2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)

# Indistinguishability Obfuscation via Mathematical Proofs of Equivalence

Abhishek Jain
*Department of Computer Science*
*Johns Hopkins University*
Baltimore, USA
abhishek@cs.jhu.edu

Zhengzhong Jin
*Department of Computer Science*
*Johns Hopkins University*
Baltimore, USA
zjin12@jhu.edu

***Abstract*—Over the last decade, indistinguishability obfuscation (iO) has emerged as a seemingly omnipotent primitive with numerous applications to cryptography and beyond. Moreover, recent breakthrough work has demonstrated that iO can be realized from well-founded assumptions. A thorn to all this remarkable progress is a limitation of all known constructions of general-purpose iO: the security reduction incurs a loss that is *exponential in the input length* of the function. This "input-length barrier" to iO stems from the non-falsifiability of the iO definition and is discussed in folklore as being possibly inherent. It has many negative consequences; notably, constructing iO for programs with inputs of *unbounded* length remains elusive due to this barrier.**

**We present a new framework aimed towards overcoming the input-length barrier. Our approach relies on short *mathematical proofs* of functional equivalence of circuits (and Turing machines) to avoid the brute-force "input-by-input" check employed in prior works.**

- **We show how to obfuscate circuits that have efficient proofs of equivalence in Propositional Logic with a security loss *independent* of input length.**
- **Next, we show how to obfuscate Turing machines with *unbounded* length inputs, whose functional equivalence can be proven in Cook's Theory $PV$.**
- **Finally, we demonstrate applications of our results to succinct non-interactive arguments and witness encryption, and provide guidance on using our techniques for building new applications.**

**To realize our approach, we depart from prior work and develop a new gate-by-gate obfuscation template that preserves the topology of the input circuit.**

***Index Terms*—cryptography, logic**

## I. Introduction

Program obfuscation is the technique of converting a computer program into a new version that retains the functionality of the original but is immune to reverse-engineering. While a formal study of this notion was initiated at the turn of this century [45], [8], the past decade has seen a renewed push towards its study. The notion of indistinguishability obfuscation (iO) [8] has emerged as the central figure, with a long sequence of works aimed towards investigating its existence (see e.g., [33], [64], [43], [6], [14], [53], [58], [56], [55], [7], [54], [57], [1], [48], [4], [20], [2], [41], [69], [42]). This line of work recently led to the breakthrough result of [49] who constructed iO for general functions from well-founded assumptions.

A parallel line of research over the last decade has demonstrated that most cryptographic primitives, including several powerful ones such as witness encryption [35], multiparty non-interactive key exchange [17], succinct non-interactive arguments [67], [10], software watermarking [27], and deniable encryption [67] can be built from iO. Moreover, iO has also found appeal outside cryptography, such as for establishing hardness of Nash equilibrium [13] and the hardness of certain tasks in differential privacy [17], [21]. These results have established iO as a "central hub" of theoretical cryptography.

**Input-Length Barrier.** A thorn to all this remarkable progress is a limitation of all known constructions of iO: the security reduction incurs a loss that is *exponential in the input length* of the function. This has severe negative consequences on the necessary assumptions and the efficiency of the scheme. In particular, it requires the program input length to be a priori bounded. This, in turn, prevents us from realizing iO for efficient computing models such as Turing machines with *unbounded* input length.[1]

This state of affairs motivates the following question:

*Can we build iO with a loss in the security reduction independent of the input length?*

To answer the above question, it is first important to understand whether the input-length barrier stems from technical limitations or something more fundamental. To develop intuition, it is useful to recall a folklore argument that explains the origin of the input-length barrier. Here, we sketch the informal idea[2] (adapted from [35], [59]) based on the meta-reduction technique [15].

Let us first recall the security definition of iO: if two programs $P_1$ and $P_2$ are *functionally equivalent* (i.e., for

The authors were supported in part by NSF CNS-1814919, NSF CAREER 1942789 and Johns Hopkins University Catalyst award. The first author was additionally supported in part by AFOSR Award FA9550-19-1-0200 and the Office of Naval Research Grant N00014-19-1-2294. This work was done while the authors were visiting University of California Berkeley.

