1552. ACM Press, June 2023. 2, 8, 24, 25, 34

- [KP16] Yael Tauman Kalai and Omer Paneth. Delegating RAM computations. In Martin Hirt and Adam D. Smith, editors, TCC 2016-B, Part II, volume 9986 of LNCS, pages 91–118. Springer, Berlin, Heidelberg, October / November 2016. 2
- [KPY19] Yael Tauman Kalai, Omer Paneth, and Lisa Yang. How to delegate computations publicly. In Moses Charikar and Edith Cohen, editors, 51st ACM STOC, pages 1115– 1124. ACM Press, June 2019. 2
- [KR09] Yael Tauman Kalai and Ran Raz. Probabilistically checkable arguments. In Shai Halevi, editor, CRYPTO 2009, volume 5677 of LNCS, pages 143–159. Springer, Berlin, Heidelberg, August 2009. 2
- [KRR14] Yael Tauman Kalai, Ran Raz, and Ron D. Rothblum. How to delegate computations: the power of no-signaling proofs. In David B. Shmoys, editor, 46th ACM STOC, pages 485–494. ACM Press, May / June 2014. 2, 9, 30
- [KVZ21] Yael Tauman Kalai, Vinod Vaikuntanathan, and Rachel Yun Zhang. Somewhere statistical soundness, post-quantum security, and SNARGs. In Kobbi Nissim and Brent Waters, editors, TCC 2021, Part I, volume 13042 of LNCS, pages 330–368. Springer, Cham, November 2021. 2, 8, 24, 31, 34
- [LP09] Yehuda Lindell and Benny Pinkas. A proof of security of Yao's protocol for two-party computation. *Journal of Cryptology*, 22(2):161–188, April 2009. 19, 100, 101, 102, 103
- [LPST16a] Huijia Lin, Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation with non-trivial efficiency. In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *PKC 2016, Part II*, volume 9615 of *LNCS*, pages 447–462. Springer, Berlin, Heidelberg, March 2016. 19, 60, 64, 66
- [LPST16b] Huijia Lin, Rafael Pass, Karn Seth, and Sidharth Telang. Output-compressing randomized encodings and applications. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part I*, volume 9562 of *LNCS*, pages 96–124. Springer, Berlin, Heidelberg, January 2016. 19, 60, 64, 66, 69, 70, 71
- [Mer88] Ralph C. Merkle. A digital signature based on a conventional encryption function. In Carl Pomerance, editor, CRYPTO'87, volume 293 of LNCS, pages 369–378. Springer, Berlin, Heidelberg, August 1988. 8, 76, 77
- [Mic94] Silvio Micali. CS proofs (extended abstracts). In 35th FOCS, pages 436–453. IEEE Computer Society Press, November 1994. 1, 6, 19, 23, 75
- [MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, EUROCRYPT 2012, volume 7237 of LNCS, pages 700–718. Springer, Berlin, Heidelberg, April 2012. 50
- [MPV24] Surya Mathialagan, Spencer Peters, and Vinod Vaikuntanathan. Adaptively sound zero-knowledge SNARKs for UP. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part X*, volume 14929 of *LNCS*, pages 38–71. Springer, Cham, August 2024. , 4, 5, 6, 15, 16, 18, 19, 46, 48, 51, 52, 53, 72, 73, 74, 88, 90, 91, 92, 93, 94, 95

- [Nao03] Moni Naor. On cryptographic assumptions and challenges (invited talk). In Dan Boneh, editor, CRYPTO 2003, volume 2729 of LNCS, pages 96–109. Springer, Berlin, Heidelberg, August 2003. 1
- [OPWW15] Tatsuaki Okamoto, Krzysztof Pietrzak, Brent Waters, and Daniel Wichs. New realizations of somewhere statistically binding hashing and positional accumulators. In Tetsu Iwata and Jung Hee Cheon, editors, ASIACRYPT 2015, Part I, volume 9452 of LNCS, pages 121–145. Springer, Berlin, Heidelberg, November / December 2015. 8
- [Par71] Rohit Parikh. Existence and feasibility in arithmetic. The Journal of Symbolic Logic, 36(3):494–508, 1971. 3
- [Pic15] Ján Pich. Logical strength of complexity theory and a formalization of the PCP theorem in bounded arithmetic. Logical Methods in Computer Science, Volume 11, Issue 2, June 2015. 5, 20, 69, 75
- [Pom02] Carl Pomerance. The expected number of random elements to generate a finite abelian group. *Periodica Mathematica Hungarica*, 43(1):191–198, 2002. 59, 60
- [PP22] Omer Paneth and Rafael Pass. Incrementally verifiable computation via rate-1 batch arguments. In 63rd FOCS, pages 1045–1056. IEEE Computer Society Press, October / November 2022. 2
- [PR17] Omer Paneth and Guy N. Rothblum. On zero-testable homomorphic encryption and publicly verifiable non-interactive arguments. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part II*, volume 10678 of *LNCS*, pages 283–315. Springer, Cham, November 2017. 29
- [Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, 37th ACM STOC, pages 84–93. ACM Press, May 2005. 21, 23
- [Ros19] Sheldon Ross. First Course in Probability, A. Pearson Higher Ed, 2019. 95
- [SC04] Michael Soltys and Stephen Cook. The proof complexity of linear algebra. Annals of Pure and Applied Logic, 130(1):277–323, 2004. Papers presented at the 2002 IEEE Symposium on Logic in Computer Science (LICS). 16, 46, 89, 96
- [SD17] Noah Stephens-Davidowitz. On the Gaussian Measure Over Lattices. PhD thesis, New York University, USA, 2017. 21
- [SW05] Amit Sahai and Brent R. Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, EUROCRYPT 2005, volume 3494 of LNCS, pages 457–473. Springer, Berlin, Heidelberg, May 2005. 103
- [SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, 46th ACM STOC, pages 475–484. ACM Press, May / June 2014. 1, 5, 6, 18, 71, 72, 73

- [Tsa22] Rotem Tsabary. Candidate witness encryption from lattice techniques. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part I*, volume 13507 of *LNCS*, pages 535–559. Springer, Cham, August 2022. 4, 15
- [Vai24] Vinod Vaikuntanathan. Personal communication, 2024. 19, 60, 64
- [VWW22] Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. Witness encryption and null-IO from evasive LWE. In Shweta Agrawal and Dongdai Lin, editors, ASIACRYPT 2022, Part I, volume 13791 of LNCS, pages 195–221. Springer, Cham, December 2022. 3, 4, 15, 21, 46, 48, 49, 53, 88
- [Wee22] Hoeteck Wee. Optimal broadcast encryption and CP-ABE from evasive lattice assumptions. In Orr Dunkelman and Stefan Dziembowski, editors, EUROCRYPT 2022, Part II, volume 13276 of LNCS, pages 217–241. Springer, Cham, May / June 2022. 4, 48
- [WW22] Brent Waters and David J. Wu. Batch arguments for NP and more from standard bilinear group assumptions. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 433–463. Springer, Cham, August 2022. 2, 8, 25
- [WW24a] Brent Waters and David J. Wu. Adaptively-sound succinct arguments for NP from indistinguishability obfuscation. In Bojan Mohar, Igor Shinkar, and Ryan O'Donnell, editors, 56th ACM STOC, pages 387–398. ACM Press, June 2024. 1, 5, 6, 34
- [WW24b] Brent Waters and David J. Wu. A pure indistinguishability obfuscation approach to adaptively-sound SNARGs for NP. Cryptology ePrint Archive, Report 2024/933, 2024. 1, 5, 6
- [WZ24] Brent Waters and Mark Zhandry. Adaptive security in SNARGs via iO and lossy functions. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part X*, volume 14929 of *LNCS*, pages 72–104. Springer, Cham, August 2024. 1, 5, 6
- [Yao82] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In 23rd FOCS, pages 160–164. IEEE Computer Society Press, November 1982. 100, 107
- [Zha16] Mark Zhandry. How to avoid obfuscation using witness PRFs. In Eyal Kushilevitz and Tal Malkin, editors, TCC 2016-A, Part II, volume 9563 of LNCS, pages 421–448. Springer, Berlin, Heidelberg, January 2016. 6

A Deferred proofs and constructions from Section 6.3.2

A.1 Proof of Lemma 6.10

Proof. We sketch the polynomial-sized *PV* proof of correctness (following the proofs from [CVW18, VWW22, MPV24]).

If Construction 6.9 does not output \perp , then the following equations must be satisfied:

$$\begin{aligned} \mathbf{D}_{1,b} &= \mathbf{M}_{1,b}\mathbf{A}_1 + \mathbf{E}_{1,b}, \\ \mathbf{A}_{i-1}\mathbf{D}_{i,b} &= \mathbf{M}_{i,b}\mathbf{A}_i + \mathbf{E}_{i,b} \\ \mathbf{A}_{h-1}\mathbf{D}_{h,b} &= \mathbf{M}_{h,b} + \mathbf{E}_{h,b} \end{aligned} \qquad \text{for } 2 \leq i \leq h-1 \end{aligned}$$

For each $\mathbf{x} \in \{0, 1\}^h$, one can inductively prove for $d \leq h - 1$, we have

$$\mathbf{D}_{\mathbf{x}[1:d]} = \mathbf{M}_{\mathbf{x}[1:d]}\mathbf{A}_d + \sum_{j=1}^{d} \mathbf{M}_{\mathbf{x}[1:j-1]} \cdot \mathbf{E}_{j,x_j} \cdot \mathbf{D}_{\mathbf{x}[j+1:d]}$$

where we use the short-hand $\mathbf{S}_{\mathbf{x}[i,j]}$ to denote $\prod_{k=i}^{j} \mathbf{S}_{k,x_k}$. The base case when d = 1 is clearly true by definition. Assuming the statement holds for $d \leq k \leq h-2$, for d = k+1, we have:

$$\begin{aligned} \mathbf{D}_{\mathbf{x}[1:d]} &= \mathbf{D}_{\mathbf{x}[1:d-1]} \cdot \mathbf{D}_{d,x_d} \\ &= \left(\mathbf{M}_{\mathbf{x}[1:d-1]} \mathbf{A}_{d-1} + \sum_{j=1}^{d-1} \mathbf{M}_{\mathbf{x}[1:j-1]} \cdot \mathbf{E}_{j,x_j} \cdot \mathbf{D}_{\mathbf{x}[j+1:d-1]} \right) \cdot \mathbf{D}_{d,x_d} \\ &= \mathbf{M}_{\mathbf{x}[1:d-1]} \mathbf{A}_{d-1} \mathbf{D}_{d,x_d} + \left(\sum_{i=1}^{d-1} \mathbf{M}_{\mathbf{x}[1:j-1]} \cdot \mathbf{E}_{j,x_j} \cdot \mathbf{D}_{\mathbf{x}[j+1:d-1]} \right) \cdot \mathbf{D}_{d,x_d} \\ &= \mathbf{M}_{\mathbf{x}[1:d-1]} (\mathbf{M}_{d,x_d} \mathbf{A}_d + \mathbf{E}_{d,x_d}) + \sum_{i=1}^{d-1} \mathbf{M}_{\mathbf{x}[1:j-1]} \cdot \mathbf{E}_{j,x_j} \cdot \mathbf{D}_{\mathbf{x}[j+1:d]}. \\ &= \mathbf{M}_{\mathbf{x}[1:d]} \mathbf{A}_d + \sum_{i=1}^{d} \mathbf{M}_{\mathbf{x}[1:j-1]} \cdot \mathbf{E}_{j,x_j} \cdot \mathbf{D}_{\mathbf{x}[j+1:d]}. \end{aligned}$$

Here, we used basic associative and distributive properties of matrix multiplication, which we can be proven in PV [SC04]. Moreover, since this is a polynomial-time induction rule (inducts on the length of \mathbf{x}), it can be proven by a polynomial-time PV proof.

Then, for d = h (we deal with this case separately due to the asymmetry in the definition), we have that

$$\begin{aligned} \mathbf{D}_{\mathbf{x}} &= \mathbf{D}_{\mathbf{x}[1:h-1]} \cdot \mathbf{D}_{h,x_h} \\ &= \mathbf{M}_{\mathbf{x}[1:h-1]} (\mathbf{M}_{h,x_h} + \mathbf{E}_{h,x_h}) + \sum_{i=1}^{h-1} \mathbf{M}_{\mathbf{x}[1:j-1]} \cdot \mathbf{E}_{j,x_j} \cdot \mathbf{D}_{\mathbf{x}[j+1:h]} \\ &= \mathbf{M}_{\mathbf{x}} + \sum_{i=1}^{h} \mathbf{M}_{\mathbf{x}[1:j-1]} \cdot \mathbf{E}_{j,x_j} \cdot \mathbf{D}_{\mathbf{x}[j+1:h]}. \end{aligned}$$

Therefore, to bound $\|\mathbf{D}_{\mathbf{x}} - \mathbf{M}_{\mathbf{x}}\|_{\infty}$, it suffices to compute this norm bound via:

$$\begin{split} & \left\| \sum_{j=1}^{h} \left(\prod_{i=1}^{j-1} \mathbf{M}_{i,x_{i}} \right) \cdot \mathbf{E}_{j,x_{j}} \cdot \left(\prod_{i=j+1}^{h} \mathbf{D}_{i,x_{i}} \right) \right\|_{\infty} \\ & \leq h \cdot \max_{j \in [h]} \left(\prod_{i=1}^{j-1} \mathbf{M}_{i,x_{i}} \right) \cdot \mathbf{E}_{j,x_{j}} \cdot \left(\prod_{i=j+1}^{h} \mathbf{D}_{i,x_{i}} \right) \\ & \leq h \cdot \left(\prod_{i=1}^{h-1} m_{i} \right) \cdot B^{h} . \end{split}$$

where we used the fact that ggh.encode ensures that $\|\mathbf{E}_{i,b}\|_{\infty} \leq \sigma\sqrt{n} \leq B$, $\|\mathbf{D}_{i,b}\|_{\infty} \leq \sigma\sqrt{n} \leq B$, and $\|\mathbf{M}_{i,b}\|_{\infty} \leq B$. The first inequality is trivial, and the second repeatedly uses the simple fact that if $\|\mathbf{A} \in \mathbb{Z}_q^{x \times d}\|_{\infty} \leq a$ and $\|\mathbf{B} \in \mathbb{Z}_q^{d \times y}\|_{\infty} \leq b$, then $\|\mathbf{AB}\|_{\infty} \leq dab$, along with Lemma 3.1, for all $i \in [h]$ and $b \in \{0, 1\}$, $\|\mathbf{D}_{i,b}\|_{\infty} \leq \sigma\sqrt{n}$.

A.2 Read- $c \sigma$ -PRFs

In Section 6.3.2, we defined read-once σ -PRFs. Looing forward, for the construction of the designatedverifier SNARG in Appendix A.3, we additionally need to define read- $c \sigma$ -PRFs. We then recall the technique of [MPV24] (also used in prior works such as [CVW18]) to transform any read- $c \sigma$ -PRF into a functionality preserving read-once σ -PRF, which we can then obfuscate with using the average-case obfuscator in Construction 6.13.

Read *c*-branching programs. We additionally consider MBPs on inputs of length ℓ specified by a collection of matrices $(\mathbf{M}_{i,b} : i \in [h := c \cdot \ell], b \in \{0,1\})$ and two vectors \mathbf{u}, \mathbf{v} (all over some ring \mathcal{R} , which, for us, will always be \mathbb{Z}_q for a prime q) such that for all $\mathbf{x} \in \{0,1\}^{\ell}$,

$$f(\mathbf{x}) = \mathbf{u}^T \left(\prod_{i=1}^h \mathbf{M}_{i,x_{j_i}} \right) \mathbf{v} \ ,$$

where $j_i := (i - 1) \mod \ell + 1$. More explicitly,

$$f(\mathbf{x}) = \mathbf{u}^T \big((\mathbf{M}_{1,x_1} \cdot \ldots \cdot \mathbf{M}_{\ell,x_\ell}) \cdot (\mathbf{M}_{\ell+1,x_1} \cdot \ldots \cdot \mathbf{M}_{2\ell,x_\ell}) \cdot \ldots \cdot (\mathbf{M}_{(c-1)\ell+1,x_1} \cdot \ldots \cdot \mathbf{M}_{c\ell,x_\ell}) \big) \mathbf{v} \,.$$

Such MBPs are called *read-c* MBPs. When c = 1, we say the MBP is *read-once*. The following classical result shows that any function computable by logarithmic-depth Boolean circuits can be represented by a matrix branching program.

Theorem A.1 (Barrington's Theorem [Bar86]). If $f : \{0,1\}^{\ell} \to \{0,1\}$ can be computed by a circuit of depth d, then it can be computed by a matrix branching program

$$\{\mathbf{M}_{i,b}: i \in [h := c \cdot \ell], b \in \{0,1\}\}, \mathbf{u}, \mathbf{v}\}$$

where $h = O(4^d)$, and all matrices $\mathbf{M}_{i,b} \in \{0,1\}^{5 \times 5}$ are permutations.

Transforming Read- $c \sigma$ **-PRFs into Read-Once** σ **-PRFs** Notice that for general read- $c \sigma$ matrix PRFs (recall that σ -PRFs were defined in Definition 6.12), it is not the case that all noisy products $\{\mathbf{uM_xv} + e_x\}_{x \in \{0,1\}^h}$ are pseudorandom— the σ -PRF guarantee only requires that noisy products corresponding to inputs $\mathbf{x}' \in \{0,1\}^\ell$ (i.e., with $\mathbf{x} = \mathbf{x}' | \mathbf{x}' | \cdots | \mathbf{x}'$) need be pseudorandom. However, our proof techniques will require all products to be pseudorandom. The following construction is a generic transformation that modifies a read- $c \sigma$ -matrix PRF so that all its products are pseudorandom, without losing functionality. We highlight our modifications in blue.

Construction A.2. On input a read-c MBP $f := (\{\mathbf{M}_{i,b}\}_{i \in [h], b \in \{0,1\}}, \mathbf{u}, \mathbf{v})$ on inputs of length ℓ (where $h = c \cdot \ell$):

- Define $\mathbf{C}_{i,b} = \operatorname{diag}(y_1, \dots, y_{2\ell}, \underbrace{1, \dots, 1}_{\ell \ times}) \in \mathbb{Z}_q^{3\ell \times 3\ell}$, where $h = c \cdot \ell$, and for $1 \leq j \leq \ell$ and $b \in \{0, 1\}$, $y_{2j-b} = \begin{cases} 0 & i \mod \ell = j \ and \ b' = b \\ 1 & otherwise. \end{cases}$
- Sample matrices $\mathbf{S}_{i,b} \leftarrow \mathcal{D}_{\mathbb{Z},2\sqrt{n}}^{n \times n}$, and a vector $\mathbf{a} \leftarrow \mathbb{Z}_q^n$. If $\|\mathbf{S}_{i,b}\|_{\infty} \ge \sigma \sqrt{n}$ for any $\mathbf{S}_{i,b}$, replace with $\mathbf{0}$.
- Output $g := ({\mathbf{M}_{i,b}}_{i \in [h], b \in \{0,1\}}, \mathbf{u}, \mathbf{v}), where$

$$- \mathbf{M}'_{i,b} = \operatorname{diag}(\mathbf{M}_{i,b}, \mathbf{C}_{i,b} \otimes \mathbf{S}_{i,b}),$$

$$- \mathbf{u}' := \left(\underbrace{(1, 1, \dots, 1)}_{2\ell \ times} | \underbrace{-1, -1, \dots, -1}_{\ell \ times} \right)^T \otimes \mathbf{e}_1 \right) \ where \ \mathbf{e}_1 = (1, 0, \dots, 0)^T \in \mathbb{Z}_q^n.$$

$$- \mathbf{v}' := \left(\mathbf{v} \\ \mathbf{1}_{3\ell} \otimes \mathbf{a} \right) \ where \ \mathbf{1}_{3\ell} = (1, \dots, 1)^T \in \mathbb{Z}_q^{3\ell}.$$

Lemma A.3 ([MPV24, Lemma 4.5]). Let $g, \{\mathbf{S}_{i,b}\}_{i \in [h], b \in \{0,1\}}$ be as in Construction A.2 on input f. Then for all $\mathbf{x} \in \{0,1\}^{\ell}$,

$$g(\mathbf{x} \mid \mathbf{x} \mid \cdots \mid \mathbf{x}) = f(\mathbf{x}) ,$$

where $f(\mathbf{x}) = \mathbf{u}^T \mathbf{M}_{\mathbf{x}|\mathbf{x}|...|\mathbf{x}} \mathbf{v}$ and $\mathbf{x} | \mathbf{x} | \cdots | \mathbf{x}$ is the c-fold concatenation of \mathbf{x} with itself. Moreover, there is a polynomial-sized PV proof of the fact that:

for any read-c MBPs f, all outputs g from Construction A.2 on any randomness r satisfies that: for all $\mathbf{x} \in \{0, 1\}^{\ell}$, $g(\mathbf{x} \mid \mathbf{x} \mid \cdots \mid \mathbf{x}) = f(\mathbf{x})$.

Let $\sigma' \geq 2^{h^3\lambda} \cdot (n^2\sigma)^{h+1}$. Assuming subexponential LWE with $2^{n^{\delta}}$ hardness,

- $\{e_{\mathbf{x}} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma'}\}_{\mathbf{x} \in \{0,1\}^{\ell}},$
- $\{e_{\mathbf{y}} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma'}\}_{\mathbf{y} \in \{0,1\}^h}, and$
- $\mathbf{y}' \in \{0,1\}^h$ range over all \mathbf{y}' not of the form $\mathbf{x} \mid \mathbf{x} \mid \cdots \mid \mathbf{x}$,

we have that

$$\{g(\mathbf{y}) + e_{\mathbf{y}}\}_{\mathbf{y} \in \{0,1\}^h}, \{\mathbf{S}_{i,b}\}_{i \in [h], b \in \{0,1\}} \approx_c \{f(\mathbf{x}) + e_{\mathbf{x}}\}_{\mathbf{x} \in \{0,1\}^\ell}, \{\mathcal{U}(\mathbb{Z}_q)\}_{\mathbf{y}'}, \{\mathbf{S}_{i,b}\}_{i \in [h], b \in \{0,1\}^\ell}\}$$

that is, all $poly(2^{n^{\delta}})$ -time adversaries have distinguishing advantage at most $2^{-n^{\delta}}$.