[1] Some prior works overcome this barrier by relying on non-standard assumptions; we discuss this later.

[2] We stress that this is *not* a formal proof. Turning this argument into a formal proof runs into subtle technical challenges.

2575-8454/22/$31.00 ©2022 IEEE
DOI 10.1109/FOCS54457.2022.00100

any input $x$, $P_1(x) = P_2(x)$), then their obfuscations must be indistinguishable to any polynomial-time algorithm. Now, suppose that there is a construction of iO whose security can be based on some polynomial-time hardness assumption (say) $Y$. That is, there is a polynomial-time reduction such that given black-box access to an adversary for the iO scheme, it can break the assumption $Y$. Consider the following "trivial" polynomial-time adversary that chooses two programs $P_1, P_2$ that are functionally equivalent except that their outputs differ at some input (say) $x^*$. Such an adversary can easily distinguish between obfuscations of $P_1$ and $P_2$ by evaluating them on $x^*$; yet the reduction must seemingly work for such an adversary as well. Then, combining the reduction with this trivial adversary, we have found a polynomial-time algorithm for $Y$, which is unlikely.

To prevent the above argument, it seems that the reduction must check whether the two programs $P_1, P_2$ are functionally equivalent so as to not be "fooled" by the trivial adversary. But how can the reduction check equivalence? One natural way is to iterate through all the inputs one by one. Indeed, this is the strategy implicit in the security proofs of all general-purpose constructions of iO. This strategy, however, leads to a security loss that is exponential in the input length.

Can we use an alternative strategy that does not incur such a loss? A sequence of prior works [37], [39], [38], [59] demonstrate that the exponential loss can be avoided in some cases when functional equivalence can be decided in *polynomial time* [59]. This naturally limits their applicability (see Section I-D for discussion). Indeed, in general, functional equivalence may not be efficiently checkable. We ask whether it is possible to overcome the input-length barrier in such cases as well.

**A Broader Perspective.** The seeming necessity of checking functional equivalence and its consequences is in fact an example of a broader phenomenon in cryptography. The security definition of many cryptographic primitives is predicated on a mathematical premise that is not decidable in $\mathcal{NP}$. For example, the security of witness encryption [35] for a language $L$ requires that a ciphertext encrypted using an instance $x \notin L$ must remain semantically secure. Similarly, the soundness definition of a proof system for a language $L$ requires that any proof for an instance $x \notin L$ must be rejected by the verifier. In both of these cases, "$x \notin L$" is the mathematical premise, and deciding its truthfulness is a $co\mathcal{NP}$ problem that might require exponential time.

The difficulty of checking the mathematical premise can be leveraged to employ a similar meta-reduction technique as discussed above to establish barriers for other cryptographic primitives. This is reflected in the case of witness encryption, where all known constructions incur a security loss exponential in the witness length. In the regime of proof systems, Gentry and Wichs [44] leverage this observation to rule out adaptively-sound succinct non-interactive arguments [60] based on falsifiable assumptions [61]. Moreover, even known non-adaptively sound constructions (obtained by instantiating [67] with existing iO constructions) incur an exponential loss in the witness length.[3]

**A New Approach.** We present a new framework aimed towards overcoming the input-length barrier to iO. We then leverage the power of iO to overcome analogous barriers for other cryptographic primitives.

Our starting point is the following simple observation: suppose we are given a secure indistinguishability obfuscator. In order to leverage its security for a given pair of programs, we first write a *mathematical proof* to convince ourselves (and others) that the two programs are functionally equivalent. Importantly, this proof is *short* so that anyone can verify it. In particular, it is significantly shorter than the "brute-force" proof that involves iterating over every input. Our key insight is to rely on such (short) mathematical proofs of functional equivalence *for proving the security of the obfuscator*.