Proof (sketch). We only outline the PV proof here, and we refer the reader to [MPV24] for a proof of the full theorem.

Note that for all $\mathbf{y} \in \{0,1\}^h$, $j \in [\ell]$ and $b \in \{0,1\}$ it is easy to prove that since $\mathbf{C}_{i,b}$ are all diagonal matrices,

$$\mathbf{C}_{\mathbf{y}}[2j-b,2j-b] = \prod_{i=1}^{h} \mathbf{C}_{i,y_i}[2j-b,2j-b]$$

via matrix multiplication and polynomial-time induction in h (and hence polynomial-sized PV).

We note that when $\mathbf{y} = \underbrace{\mathbf{x} | \mathbf{x} | \dots | \mathbf{x}}_{a \text{ times}}$ for $\mathbf{x} \in \{0, 1\}^{\ell}$, then for $j \in [\ell]$ and $b \in \{0, 1\}$

times

$$\mathbf{C}_{\mathbf{y}}[2j-b,2j-b] = \begin{cases} 1 & \text{if } x_j = b \\ 0 & \text{if } x_j \neq b. \end{cases}$$

For $k \ge 2\ell + 1$, we have that $\mathbf{C}_{\mathbf{y}}[k, k] = \prod_{i=1}^{h} \mathbf{C}_{i, y_i}[k, k] = 1$ since all $\mathbf{C}_{i, y_i}[k, k] = 1$ by construction. Therefore, we have that

$$(\underbrace{1,1,\ldots,1}_{2\ell \text{ times}} | \underbrace{-1,-1,\ldots,-1}_{\ell \text{ times}}) \mathbf{C}_{\mathbf{y}} \mathbf{1}_{3\ell} = \sum_{k=1}^{2\ell} \mathbf{C}_{\mathbf{y}}[k,k] - \sum_{k=2\ell+1}^{3\ell} \mathbf{C}_{\mathbf{y}}[k,k] = \ell - \ell = 0.$$

Here, we have used basic properties of matrix multiplication, and hence there is an PV proof of the above fact. Therefore,

$$g(\mathbf{y}) = \mathbf{u}^{T} \mathbf{M}_{\mathbf{y}}^{\prime} \mathbf{v}^{\prime}$$

= $\mathbf{u}^{T} \mathbf{M}_{\mathbf{y}} \mathbf{v} + ((\mathbf{1}_{2\ell} | -\mathbf{1}_{\ell}) \mathbf{C}_{\mathbf{y}} \mathbf{1}_{3\ell}) \otimes (\mathbf{e}_{1}^{T} \mathbf{S}_{\mathbf{y}} \mathbf{a})$
= $\mathbf{u}^{T} \mathbf{M}_{\mathbf{y}} \mathbf{v} + 0 \otimes (\mathbf{e}_{1}^{T} \mathbf{S}_{\mathbf{y}} \mathbf{a})$
= $\mathbf{u}^{T} \mathbf{M}_{\mathbf{y}} \mathbf{v} = f(\mathbf{x})$

as desired. Here, we used basic properties of matrix multiplication and the mixed tensor property, and hence this can be shown in polynomial-sized PV.

A.3 Modified Designated-Verifier SNARG for UP based on [MPV24]

Now recall the construction of [MPV24], and we present our modifications in blue. Our modifications allow us to achieve a polynomial-sized PV proof of completeness for our scheme. We note that [MPV24] formalized this construction in terms of a witness PRF, and then transformed the witness PRF into a dv-SNARG. We present the dv-SNARG directly.

Construction A.4 (dv-SNARG for UP). Fix a relation $\mathcal{R}_{\mathcal{L}}$ for a UP relation. By a classical reduction to Circuit-SAT, we may assume without loss of generality that $\mathcal{R}(\mathbf{x}, \mathbf{w})$ is represented by a circuit of depth $O(\log \ell)$ for inputs $\mathbf{x} \in \{0, 1\}^{\ell}$. Then, use Barrington's theorem (Theorem A.1) to construct a read-c MBP

$$\Gamma = \left(\{ \mathbf{M}_{i,b} \in \{0,1\}^{v \times v} \}_{i \in [h], b \in \{0,1\}}, \mathbf{u}, \mathbf{v} \in \{0,1\}^{v}, \right) ,$$

such that $\Gamma(\mathbf{x}, \mathbf{w}) := \mathbf{u} \mathbf{M}_{(\mathbf{x}, \mathbf{w})|(\mathbf{x}, \mathbf{w})|\dots|(\mathbf{x}, \mathbf{w})} \mathbf{v} = 0 \iff \mathcal{R}(\mathbf{x}, \mathbf{w}) = 1$. (From here on, we treat the branching program as the relation circuit instead.) Set $h := c \cdot \ell$ (length of the branching program). Set n such that $h + \lambda \leq n^{\delta/20}$. Set $q = 2^{n^{\delta}}$, and $p = 2^{\kappa}$, for some $\kappa = \lambda + \omega(\log(h + T + \lambda))$.

Gen $(1^{\lambda}, \mathcal{R}_{\mathcal{L}})$: • For $i \in [h]$, it samples $\mathbf{S}_{i,b} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}^{n \times n}$, and for $i \leq \ell$, it samples $\mathbf{T}_{i,b} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}^{n \times n}$, where $\sigma = \sqrt{2n}$. If $\|\mathbf{S}_{i,b}\|_{\infty} \geq \sigma \sqrt{n}$ for any $\mathbf{S}_{i,b}$, replace with **0**. Similarly, if any $\|\mathbf{T}_{i,b}\|_{\infty}$, replace with **0**.

- Sample $\mathbf{a}, \mathbf{b} \leftarrow \mathbb{Z}_q^n$.
- Set matrices

$$\mathbf{Q}_{i,b} = \begin{cases} \begin{pmatrix} \mathbf{M}_{i,b} \otimes \mathbf{S}_{i,b} & \\ & \mathbf{T}_{i,b}, \end{pmatrix} & 1 \le i \le \ell, \\ \begin{pmatrix} \mathbf{M}_{i,b} \otimes \mathbf{S}_{i,b} & \\ & \mathbf{I} \end{pmatrix} & \ell < i \le h, \\ \mathbf{L} = \begin{pmatrix} (\mathbf{u}^T \otimes \mathbf{e}_1^T) \mid \mathbf{e}_1 \end{pmatrix}, \\ \mathbf{R} = \begin{pmatrix} \mathbf{v} \otimes \mathbf{a} \\ \mathbf{b} \end{pmatrix}, \end{cases}$$

where \mathbf{e}_1 denotes the vector $(1, 0, \dots, 0)^T \in \mathbb{Z}_q^n$. and sets $\mathcal{F} := (\mathbf{L}, {\mathbf{Q}_{i,b}}_{i \in [h], b \in \{0,1\}}, \mathbf{R})$.

- Apply the read-c to read-once transformation in Construction A.2 to obtain a new MBP

 F' = (L, ' {Q'_{i,b}}_{i∈[h],b∈{0,1}}, R')
- Set $pk = \mathcal{O}(\mathcal{F}')$, where \mathcal{O} is the obfuscation algorithm from Theorem 6.14.
- Set the verifier key $vk = \{ \{ \mathbf{T}_{i,b} \}_{i \in [\ell, b \in \{0,1\}\}}, \mathbf{b} \}.$
- If \mathcal{O} outputs \perp , instead set $(pk, vk) = (\perp, \perp)$.

 $\mathcal{P}(\mathsf{pk}, \mathbf{x}, \mathbf{w})$: If $\mathcal{R}(\mathbf{x}, \mathbf{w}) = 0$, output \perp . Else, compute

$$\pi := \lfloor \mathcal{O}.\mathsf{Eval}(\mathsf{pk}, \underbrace{\mathbf{x}, \mathbf{w} | \dots | \mathbf{x}, \mathbf{w}}_{c \ times}) \rceil_p \in \mathbb{Z}_p$$

Output π . Here, $\lfloor a \rceil_p$ takes $a \in \mathbb{Z}_q$ and identifies a with a number $\{0, 1, \ldots, q-1\}$, and outputs $\lfloor (p/q) \cdot a \rfloor$.

 $\mathcal{V}(crs, \mathbf{x}, \pi)$: If $vk = \bot$, always output 1. Else,

- Compute $\pi' = \lfloor \mathbf{e}_1^T \mathbf{T}_{\mathbf{x}} \cdot \mathbf{b} \rfloor_p \in \mathbb{Z}_p$.
- Accept if $|\pi \pi' \pmod{p}| \le 1$. (In [MPV24], proof was only accepted if $\pi = \pi'$.)

Remark A.5. At a high level, the main modification that we make (other than truncating Gaussians) is that the verifier accepts even if the prover message is "off by 1" from the verifier's computed message. Although this is statistically unlikely assuming subexponential LWE, it is unclear how to prove this using standard properties of linear algebra. However, we can show using an PV proof that the prover message is at most "off by 1". We then show that if one can break the soundness of the new scheme, one can also break the soundness of the scheme in [MPV24].

Claim A.6. Construction A.4 is a perfectly complete designated-verifier SNARG for UP with a PV proof of completeness, i.e. a PV proof of the following statements:

for all randomness r, instances x and witnesses w, for $(pk, vk) \leftarrow Gen(1^{\lambda}, \mathcal{R})$, if $\Gamma(\mathbf{x}, \mathbf{w}) = 0^{30}$ then $\mathcal{V}(vk, \mathbf{x}, \mathcal{P}(pk, \mathbf{x}, \mathbf{w})) = 1$.

Proof (sketch). If $pk = \bot$, then by construction, $vk = \bot$ as well. In this case, $\mathcal{V}(vk, \mathbf{x}, \pi)$ accepts all inputs, and hence the completeness condition is trivially satisfied. Since these claims follow from simple implications, this can be shown in PV.

Otherwise, it must be the case that \mathcal{O} does not output \perp , in which case the completeness condition in Theorem 6.14 guarantees that

$$|\mathcal{F}'(\underbrace{(\mathbf{x},\mathbf{w})|\dots|(\mathbf{x},\mathbf{w})}_{c \text{ times}}) - \mathcal{O}.\mathsf{Eval}(P,\underbrace{(\mathbf{x},\mathbf{w})|\dots|(\mathbf{x},\mathbf{w})}_{c \text{ times}})| \le h(mB)^{h-1}$$

where we denote by $\mathcal{F}'(\mathbf{y}) = \mathbf{L}'\mathbf{Q}'_{\mathbf{y}}\mathbf{R}'$. Moreover, this inequality has a PV of completeness. Additionally, by Lemma A.3, we have that

$$\mathcal{F}'(\underbrace{(\mathbf{x},\mathbf{w})|\ldots|(\mathbf{x},\mathbf{w})}_{c \text{ times}}) = \mathcal{F}(\underbrace{(\mathbf{x},\mathbf{w})|\ldots|(\mathbf{x},\mathbf{w})}_{c \text{ times}}),$$

and this fact has a polynomial-sized PV proof.