This raises the following question: *How can we use the mathematical proof in proving security?* Our approach involves two principal steps:

– **Incremental Proofs of Equivalence:** We first rely on the following *local* property of mathematical proofs: recall that a mathematical proof consists of a series of true propositions, one followed by another. The truthfulness of each proposition is derived from only a constant number of previous propositions and an inference rule. We leverage this property to show that a short mathematical proof (of specific form) of "$C_1(x) = C_2(x)$" for two circuits $C_1$ and $C_2$ can be translated to a small number of *incremental changes* that transform the circuit $C_1$ into $C_2$. Crucially, each incremental change is of small size.
– **New Template for iO:** Next, we provide a new construction template for iO to leverage the above proofs of equivalence. Our template involves obfuscating an input circuit in a gate-by-gate manner to *preserve its topology* in the obfuscated circuit. This allows us to devise a security proof consisting of a polynomial number of steps, where in each step we only switch an obfuscated *subcircuit* corresponding to an incremental change. This results in a security loss exponential only in the size of the subcircuit but *independent* of the input length.

## A. *Our Results*

We now proceed to describe our results.

**I. iO for Circuits.** We first consider the circuit model of computation. Our results rely on proofs in Propositional Logic [23] — a branch of logic that deals with propositions and relations among them.

We define a notion of *propositional proof of equivalence* for circuits. Roughly speaking, we say that two circuit families $\{C_n^1\}_{n\in\mathsf{N}}$ and $\{C_n^2\}_{n\in\mathsf{N}}$ have a propositional proof of equivalence, if there exists a proof in propositional logic system to establish that $C_n^1$ and $C_n^2$ are functionally equivalent. Furthermore, we say that the proof is *efficient* if it is polynomial-sized.

[3]We discuss more on this later in Section I-B.

Our first result is an obfuscation scheme for any two families of circuits with efficient propositional proofs of equivalence, with security loss independent of input length.

**Theorem 1** (iO for Circuits from Propositional Proofs of Equivalence, Informal)**.** *There exist polynomials $p_1(\cdot), p_2(\cdot,\cdot,\cdot)$, such that assuming the hardness of the following, there exists a construction of* iO *for any two families of circuits $\{C_n^1\}_{n\in\mathsf{N}}, \{C_n^2\}_{n\in\mathsf{N}}$ with* efficient *propositional proofs of equivalence:*

- *Polynomial-hardness of Learning with Errors (LWE),*
- *$2^{p_1(\lambda)}$-secure one-way functions,*
- *$2^{p_1(\lambda)}$-secure indistinguishability obfuscation for circuits of size $p_2(\lambda, \log|C_n^1|, \log|C_n^2|)$,*

*where $\lambda$ is the security parameter of the* iO *scheme.*

A few remarks are in order:

- Unlike prior works, we allow $n$, namely, the input length of circuits $C_n^1, C_n^2$ (and their sizes) to arbitrarily depend on $\lambda$, and not be bounded by $p_1, p_2$.
- The above theorem only requires an underlying indistinguishability obfuscator for *small* circuits of size essentially independent of $C_n^1, C_n^2$.

We obtain the above result in two steps: we first define a new notion of $\Delta$-equivalent circuits and show how $\Delta$-equivalent circuits can be constructed via Proofs in Propositional Logic [23]. We then show how to construct iO for $\Delta$-equivalent circuits, with security loss independent of input length.

*Step I: $\Delta$-Equivalent Circuits.* Informally, we say that two circuit families are *$\Delta$-equivalent*, if there exist a polynomial number of intermediate circuits such that each two adjacent circuits only differ by a *logarithmic* number of gates, and the two subcircuits formed by these gates are functionally equivalent.

We demonstrate that efficient propositional proof of equivalence implies $\Delta$-equivalence for circuits.