Moreover, via basic linear algebra and the mixed tensor property, one can prove with a polynomial sized PV proof that that if $\Gamma(\mathbf{x}, \mathbf{w}) = \mathbf{u}^T \mathbf{M}_{(\mathbf{x}, \mathbf{w})|\dots|(\mathbf{x}, \mathbf{w})} \mathbf{v} = 0$, then

$$\mathcal{F}((\mathbf{x}, \mathbf{w})| \dots | (\mathbf{x}, \mathbf{w})) = \mathbf{L} \mathbf{Q}_{(\mathbf{x}, \mathbf{w})| \dots | (\mathbf{x}, \mathbf{w})} \mathbf{R}$$
$$= \mathbf{e}_1^T \cdot \mathbf{T}_{\mathbf{x}} \cdot \mathbf{b}.$$

By construction in Construction 6.9, Construction A.2 and Construction A.4, the matrix entries are bounded by $\sigma\sqrt{n}$ (since we sample $\mathbf{S}_{i,b}$ and $\mathbf{T}_{i,b}$ from truncated Gaussians). Also, the size m of the matrices is $7n + 3n\ell \leq n^2$ for large n.

Therefore, combining the above equations, we have that

$$|\mathbf{e}_1^T \cdot \mathbf{T}_{\mathbf{x}} \cdot \mathbf{b} - \mathcal{O}.\mathsf{Eval}(P, \underbrace{(\mathbf{x}, \mathbf{w}) | \dots | (\mathbf{x}, \mathbf{w})}_{c \text{ times}})| \le h(m\sigma\sqrt{n})^{h-1} \le h \cdot (2n^3)^h \le \frac{q}{2p}$$

since we chose $h + \lambda \leq n^{\delta/20}$ and $p = 2^{\kappa}$, where $\kappa = \lambda + \omega(\log(\lambda, h))$. Therefore, we have that

$$\left| \begin{bmatrix} \mathbf{e}_1^T \cdot \mathbf{T}_{\mathbf{x}} \cdot \mathbf{b} \end{bmatrix}_p - \lfloor \mathcal{O}.\mathsf{Eval}(P, \underbrace{(\mathbf{x}, \mathbf{w}) | \dots | (\mathbf{x}, \mathbf{w})}_{c \text{ times}}) \right]_p \right| = |\pi' - \pi| \le 1$$

³⁰Recall that $\Gamma(\mathbf{x}, \mathbf{w}) = 0$ if and only if $\mathcal{R}(\mathbf{x}, \mathbf{w}) = 1$.

by definition of the rounding function. Hence, the verifier in fact accepts a correctly computed proof, as desired.

By concatenating all of the proofs of the above steps, we obtain a polynomial-sized PV proof for this fact.

To prove soundness of the scheme, we reduce the soundness to that of the original construction of [MPV24].

Claim A.7. Construction 6.5 is an adaptively sound³¹ designated-verifier SNARG for $\mathcal{L} \in UP$ assuming LWE and evasive LWE.

Proof. We reduce the adaptive soundness of Construction A.4 to the soundness of the construction in [MPV24].

Sampling from truncated Gaussians and applying the the modified obfuscation scheme only change the distribution of (pk, vk) by a TV distance of $2^{-n} \cdot poly(n, h)$.

Now, we argue that if there exists an adaptive adversary \mathcal{P}^* that breaks the adaptive soundness of Construction A.4 relative to verifier \mathcal{V} , then there exists a \mathcal{P}^* that breaks the adaptive soundness of [MPV24] scheme relative to \mathcal{V}' , which only accepts if $\pi = \pi'$ rather than if $|\pi - \pi'| \leq 1$. If \mathcal{P}^* sends the instance-proof pair (x, π) , choose $\pi^* \leftarrow \{\pi - 1, \pi, \pi + 1\}$, and send (x^*, π^*) to \mathcal{V}' . Since π was accepted, it is the case that $|\pi - \pi'| \leq 1$. Therefore, π^* is equal to π' with probability 1/3. Hence, there exists an adaptive adversary that also breaks the scheme in [MPV24] with only a 1/3 factor loss in advantage. Therefore, the modified scheme is sound assuming the scheme in [MPV24] (which is true under subexponential LWE and evasive LWE).

Definition A.8. We say that a distribution \mathcal{D} is ϵ -transparent if there exists a polynomial-time deterministic sampler Samp and a polynomial time (possibly randomized) simulator \mathcal{S} such that the following distributions are ϵ -statistically indistinguishable:

- H_0 : Given $r \leftarrow \{0,1\}^{\ell}$, output $(\mathsf{Samp}(r), r)$.
- H_1 : Given $x \leftarrow D$, output (x, S(x)).

i.e. it is possible to efficiently simulate the randomness used to sample from \mathcal{D} up to a statistical distance of ϵ .

Lemma A.9. Let $\sigma = O(n^c)$ for some constant c > 0 (i.e. polynomial variance). Then, $\mathcal{D}_{\mathbb{Z},\sigma}$ is a $2^{O(n)}$ -transparent distribution.

This follows directly from inverse sampling since the variance in poly(n) (see for example [Ros19]).

Claim A.10. The common reference string pk generated by Construction A.4 is $2^{-\ell}$ indistinguishable from a transparent distribution \mathcal{D} .

Proof (sketch). Recall that pk in Construction A.4 is generated via Construction 6.13. As shown in [MPV24, Lemma 5.4], we can view \mathcal{F}' as a σ' -PRF, for $\sigma' = 2^{h^3\lambda} \cdot (n^2\sigma)^{h+2}$, one can apply the σ -PRF indistinguishability guarantee of Theorem 6.14 to obtain that pk is $2^{n^{\delta}}$ -indistinguishable from some transparent distribution \mathcal{D} . In particular, since $n^{\delta} \geq \ell$, we have that the the construction is in fact 2^{ℓ} -indistinguishable from a transparent distribution \mathcal{D} .

 $^{^{31}}$ It is in fact reusably sound, but we do not need this additional property for our transformation in Theorem 5.1.

B PV Proof for Leveled GSW Fully Homomorphic Encryption

In this section, we recall and prove that the GSW [GSW13] encryption scheme (with minor modifications) has an PV proof of correctness.

Algorithm 11 The leveled Gentry-Sahai-Waters [GSW13] encryption scheme for circuits of depth at most D. All modifications to help achieve the proof of correctness are in blue.

Let $w := \lceil \log q \rceil - 1$ and $\mathbf{g} := (1, 2, 4, \dots, 2^w)^T$. Let $\mathbf{G} := (\mathbf{I}_{n+1} \otimes \mathbf{g}) \in \mathbb{Z}_q^{(n+1) \times (n+1)w}$.

Let \mathbf{G}^- denotes the bit-decomposition inverse mapping; i.e., $\mathbf{G}^-(\mathbf{A}) = \mathbf{B}$ if \mathbf{w} has $\{0, 1\}$ entries, and $\mathbf{G}\mathbf{B} = \mathbf{A}$.

• Key generation: GSW.Gen $(1^{\lambda}, 1^{D})$, and choose $q \geq 2^{2D}(n+1)^{2D}w^{2D}\sigma$, and outputs $\mathbf{s}^{T} \leftarrow \mathbb{Z}_{q}^{n}$. Sample randomness $\mathbf{A} \leftarrow \mathbb{Z}_{q}^{n \times (n+1)w}$ and $\mathbf{e} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}^{(n+1)w}$. If $||\mathbf{e}||_{\infty} \geq \sigma \sqrt{(n+1)w}$, set $\mathbf{e} = \mathbf{0}$. Now, set $\mathbf{sk} = \mathbf{s}$ and

$$\mathsf{pk} = \begin{pmatrix} \mathbf{A} \\ -\mathbf{s}^T \mathbf{A} + \mathbf{e}^T \end{pmatrix}.$$

- Encryption: GSW.Enc_{pk} $(m; \mathbf{R})$ takes as input m and randomness $\mathbf{R} \leftarrow \{0, 1\}^{(n+1)w \times (n+1)w}$ and outputs $\mathsf{pk} \cdot \mathbf{R} + m\mathbf{G}$.
- Decryption: On input $\mathbf{C} \in \mathbb{Z}_q^{(n+1) \times (n+1)w}$, GSW.Dec_s proceeds as follows:
 - Let $\mathbf{w} \leftarrow \mathbb{Z}_q^{n+1}$ such that $\mathbf{w} = (0, 0, \dots, 0, \lfloor q/2 \rfloor)^T$.
 - Compute $\mu = (\mathbf{s}^T | 1) \mathbf{C} \cdot \mathbf{G}^-(\mathbf{w}).$
 - If $|\mu| \leq |q/2|$, output 0. Else, output 1.
- Homomorphic computation: On input a NAND circuit **C** of depth at most L) and values for its input gates, GSW.Eval evaluates the circuit gate by gate. Given a gate with inputs C_1 and C_2 , the output of the gate is defined as $\mathbf{G} \mathbf{C}_1 \cdot \mathbf{G}^-(\mathbf{C}_2)$.

Lemma B.1. Algorithm 11 is a correct leveled fully homomorphic scheme with a PV proof of correctness, i.e. a proof of the fact that

for all $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{GSW}.\mathsf{Gen}(1^{\lambda}, 1^d)$, for all $m_1, \ldots, m_{\ell} \in \{0, 1\}$, and $C : \{0, 1\}^{\ell} \rightarrow \{0, 1\}$ of depth at most D, for all $\mathsf{ct}_i \leftarrow \mathsf{GSW}.\mathsf{Enc}_{\mathsf{pk}}(m_i)$, if $\mathsf{ct}_{out} \leftarrow \mathsf{GSW}.\mathsf{Eval}(\mathsf{pk}, C, (\mathsf{ct}_1, \ldots, \mathsf{ct}_{\ell}))$, then $b \leftarrow \mathsf{GSW}.\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}_{out})$ satisfies $b = C(m_1, \ldots, m_{\ell})$.

Proof sketch. We simply argue that the proof strategy from [GSW13] can be formalized in PV. The only difference in our algorithm is that we sample from a *truncated* Gaussian. Here is the main fact we use:

if
$$\mathbf{A} \in \mathbb{Z}^{n \times m}$$
 and $\mathbf{B} \in \mathbb{Z}^{m \times p}$, $|\mathbf{AB}|_{\infty} \leq m |\mathbf{A}|_{\infty} \cdot |\mathbf{B}|_{\infty}$.

This can be proven easily using basic arithmetic and matrix properties as shown in [Bus86, SC04], and hence can be easily shown to have a PV proof.

We proceed by induction on the depth d of the C input to GSW.Eval to show the following claim:

$$\left|\mathbf{v}^T \cdot \mathsf{GSW}(\mathsf{pk}, C, (\mathsf{ct}_1, \dots, \mathsf{ct}_\ell)) - b \cdot \mathbf{v}^T \mathbf{G}\right|_{\infty} \leq 2^d \cdot (n+1)^{d+2} w^{d+2} \cdot \sigma.$$

where ct_i is a fresh encryption of the message m_i , and $b = C(m_1, \ldots, m_\ell)$.

Base case. First consider d = 0 (i.e., no homomorphic operations are performed the ciphertext). Set $\mathbf{v} = (\mathbf{s}^T | 1)^T$. Then, given $\mathbf{C} \leftarrow \mathsf{GSW}.\mathsf{Enc}_{\mathsf{pk}}(m; \mathbf{R})$

$$\mathbf{v}^{T}\mathbf{C} = \mathbf{v}^{T} \left(\begin{pmatrix} \mathbf{A} \\ -\mathbf{s}^{T}\mathbf{A} + \mathbf{e}^{T} \end{pmatrix} \cdot \mathbf{R} + m\mathbf{G} \right)$$
$$= (\mathbf{s}^{T} \mid 1) \begin{pmatrix} \mathbf{A} \\ -\mathbf{s}^{T}\mathbf{A} + \mathbf{e}^{T} \end{pmatrix} \cdot \mathbf{R} + m\mathbf{v}^{T}\mathbf{G}$$
$$= \mathbf{e}^{T}\mathbf{R} + m\mathbf{v}^{T}\mathbf{G}.$$

Hence,

$$\begin{aligned} |\mathbf{v}^T \mathbf{C} - m \mathbf{v}^T \mathbf{G}|_{\infty} &\leq |\mathbf{e}^T \mathbf{R}|_{\infty} \\ &\leq (n+1)w \cdot |\mathbf{e}|_{\infty} \qquad \qquad \text{(Since } \mathbf{R} \text{ is } 0/1) \\ &\leq (n+1)w \sqrt{(n+1)w} \cdot \sigma \qquad \qquad \text{(Construction of } \mathbf{e}) \\ &\leq (n+1)^2 w^2 \sigma \qquad \qquad \text{(Loose upper bound)} \end{aligned}$$

where we use the fact that **R** is a matrix with 0/1 entries with dimension $(n + 1)w \times (n + 1)w$. Hence, the induction hypothesis holds.

Inductive Step. Suppose the claim holds for depth d. Then consider a circuit C of depth d + 1. Suppose the one has performed homomorphic evaluations up to the final output gate to obtain ciphertexts C_1 and C_2 . Suppose that the correctly evaluated values of these gates (without FHE) are m_0 and m_1 . By the inductive hypothesis,

$$\left| \mathbf{v}^T \mathbf{C}_1 - m_1 \mathbf{v}^T \mathbf{G} \right|_{\infty} \leq 2^d \cdot (n+1)^{d+2} w^{d+2} \cdot \sigma$$
$$\left| \mathbf{v}^T \mathbf{C}_2 - m_2 \mathbf{v}^T \mathbf{G} \right|_{\infty} \leq 2^d \cdot (n+1)^{d+2} w^{d+2} \cdot \sigma.$$

Now, we prove the following claim.

Claim B.2. Let $\mathbf{v} = (\mathbf{s}^T \mid 1)^T$. Suppose that two ciphertexts \mathbf{C}_1 and \mathbf{C}_2 corresponding to encryptions of messages m_1 and m_2 satisfy the following inequalities:

$$\begin{aligned} \left| \mathbf{v}^T \mathbf{C}_1 - m_1 \mathbf{v}^T \mathbf{G} \right|_{\infty} &\leq \delta_1 \\ \left| \mathbf{v}^T \mathbf{C}_2 - m_2 \mathbf{v}^T \mathbf{G} \right|_{\infty} &\leq \delta_2 \end{aligned}$$

Then,

$$|\mathbf{v}^T \cdot \mathsf{GSW}.\mathsf{Eval}(\mathsf{pk}, \mathrm{NAND}, (\mathbf{C}_1, \mathbf{C}_2)) - \mathrm{NAND}(m_1, m_2) \cdot \mathbf{v}^T \mathbf{G}|_{\infty} \leq \delta_2 + (n+1)w \cdot \delta_1$$

Proof. Recall

$$\mathbf{v}^{T} \cdot \mathbf{GSW}.\mathbf{Eval}(\mathbf{pk}, \mathbf{NAND}, (\mathbf{C}_{1}, \mathbf{C}_{2})) = \mathbf{v}^{T}(\mathbf{G} - \mathbf{C}_{1} \cdot \mathbf{G}^{-}(\mathbf{C}_{2})) = \mathbf{v}^{T}\mathbf{G} - \mathbf{v}^{T}\mathbf{C}_{1} \cdot \mathbf{G}^{-}(\mathbf{C}_{2})$$
(Distributive law)
$$= \mathbf{v}^{T}\mathbf{G} - \mathbf{v}^{T}\mathbf{C}_{1} \cdot \mathbf{G}^{-}(\mathbf{C}_{2})$$
(Substituting $\mathbf{v}^{T}\mathbf{C}_{1}$)
$$= \mathbf{v}^{T}\mathbf{G} - (m_{1}\mathbf{v}^{T}\mathbf{G} + \mathbf{e}_{1}) \cdot \mathbf{G}^{-}(\mathbf{C}_{2})$$
(Distributive law)
$$= \mathbf{v}^{T}\mathbf{G} - m_{1}\mathbf{v}^{T}\mathbf{G} \cdot \mathbf{G}^{-}(\mathbf{C}_{2}) + \mathbf{e}_{1} \cdot \mathbf{G}^{-}(\mathbf{C}_{2})$$
(Distributive law)
$$= \mathbf{v}^{T}\mathbf{G} - m_{1}\mathbf{v}^{T}\mathbf{C}_{2} + \mathbf{e}_{1} \cdot \mathbf{G}^{-}(\mathbf{C}_{2})$$
(Using $\mathbf{G} \cdot \mathbf{G}^{-}(\mathbf{C}) = \mathbf{C}$)
$$= \mathbf{v}^{T}\mathbf{G} - m_{1}(m_{2}\mathbf{v}^{T}\mathbf{G} + \mathbf{e}_{2}) + \mathbf{e}_{1} \cdot \mathbf{G}^{-}(\mathbf{C}_{2})$$
(Substituting $\mathbf{v}^{T}\mathbf{C}_{2}$)
$$= \mathbf{v}^{T}\mathbf{G} - m_{1}m_{2}\mathbf{v}^{T}\mathbf{G} + (m_{1}\mathbf{e}_{2} + \mathbf{e}_{1} \cdot \mathbf{G}^{-}(\mathbf{C}_{2}))$$
(Distributive law)
$$= (1 - m_{1}m_{2})\mathbf{v}^{T}\mathbf{G} + (m_{1}\mathbf{e}_{2} + \mathbf{e}_{1} \cdot \mathbf{G}^{-}(\mathbf{C}_{2}))$$
(Scalar multiplication)
$$= \mathbf{NAND}(m_{1}, m_{2}) \cdot \mathbf{v}^{T}\mathbf{G} + (m_{1}\mathbf{e}_{2} + \mathbf{e}_{1} \cdot \mathbf{G}^{-}(\mathbf{C}_{2}))$$
(Definition of NAND)

Therefore, rearranging, we have that:

$$\begin{aligned} \left| \mathbf{v}^T \cdot \mathsf{GSW}.\mathsf{Eval}(\mathsf{pk}, \mathrm{NAND}, (\mathbf{C}_1, \mathbf{C}_2)) - \mathrm{NAND}(m_1, m_2) \cdot \mathbf{v}^T \mathbf{G} \right|_{\infty} \\ &\leq \left| m_1 \mathbf{e}_2 + \mathbf{e}_1 \cdot \mathbf{G}^-(\mathbf{C}_2) \right|_{\infty} \\ &\leq \left| \mathbf{e}_2 \right|_{\infty} + (n+1)w \cdot \left| \mathbf{e}_1 \right|_{\infty} \\ &\leq \delta_2 + (n+1)w \cdot \delta_1. \end{aligned}$$

Therefore, we have that:

$$\begin{aligned} \left| \mathbf{v}^{T} \cdot \mathsf{GSW}(\mathsf{pk}, C, (\mathsf{ct}_{1}, \dots, \mathsf{ct}_{\ell})) - m \cdot \mathbf{v}^{T} \mathbf{G} \right|_{\infty} \\ &= \left| \mathbf{v}^{T} \cdot \mathsf{GSW}(\mathsf{pk}, \mathrm{NAND}, (\mathbf{C}_{1}, \mathbf{C}_{2})) - m \cdot \mathbf{v}^{T} \mathbf{G} \right|_{\infty} \\ &\leq \left| \mathbf{v}^{T} \mathbf{C}_{2} - m_{2} \mathbf{v}^{T} \mathbf{G} \right|_{\infty} + (n+1) \cdot w \left| \mathbf{v}^{T} \mathbf{C}_{1} - m_{1} \mathbf{v}^{T} \mathbf{G} \right|_{\infty} \\ &\leq 2(n+1) \cdot w \cdot 2^{d} \cdot (n+1)^{d+2} w^{d+2} \cdot \sigma \\ &\leq 2^{d+1} \cdot (n+1)^{d+3} w^{d+3} \sigma \end{aligned}$$

as desired.

Now, consider the decryption algorithm. Let $\mathbf{C} \leftarrow \mathsf{GSW}.\mathsf{Eval}(\mathsf{pk}, C, (\mathsf{ct}_1, \dots, \mathsf{ct}_\ell))$ where circuit C has depth at most D. By the inductive hypothesis, there exists some vector \mathbf{u} with $|\mathbf{u}|_{\infty} \leq 2^D \cdot (n+1)^{D+2} w^{D+2} \sigma$ such that $\mathbf{v}^T \mathbf{C} = m \mathbf{v}^T \mathbf{G} + \mathbf{u}$.

$$\mathbf{v}^{T}\mathbf{C}\mathbf{G}^{-}(\mathbf{w}) = (m\mathbf{v}^{T}\mathbf{G} + \mathbf{u})\mathbf{G}^{-}(\mathbf{w})$$
(Substitution)
$$= m\mathbf{v}^{T}\mathbf{G}\mathbf{G}^{-}(\mathbf{w}) + \mathbf{u}\mathbf{G}^{-}(\mathbf{w})$$
(Distributive law)
$$= m\mathbf{v}^{T}\mathbf{w} + \mathbf{u}\mathbf{G}^{-}(\mathbf{w})$$
(Using $\mathbf{G} \cdot \mathbf{G}^{-}(\mathbf{w}) = \mathbf{w}$)
$$= m\lfloor q/2 \rfloor + \mathbf{u}\mathbf{G}^{-}(\mathbf{w}).$$
(Using $\sum_{i} \mathbf{v}_{i}\mathbf{w}_{i} = \lfloor q/2 \rfloor$)

Note that

$$\begin{aligned} \left| \mathbf{v}^T \mathbf{C} \mathbf{G}^-(\mathbf{w}) - m \lfloor q/2 \rfloor \right| &\leq |\mathbf{u} \mathbf{G}^-(\mathbf{w})|_{\infty} \\ &\leq |\mathbf{u}|_{\infty} \cdot (n+1) \\ &\leq 2^D \cdot (n+1)^{D+3} w^{D+2} \sigma \leq q/8 \end{aligned}$$

by our choice of q. Therefore, $\mathsf{GSW}.\mathsf{Dec}_{\mathbf{s}}(\mathbf{C}) = m$, completing the proof.