**Lemma 1** ($\Delta$-Equivalence from Propositional Logic Proofs)**.** *If there exist* polynomial-size *propositional proofs of equivalence for the circuit families $\{C_n^1\}_{n\in\mathsf{N}}$ and $\{C_n^2\}_{n\in\mathsf{N}}$, then $\{C_n^1\}_{n\in\mathsf{N}}$ and $\{C_n^2\}_{n\in\mathsf{N}}$ are $\Delta$-equivalent.*

Given a pair of circuits $(C_n^1, C_n^2)$ and a propositional proof of equivalence, we prove this lemma by embedding the propositional formulas (in the proof of equivalence) inside $C_n^1$ to gradually transform it into $C_n^2$, while preserving the functionality. We leverage the "local" property of the proof as well as the truthfulness of each formula to establish $\Delta$-equivalence. See Section II-A for an overview of the proof.

*Step II: iO for $\Delta$-Equivalent Circuits.* We next provide a construction of iO for $\Delta$-equivalent circuits.

**Lemma 2** (iO for $\Delta$-Equivalent Circuits, Informal)**.** *There exist polynomials $p_1(\cdot), p_2(\cdot,\cdot,\cdot)$, such that assuming the same hardness assumptions as in Theorem 1, there exists a construction of* iO *for any two $\Delta$-equivalent circuit families $\{C_n^1\}_{n\in\mathsf{N}}, \{C_n^2\}_{n\in\mathsf{N}}$.*

In order to prove the above lemma, we depart from prior templates for iO [6], [14]. To leverage $\Delta$-equivalence, we develop a new (albeit, natural) *gate-by-gate* obfuscation template that preserves the topology of the input circuit. Due to such a design, a key challenge is to overcome various "mix-and-match" attacks, and we develop several techniques towards that end. A central component in our construction is a new notion of *somewhere extractable hash functions with consistency proofs*. We show how to build this object by combining somewhere extractable hash functions [46] with (publicly-verifiable) non-interactive batch arguments [25]. Both of these objects, in turn, can be based on the LWE assumption. We refer the reader to Section II for an overview of our technical approach.

**II. iO for Turing Machines.** We next tackle the challenging problem of constructing iO for Turing machines with unbounded length inputs. All prior results can either handle inputs of a priori bounded length [12], [24], [51], or require very strong assumptions [18], [3], [47], [56] (some of which are in fact known to be implausible in general [11], [34], [9], [56]).

We show how to obfuscate Turing machines with arbitrary length inputs based on similar assumptions as used for obfuscating circuits. Our approach is applicable to Turing machines whose functional equivalence can be proven in Cook's theory $PV$ [30]. Cook introduced the theory $PV$ in 1975 to formalize the intuition of polynomial-time reasoning. $PV$ is a fundamental theory in the area of proof complexity [62], [30], [22], and is useful for translating theorems to propositional logic proofs.

We say that two Turing machines $M_1$ and $M_2$ have a *$PV$-proof of equivalence* if the functional equivalence of $M_1$ and $M_2$ is provable in $PV$. We prove the following result:

**Theorem 2** (iO for Turing Machines, Informal)**.** *Assuming quasi-polynomial hardness of Learning with Errors, sub-exponentially secure one-way functions, and sub-exponentially secure indistinguishable obfuscation for circuits, there exists a construction of* iO *for Turing machines with unbounded-length inputs and $PV$-proofs of equivalence.*

**On the use of Sub-exponential Assumptions.** Although we rely on the sub-exponential security of the underlying primitives in our results, the hardness requirement for the underlying primitives is independent of the input length of the input circuits.

To our understanding, there is no obvious barrier to avoiding these sub-exponential assumptions due to the following observation: given a series of intermediate circuits, verifying $\Delta$-equivalence only takes polynomial time, since checking whether two subcircuits of size $O(\log n)$ are functionally equivalent or not only takes $2^{O(\log n)} = \mathsf{poly}(n)$ time. Hence, constructing $\Delta$iO for $\Delta$-equivalent circuits from polynomial hardness is not ruled out by the input-length barrier. We therefore view our use of sub-exponential assumptions as a technical limitation that we can hope to overcome in the future.