C PV Proof for Succinct Functional Encryption

In this section, we show that the succinct functional encryption construction of $[GKP^+13]$ can be instantiated with a PV proof of correctness. We instantiate the three main ingredients in the construction with PV proofs of correctness:

- Fully homomorphic encryption (Appendix B)
- Garbled circuits (Appendix C.1)
- Two-outcome attribute-based encryption (Appendix C.2)

With these ingredients, the proof of correctness of the succinct functional encryption follows straightforwardly (Appendix C.3).

C.1 *PV* Proof for Garbled circuits

Definition C.1 (Garbled circuits). A garbling scheme for a family of circuits $C = \{C_n\}_{n \in \mathbb{N}}$ with C_n is a set of boolean circuits taking as input n bits, is a tuple of algorithm GC = (GC.Garble, GC.Enc, GC.Eval) such that

- GC.Garble $(1^{\lambda}, C)$: Takes as input security parameter λ and a circuit $C \in C_n$, and outputs the garbled circuit Γ and a secret key sk.
- GC.Enc(sk, x): Takes as input the secret key sk and input x ∈ {0,1}* and outputs an encoding c.
- GC.Eval (Γ, c) : Takes as input a garbled circuit Γ , an encoding c, and outputs a value y.

We require that:

Correctness. For any $\lambda, n \in \mathbb{N}$, there is a negligible function μ such that

$$\Pr\left[\begin{array}{cc} (\Gamma,\mathsf{sk}) \leftarrow \mathsf{GC}.\mathsf{Garble}(1^{\lambda},C), \\ C(x) = y \ : \ c \leftarrow \mathsf{GC}.\mathsf{Enc}(\mathsf{sk},x), \\ y \leftarrow \mathsf{GC}.\mathsf{Eval}(\Gamma,c) \end{array}\right] \ge 1 - \mathsf{negl}(\lambda).$$

We say that the algorithm has perfect correctness if the above probability is exactly 1.

Efficiency. We require the following efficiency guarantees from the algorithm:

• GC.Garble and GC.Eval algorithm runs in time $poly(\lambda, |C|, |x|)$.

- GC.Enc runs in time $poly(\lambda, |x|)$, independent of |C|.
- Input and circuit privacy. A garbled circuit algorithm GC is input and circuit private if there exists a simulator Sim such that for every PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, and for sufficiently large security parameters λ ,

$$\left| \Pr \left[\begin{array}{cc} (x, \Gamma, \mathsf{st}) \leftarrow \mathcal{A}_1(1^{\lambda}), \\ \mathcal{A}_2(\Gamma, c, \mathsf{st}) = 1 & : & (\Gamma, \mathsf{sk}) \leftarrow \mathsf{GC}.\mathsf{Garble}(1^{\lambda}, C), \\ c \leftarrow \mathsf{GC}.\mathsf{Enc}(\mathsf{sk}, x) \end{array} \right] \\ - \Pr \left[\begin{array}{cc} \mathcal{A}_2(\tilde{\Gamma}, \tilde{c}, \mathsf{st}) = 1 & : & (x, \Gamma, \mathsf{st}) \leftarrow \mathcal{A}_1(1^{\lambda}), \\ (\tilde{\Gamma}, \tilde{c}) \leftarrow \mathsf{Sim}(1^{\lambda}, C(x), 1^{|C|}, 1^{|x|}) \end{array} \right] \right| \leq \mathsf{negl}(\lambda)$$

We say that the construction is subexponentially secure if the above absolute value is instead bounded by $2^{-\lambda^{\epsilon}}$ for some $0 < \epsilon < 1$.

C.1.1 Construction and PV proof

In this section, we show that Yao's construction [Yao82] for Garbled Circuits (as in Definition C.1), following the exposition from [LP09] has a PV proof of correctness (with some minor modifications). In fact, [LP09] detail a proof of correctness, and argue how to achieve perfect correctness. First, we recap the main tool for the construction.

Definition C.2 (Secret-key encryption with elusive range, [LP09]). Consider a secret-key encryption scheme SKE = (SKE.Gen, SKE.Enc, SKE.Dec), and let $Range_n(sk) = \{Enc_{sk}(x)\}_{x \in \{0,1\}^n}$.

• (Elusive range) We say SKE has an elusive range if for every PPT adversary A, we have that

 $\Pr_{\mathsf{sk}\leftarrow\mathsf{SKE}.\mathsf{Gen}(1^\lambda)}[\mathcal{A}(1^\lambda)\in\mathsf{Range}_n(\mathsf{sk})]=\mathsf{negl}(\lambda).$

(Efficiently verifiable range) We say SKE has an efficiently verifiable range if there exists an algorithm M such that M(sk, ct) = 1 ⇐⇒ ct ∈ Range_n(sk).

Without loss of generality, we can define $\text{Dec}_{sk}(ct) = \perp$ if $ct \notin \text{Range}_n(sk)$ (by using M in the decryption algorithm).

Construction C.3 (SKE with efficiently verifiable range, from [LP09]). Let $\mathcal{F} = \{f_k\}_k$ be a family of PRF families where $f_k : \{0,1\}^n \to \{0,1\}^{n+\lambda}$ for $k \in \{0,1\}^n$. Then,

- SKE.Gen $(1^{\lambda}, 1^{n})$: Sample $k \leftarrow \mathsf{PRF.Gen}(1^{\lambda})$. Output $\mathsf{sk} = k$.
- SKE.Enc(sk, x): Sample $r \leftarrow \{0,1\}^{\lambda}$. Output $ct = (r, f_k(r) \oplus x || 0^{\lambda})$, where $x || 0^n$ is the concatenation of the message x with n zeros.
- SKE.VerRange(sk, ct): Parse ct = (r, s). Compute x' = f_{sk}(r) ⊕ s. If x' is of the form x||0^λ, accept. Else, if it does not end in n zeros, reject.
- SKE.Dec(sk, ct): Parse ct = (r, s), compute x' = f_{sk}(r) ⊕ s. If x' is of the form x||0^λ, output x. Else, output ⊥.

The work of Lindell and Pinkas [LP09] shows that the above construction in fact satisfies Definition C.2.

Lemma C.4 ([LP09]). Construction C.3 has an efficiently verifiable, elusive range as in Definition C.2.

Now, we show that the construction additionally has a PV proof of correctness.

Lemma C.5. The SKE scheme in Construction C.3 has PV proof of correctness, i.e. there is a proof of the fact that:

for all $\mathsf{sk} \leftarrow \mathsf{SKE.Gen}(1^{\lambda}, 1^n)$, for all $x \in \{0, 1\}^n$ and randomness r, we have that $\mathsf{SKE.Dec}_{\mathsf{sk}}(\mathsf{SKE.Enc}_{\mathsf{sk}}(x; r)) = x$.

Proof (sketch). We outline the PV proof.

- 1. For $a, b \in \{0, 1\}^m$, $a \oplus (a \oplus b) = b$.
- 2. $f_{\mathsf{sk}}(r) \oplus (\mathsf{SKE}.\mathsf{Enc}_{\mathsf{sk}}(x;r)) = f_{\mathsf{sk}}(r) \oplus (f_{\mathsf{sk}}(r) \oplus x || 0^{\lambda}).$
- 3. $f_{\mathsf{sk}}(r) \oplus (f_{\mathsf{sk}}(r) \oplus x || 0^{\lambda}) = x || 0^{\lambda}$ by (1).
- 4. $f_{\mathsf{sk}}(r) \oplus (\mathsf{SKE}.\mathsf{Enc}_{\mathsf{sk}}(x;r)) = x || 0^{\lambda}.$
- 5. SKE.Dec(sk, (SKE.Enc_{sk}(x; r)) = SKE.Dec(sk, (r, $f_{sk}(r) \oplus x || 0^{\lambda})).$

6.
$$f_{\mathsf{sk}}(r) \oplus (f_{\mathsf{sk}} \oplus x || 0^{\lambda}) = x || 0^{\lambda} \to \mathsf{SKE}.\mathsf{Dec}(\mathsf{sk}, (r, f_{\mathsf{sk}}(r) \oplus x || 0^{\lambda})) = x \text{ (by definition of SKE.Dec)}.$$

7. SKE.Dec(sk, (SKE.Enc_{sk}(x; r)) = x.

Now, we recap Yao's construction in Algorithm 12 following the exposition of [LP09]. We modify the algorithm (following a remark under Claim 6 of [LP09]) so that it has an PV proof of correctness, and all changes are highlighted.

Lemma C.6. The construction in Algorithm 12 is a secure garbled circuit construction with a PV proof of correctness:

for all circuits C and inputs x, if
$$(sk, \Gamma) \leftarrow \mathsf{GC.Garble}(1^{\lambda}, C)$$
 and $c \leftarrow \mathsf{GC.Enc}(sk, x)$,
then $y \leftarrow \mathsf{GC.Eval}(\Gamma, c)$ satisfies $y = C(x)$.

Proof sketch. The security of the construction follows from the analysis of [LP09]. The only difference in the algorithm is the check in blue, but as discussed in [LP09], this happens with negligible probability.

Algorithm 12 Yao's garbled circuit, following the exposition of [LP09]. The SKE encryption scheme is one satisfying the properties in Definition C.2.

 $\mathsf{GC.Garble}(1^{\lambda}, C):$

- For every wire w in the circuit, assign two keys: sk_w^0 and sk_w^1 .
- For gate g in the circuit with (orderd) input wires w_L and w_R , and output wire w_{out} :
 - If g is not an output gate, for $b_1, b_2 \in \{0, 1\}$, compute

$$\mathsf{ct}_{b_1,b_2} \gets \mathsf{Enc}_{\mathsf{sk}_{w_1}^{b_1}}(\mathsf{Enc}_{\mathsf{sk}_{w_2}^{b_2}}(\mathsf{sk}_{w_3}^{g(b_1,b_2)})).$$

- If g is an output gate, compute

$$\mathsf{ct}_{b_1,b_2} \gets \mathsf{Enc}_{\mathsf{sk}_{w_1}^{b_1}}(\mathsf{Enc}_{\mathsf{sk}_{w_2}^{b_2}}(g(b_1,b_2))).$$

- For any $(\beta_1, \beta_2) \neq (b_1, b_2)$, if $\mathsf{Dec}_{\mathsf{sk}_{w_2}}^{\beta_2}(\mathsf{Dec}_{\mathsf{sk}_{w_1}}^{\beta_1}(\mathsf{ct}_{b_1, b_2})) \neq \bot$, then abort and output $(\Gamma := C, \mathsf{sk} := \bot)$.
- Let L_g be the list of four ciphertexts as computed above, randomly permuted.
- Set Γ to be the list $\{L_g\}_{g \in C}$.
- Set sk ordered list $\{\mathsf{sk}_w^0, \mathsf{sk}_w^1\}_{w \in I(C)}$, where I(C) is the set of all input wires of C.
- Output (Γ, sk) .

 $\mathsf{GC}.\mathsf{Enc}(\mathsf{sk}, x)$:

- If $\mathsf{sk} = \bot$, output (\bot, x) .
- Else, parse $\mathsf{sk} := \{\mathsf{sk}_i^0, \mathsf{sk}_i^1\}_{i \in I(C)}$. Output the list $\{\mathsf{sk}_i^{x_i}\}_{i \in I(C)}$, where x_i is the input to wire $i \in I(C)$.

 $\mathsf{GC}.\mathsf{Eval}(\Gamma,\mathsf{ct}):$

- If $ct = (\bot, x)$, then output $\Gamma(x)$ and abort (in this case, Γ would have been the circuit itself).
- Parse $sk = {sk_i}_i$. Assign the value sk_i to input wire *i*.
- Parse $\Gamma = \{L_g\}_{g \in C}$.
- Else, we evaluate the circuit as follows. Starting from the bottom layer of the circuit, for each gate g:
 - Suppose the (ordered) input wires w_L and w_R , with corresponding assigned values sk_{w_L} and sk_{w_R} .
 - For each $\mathsf{ct} \in L_g$, compute

 $\mathsf{Dec}_{\mathsf{sk}_{w_{D}}}(\mathsf{Dec}_{\mathsf{sk}_{w_{I}}}(\mathsf{ct}))$

- until a decryption which is not \perp is found. Let s be the decrypted value.
- Assign s as $\mathsf{sk}_{w_{out}}$ for the output wire w_{out} of g.
- Output the value assigned to the output gate.

PV proof of correctness (sketch). This proof follows very closely to that of [LP09]. We sketch the proof in words. We induct on the depth of the circuit to argue that the key assigned to every intermediate wire (not the output wire) w is sk_w^{α} , where α is the correct assignment of w on an honest computation of the garbling. This is clearly true in the base layer, by definition of **GC.Eval**. At level i, suppose a gate g has input wires w_L and w_R with $\mathsf{sk}_{w_L}^{\alpha}$ and $\mathsf{sk}_{w_L}^{\beta}$, where α and β correspond to the correct value of the wire. Then, by the line in blue, the only ciphertext that can be decrypted with these keys is $\mathsf{ct}_{\alpha,\beta}$. Therefore, by construction, the output wire of g is assigned $\mathsf{sk}_w^{g(\alpha,\beta)}$, i.e. the "correct" output wire. If g is the output gate, then similarly, the only ciphertext that decrypts is the "correct" one, which will give $g(\alpha, \beta)$, i.e. the true output value.

Since we are inducting on the height of the circuit, this can be done in PV.

C.2 PV Proof for Two-Outcome Attribute-Based Encryption

First, we recall the definitions of attributed-based encryption [SW05, GPSW06] and two-outcome attribute-based encryption [GKP⁺13].

Syntax. An attribute-based encryption scheme is a tuple of algorithms (Setup, Enc, KeyGen, Dec) that works as follows.

- Setup $(1^{\lambda}, 1^k, 1^d)$: The setup algorithm takes as input the security parameter λ , the length of the attribute k, and the depth of the policy circuit d, and outputs a public key pk and a master secret key msk.
- Enc(pk, x, m): The encryption algorithm takes as input the public key, an attribute $x \in \{0, 1\}^k$, and a message $m \in \{0, 1\}$, and output a ciphertext ct_x associated with the policy x.
- KeyGen(msk, f): The key generation algorithm takes as input the master secret key msk, and a policy function f represented as a circuit of input length n and depth d, and outputs a secret key sk_f.
- $Dec(sk_f, ct_x)$: The decryption algorithm takes as input a secret key sk_f and a ciphertext ct_x , and outputs a decrypted message $m' \in \{0, 1\}$.

We require the following correctness and security.

Correctness. For any message $m \in \{0,1\}$, any attribute $x \in \{0,1\}^k$, and any policy f with f(x) = 1, we have that

 $\Pr[(\mathsf{pk},\mathsf{msk}) \leftarrow \mathsf{Setup}(1^{\lambda},1^k,1^d),\mathsf{ct}_x \leftarrow \mathsf{Enc}(\mathsf{pk},x,m),\mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk},f):\mathsf{Dec}(\mathsf{ct}_x,\mathsf{sk}_f)=m]=1.$

Security. The (adaptive) security requires that, for any adversary that non-uniform PPT adversary \mathcal{A} , we have

$$\Pr\left[\begin{array}{cc} (\mathsf{pk},\mathsf{msk}) \leftarrow \mathsf{Setup}(1^{\lambda},1^{k},1^{d}) \\ (f,\mathsf{st}_{1}) \leftarrow \mathcal{A}(\mathsf{pk}) \\ \mathsf{sk}_{f} \leftarrow \mathsf{KeyGen}(\mathsf{msk},f) \\ (m_{0},m_{1},x,\mathsf{st}_{2}) \leftarrow \mathcal{A}(\mathsf{st}_{1},\mathsf{sk}_{f}) \\ b \leftarrow \{0,1\}, \mathsf{ct}_{x} \leftarrow \mathsf{Enc}(\mathsf{pk},x,m_{b}) \\ b' \leftarrow \mathcal{A}(\mathsf{st}_{2},\mathsf{ct}_{x}) \end{array}\right] \leq 1/2 + \mathsf{negl}(\lambda)$$

Construction. The construction requires the lattice trapdoor and preimage sampling algorithms as an ingredient.

- Setup $(1^{\lambda}, 1^{k}, 1^{d})$: The parameter setup algorithm does the following.
 - Choose $n = n(\lambda)$, $m = m(\lambda)$, and $q = q(\lambda)$, where $m \ge 2n \cdot \log q$.
 - Sample $(\mathbf{A}, \tau) \leftarrow \mathsf{TrapSam}(1^n, 1^m, q)$, and sample $\mathbf{D} \leftarrow \mathbb{Z}_q^{n \times m}, \mathbf{B}_1, \ldots, \mathbf{B}_k \leftarrow \mathbb{Z}_q^{n \times m}$.
 - Output $\mathsf{pk} = (\mathbf{A}, \mathbf{D}, \mathbf{B}_1, \dots, \mathbf{B}_k)$ and $\mathsf{msk} = (\mathsf{pk}, \tau)$.
- Enc(pk, x, m): The encryption algorithm parses $pk = (A, D, B_1, ..., B_k)$, $x = (x_1, ..., x_k)$, and does the following.
 - It samples $\mathbf{s} \leftarrow \mathbb{Z}_q^{1 \times n}$, $\mathbf{e}, \mathbf{e}', \mathbf{e}_1, \dots, \mathbf{e}_k \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}^{m \times 1}$, and compute the ciphertext

 $\mathsf{ct}_x = \left(\mathbf{s} \cdot \mathbf{A} + \mathbf{e}, \mathbf{s} \cdot (\mathbf{B}_1 + x_1 \mathbf{G}) + \mathbf{e}_1, \dots, \mathbf{s} \cdot (\mathbf{B}_k + x_k \mathbf{G}) + \mathbf{e}_k, \mathbf{s} \cdot \mathbf{D} + \mathbf{e}' + m \cdot q/2\right).$

- KeyGen(msk, f): The key generation algorithm parses $msk = (pk, \tau)$ and does the following.
 - View $\mathbf{B}_1, \ldots, \mathbf{B}_k$ as GSW ciphertexts, view \mathbf{A} as the public key, and do almost the same as GSW homomorphically evaluation of f over them, except that for each multiplication gate, we further multiply -1 to the ciphertext.

$$\mathbf{B}_f \leftarrow \mathsf{Eval}(\mathbf{A}, f, \mathbf{B}_1, \dots, \mathbf{B}_k).$$

- Use trapdoor τ to sample a matrix \mathbf{R}_f such that

$$\begin{bmatrix} \mathbf{A} & \mathbf{G} + \mathbf{B}_f \end{bmatrix} \cdot \mathbf{R}_f = \mathbf{D}.$$

Note that the sampling algorithm may fail to output a small normed \mathbf{R}_f . In that case, we discard the sample and pick an arbitrary \mathbf{R}_f with a small norm satisfying the above equation, using the trapdoor τ .

- Output secret key for f as $\mathsf{sk}_f = \mathbf{R}_f$.
- $\operatorname{Dec}(\operatorname{sk}_f, \operatorname{ct}_x)$: The decryption algorithm parses $\operatorname{ct}_x = (\mathbf{a} \in \mathbb{Z}_q^{1 \times m}, \mathbf{b}_1 \in \mathbb{Z}_q^{1 \times m}, \dots, \mathbf{b}_k \in \mathbb{Z}_q^{1 \times m}, \mathbf{d} \in \mathbb{Z}_q^{1 \times m})$, $\operatorname{sk}_f = \mathbf{R}_f$, and computes the following.
 - For any wire w in the circuit, we keep track a matrix \mathbf{B}_w and a vector $\mathbf{b}_w \in \mathbb{Z}_q^m$. For the input wires, their corresponding matrices are $\mathbf{B}_1, \ldots, \mathbf{B}_k$ and the corresponding vectors are $\mathbf{b}_1, \ldots, \mathbf{b}_k$.
 - For any gate with output wire w in the circuit f, let the matrices corresponding to the input wires w_1, w_2 be $\mathbf{B}_1, \mathbf{B}_2$ and vectors be $\mathbf{b}_1, \mathbf{b}_2$, respectively. Depending on whether the gate is an addition gate or multiplication gate, we do the following.

If the gate is an addition gate, then we let $\mathbf{B}_w = \mathbf{B}_1 + \mathbf{B}_2$, and let $\mathbf{b}_w = \mathbf{b}_1 + \mathbf{b}_2$. Otherwise, if the gate is a multiplication gate, we let

$$\mathbf{B}_w = -\mathbf{B}_1 \cdot \mathbf{G}^{-1}[\mathbf{B}_2], \quad \mathbf{b}_w = w_1 \cdot \mathbf{b}_2 - \mathbf{b}_1 \cdot \mathbf{G}^{-1}[\mathbf{B}_2].$$

– Let $\mathbf{b}_{\mathsf{out}}$ be the vector assigned to the output wire by the above procedure. Then we compute

$$\mathbf{d} - \mathbf{b}_{\mathsf{out}} \cdot \mathbf{R}_f,$$

and round it to either 0 or q/2. Namely, we output $m' \in \{0, 1\}$ indicating whether the above value is within q/4 distance to q/2.

Lemma C.7 (PV proof of correctness for ABE). The correctness of the above construction can be formalized in Cook's theory PV. Formally, the following formula

$$\left(\begin{array}{c} (\mathsf{pk},\mathsf{msk}) = \mathsf{Setup}(1^{\lambda},1^k,1^d;r_{\mathsf{Setup}}) \\ \wedge \mathsf{ct}_x = \mathsf{Enc}(\mathsf{pk},x,m;r_{\mathsf{Enc}}) \\ \wedge \mathsf{sk}_f = \mathsf{KeyGen}(\mathsf{msk},f;r_{\mathsf{KeyGen}}) \\ \wedge f(x) = 1 \end{array} \right) \to \mathsf{Dec}(\mathsf{ct}_x,\mathsf{sk}_f) = m.$$

can be proven in PV, where Setup, Enc, KeyGen, Dec are naturally regarded as function symbols in PV by their definition, and r_{Setup} , r_{Enc} , r_{KeyGen} are variables representing the random coins used by the corresponding algorithms.

Proof Sketch. We only give a sketch of the proof here. We will use the variables in PV as defined in the above formula in PV.

• Prove inductively on every gate of f to show that, for every wire w in the circuit, we have

$$\mathbf{b}_w = \mathbf{s} \cdot \begin{bmatrix} \mathbf{A} & \mathbf{B}_w + w \cdot \mathbf{G} \end{bmatrix} + \mathbf{e}_w,$$

where \mathbf{e}_w is a noise vector with a small bounded norm. In particular, for the output wire w_{out} , since f(x) = 1 and $\mathbf{B}_{out} = \mathbf{B}_f$, we have

$$\mathbf{b}_{\mathsf{out}} = \mathbf{s} \cdot \begin{bmatrix} \mathbf{A} & \mathbf{B}_f + \mathbf{G} \end{bmatrix} + \mathbf{e}_{\mathsf{out}}.$$

PV allows a polynomial induction rule so that we can formalize this induction in PV.

• In the decryption algorithm, when we compute $\mathbf{d} - \mathbf{b}_{out} \cdot \mathbf{R}_f$, we have

$$\mathbf{d} - \mathbf{b}_{\mathsf{out}} \cdot \mathbf{R}_f = (\mathbf{s} \cdot \mathbf{D} + \mathbf{e}' + m \cdot q/2) - \mathbf{s} \cdot \begin{bmatrix} \mathbf{A} & \mathbf{B}_f + \mathbf{G} \end{bmatrix} \cdot \mathbf{R}_f + \mathbf{e}_{\mathsf{out}} \cdot \mathbf{R}_f$$
$$= (\mathbf{s} \cdot \mathbf{D} + \mathbf{e}' + m \cdot q/2) - \mathbf{s} \cdot \mathbf{D} + \mathbf{e}_{\mathsf{out}} \cdot \mathbf{R}_f$$
$$= m \cdot q/2 + \mathbf{e}' + \mathbf{e}_{\mathsf{out}} \cdot \mathbf{R}_f$$

The first equality follows from the fact that $\mathbf{d} = \mathbf{s} \cdot \mathbf{D} + \mathbf{e}' + m \cdot q/2$ is part of the definition of the function symbol Dec, and $\mathbf{b}_{out} = \begin{bmatrix} \mathbf{A} & \mathbf{B}_f + \mathbf{G} \end{bmatrix}$ is part of the definition of Enc. So the first equality can be formalized in PV.

The second equality follows from the fact that $\begin{bmatrix} \mathbf{A} & \mathbf{G} + \mathbf{B}_f \end{bmatrix} \cdot \mathbf{R}_f = \mathbf{D}$ can be derived from the definitional formulas of KeyGen. This is true because when the sampling algorithm fails, we arbitrarily pick a small \mathbf{R}_f satisfying this equation.

The third equality follows from the fact that the basic theorems about linear algebra can be formalized in PV.
• Prove that since $\mathbf{e}', \mathbf{e}_{out}$, and \mathbf{R}_f have small norms, the outcome of the decryption is correct. This step only uses the basic properties of matrices and inequalities. Hence it can be formalized in PV.

Concatenate the PV proof for each of the above steps, we obtain a PV proof of correctness for the above construction.

Now, we construct a "Two-outcome" ABE scheme $[GKP^+13]$. First, we recall the syntax.

Syntax. An two-outcome attribute-based encryption scheme is a tuple of algorithms (Setup, Enc, KeyGen, Dec) that works as follows.

- $ABE_2.Setup(1^{\lambda}, 1^k, 1^d)$: The setup algorithm takes as input the security parameter λ , the length of the attribute k, and the depth of the policy circuit d, and outputs a public key pk and a master secret key msk.
- ABE₂.Enc(pk, x, M_0, M_1): The encryption algorithm takes as input the public key, an attribute $x \in \{0, 1\}^k$, and two messages M_0 and M_1 , and outputs a ciphertext c.
- $ABE_2.KeyGen(msk, f)$: The key generation algorithm takes as input the master secret key msk, and a policy function f represented as a circuit of input length n and depth d, and outputs a secret key sk_f .
- ABE₂.Dec(sk_f, ct_x): The decryption algorithm takes as input a secret key sk_f and a ciphertext ct_x, and outputs a decrypted message \mathcal{M}^* .

We require the following properties:

Correctness. For any message $m \in \{0, 1\}$, any attribute $x \in \{0, 1\}^k$, and any policy f,

$$\Pr\left[\begin{array}{cc} (\mathsf{pk},\mathsf{msk}) \leftarrow \mathsf{Setup}(1^{\lambda},1^{k},1^{d}), \\ \mathsf{Dec}(\mathsf{ct}_{x},\mathsf{sk}_{f}) = M_{f(x)} = 1 & : & \mathsf{ct}_{x} \leftarrow \mathsf{Enc}(\mathsf{pk},x,(M_{0},M_{1})), \\ & \mathsf{sk}_{f} \leftarrow \mathsf{KeyGen}(\mathsf{msk},f) \end{array}\right] = 1.$$

The work of $[GKP^+13]$ construct ABE₂ from an ABE scheme, as follows:

- ABE₂.Setup $(1^{\lambda}, 1^k, 1^d)$:
 - $\text{ Sample } (\mathsf{pk}_0,\mathsf{fmsk}_0) \leftarrow \mathsf{ABE}.\mathsf{Setup}(1^\lambda,1^k,1^d) \text{ and } (\mathsf{pk}_1,\mathsf{fmsk}_1) \leftarrow \mathsf{ABE}.\mathsf{Setup}(1^\lambda,1^k,1^d).$
 - Set $\mathsf{fmsk} := (\mathsf{fmsk}_0, \mathsf{fmsk}_1)$ and $\mathsf{pk} = (\mathsf{pk}_0, \mathsf{pk}_1)$.
- $ABE_2.Enc(pk, x, M_0, M_1)$:
 - Let $\mathsf{ct}_b \leftarrow \mathsf{ABE}.\mathsf{Enc}(\mathsf{pk}_b, x, M_b)$.
 - Output (ct_0, ct_1) .
- ABE₂.KeyGen(msk, *f*) :
 - $\mathsf{fsk}_0 \leftarrow \mathsf{ABE}.\mathsf{KeyGen}(\mathsf{fmsk}_0, f).$

- Let \overline{f} be the negation of f.
- − $\mathsf{fsk}_1 \leftarrow \mathsf{ABE}.\mathsf{KeyGen}(\mathsf{fmsk}_1, \overline{f}).$
- Output $\mathsf{sk}_f = (\mathsf{fsk}_0, \mathsf{fsk}_1)$.
- ABE₂.Dec(sk_f, ct, f, x) :
 - Parse $\mathsf{sk}_f = (\mathsf{fsk}_0, \mathsf{fsk}_1)$ and $\mathsf{ct} = (\mathsf{ct}_0, \mathsf{ct}_1)$.
 - If f(x) = 0, compute $M_0 \leftarrow \mathsf{ABE}.\mathsf{Dec}(\mathsf{fsk}_0, \mathsf{ct}_0)$ and output M_0 .
 - If f(x) = 1, compute $M_1 \leftarrow \mathsf{ABE}.\mathsf{Dec}(\mathsf{fsk}_1, \mathsf{ct}_1)$ and if $M_1 \neq \bot$, output M_1 .

We now sketch the proof of correctness assuming that ABE has a PV proof of completeness.

- If f(x) = 0, by correctness of ABE, ABE.Dec(fsk₀, ABE.Enc(pk₀, x, M₀)) = M₀.
- Similarly, if f(x) = 1, then $\overline{f}(x) = 0$. Then, ABE.Dec(fsk₀, ABE.Enc(pk₀, x, M₁)) = M₁.
- By definition,

$$\begin{split} \mathsf{ABE}_2.\mathsf{Dec}(\mathsf{sk}_f,\mathsf{ABE}_2.\mathsf{Enc}(\mathsf{pk},x,M_0,M_1)) &= \mathsf{ABE}.\mathsf{Dec}(\mathsf{fsk}_{f(x)},(\mathsf{fsk},\mathsf{ABE}.\mathsf{Enc}(\mathsf{pk}_{f(x)},x,M_{f(x)})) \\ &= M_{f(x)}. \end{split}$$

This completes the proof.

C.3 Putting it all together

Ingredients. Suppose that there exist:

• Leveled FHE with a proof of correctness, i.e. proof of the fact that:

for all $(sk, pk, ek) \leftarrow FHE.Gen(1^{\lambda}; r)$, for all x, randomness r' and circuits C, we have that FHE.Dec_{sk}(FHE.Eval_{ek}(FHE.Enc_{pk}(x; r'), C)) = C(x).

In Appendix B, we show that this can be instantiated from LWE following the works of [BV11, GSW13].

• Garbled circuits with proof of correctness, i.e. proof of the fact that:

for all circuits C and inputs x, if $(\mathsf{sk}, \Gamma) \leftarrow \mathsf{GC.Garble}(1^{\lambda}, C)$ and $c \leftarrow \mathsf{GC.Enc}(\mathsf{sk}, x)$, then $y \leftarrow \mathsf{GC.Eval}(\Gamma, c)$ satisfies y = C(x).

In Appendix C.1.1, we showed that Yao's construction [Yao82] has this property with some minor modifications.

• Two-outcome ABE for small-depth circuits. In Appendix C.2, we showed that the construction of [GKP+13] from LWE has this property.

Lemma C.8. If instantiated with the above ingredients with proofs of correctness, the construction of sFE (as in Theorem 7.6) has a proof of correctness, i.e.

for all randomness, for all functions f and input x, if $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{sFE.Setup}(1^{\lambda})$, $\mathsf{fsk}_f \leftarrow \mathsf{sFE.KeyGen}(\mathsf{msk}, f), c \leftarrow \mathsf{sFE.Enc}(\mathsf{mpk}, x), then y \leftarrow \mathsf{FE.Dec}(\mathsf{fsk}_f, c) \text{ satisfies } y = f(x)$.

Proof sketch. Upon inspection of the construction of $[GKP^+13, Claim 3.8]$, it is clear that the PV proof of correctness follows directly from the PV proof of garbled circuits, two-outcome ABE and FHE.