## B. Applications

We now discuss applications of our results towards building witness encryption and succinct non-interactive arguments (SNARGs) with properties that were not known to be achievable earlier. Our results for these primitives apply for a subclass of $\mathcal{NP} \cap co\mathcal{NP}$ languages whose disjointness with its complement can be proven in some logic system.

We start by characterizing this class of languages.

**Mathematical Proof of Disjointness.** Intuitively, we say a language $L \in \mathcal{NP} \cap co\mathcal{NP}$ has proof of disjointness, if "$L \cap \overline{L} = \phi$" can be proven in some mathematical logic system, where $\overline{L} = \{0,1\}^* \setminus L$ is the complement of $L$ and both $L, \overline{L}$ are represented by circuits or Turing machines.

Specifically, let $\{M_n\}_{n\in\mathsf{N}}$ and $\{\overline{M}_n\}_{n\in\mathsf{N}}$ be the circuit families that define the $\mathcal{NP}$-relation of $L$ and $\overline{L}$ respectively. We say that $L$ has propositional proof of disjointness, if "$\overline{M}_n(x, \overline{w}) = 1 \to M_n(x, w) \neq 1$" has polynomial-size proofs in the extended Frege system. This intuitively requires that the statement

*"For any $x$, if $\overline{M}_n(x, \cdot)$ is satisfiable, then $M_n(x, \cdot)$ is not."*

can be proven in propositional logic sytem. Similarly, let $M, \overline{M}$ be the Turing machines that defines $L, \overline{L}$ respectively. We say $L$ has $PV$ proof of disjointness, if $\overline{M}(x, \overline{w}) = 1 \to M(x, w) \neq 1$ can be proven in Cook's theory $PV$. Since propositional translation [30] can translate a $PV$ proof to polynomial-size propositional proofs, $PV$ proof of disjointness implies propositional proof of disjointness.

What languages have proofs of disjointness? We expect that for most $\mathcal{NP} \cap co\mathcal{NP}$ languages that we are interested in, we can write a mathematical proof of disjointness. Indeed, otherwise it is hard to convince ourselves that the language is in $\mathcal{NP} \cap co\mathcal{NP}$. We give a concrete example below, namely, the language $\mathsf{TAUT}$ in computational complexity. Furthermore, we will show in Section I-C that for cryptographic applications, a large part of such mathematical proofs can be formalized in theory $PV$.

*Example.* $\mathsf{TAUT}$ is the language that contains all tautologies. Recall that a tautology is a formula that always evaluates to true for any truth assignment. $\mathsf{TAUT}$ is known to be $co\mathcal{NP}$-complete and hence is an important language in complexity theory.

By the completeness theorem of propositional logic [23], any tautology has a proof in propositional logic. However, such a proof may not have a polynomial size. Hence, to ensure the honest prover/encryptor runs in the polynomial-time in the setting of SNARGs/WE, we consider a slight variant of $\mathsf{TAUT}$, which is the following promise language $L_{\mathsf{TAUT}} = (L_{\mathsf{YES}}, L_{\mathsf{NO}})$. $L_{\mathsf{YES}}$ contains all tautologies with a polynomial-bounded propositional logic proof, whereas $L_{\mathsf{NO}}$ contains all non-tautologies. Then $PV$ proof of disjointness can be extended naturally to promise languages: we require "$L_{\mathsf{YES}} \cap L_{\mathsf{NO}} = \phi$" can be proven in theory $PV$. Cook [30] showed that the soundness of propositional logic system is provable in $PV$, which implies that $L_{\mathsf{TAUT}}$ has $PV$ proof of disjointness.

We now proceed to discuss applications to witness encryption and SNARGs.

**I. Witness Encryption.** A witness encryption (WE) scheme allows an encryptor to use an instance $x$ from a language $L$ to encrypt a message $m$ such that anyone who knows a witness $w$ for $x$ can retrieve the message $m$. Security requires that if $x \notin L$, then the ciphertext hides $m$. As discussed earlier, all prior constructions of WE only support bounded witness lengths due to the input-length barrier.

As a generic application of Theorem 1, we build a WE scheme for any language $L \in \mathcal{NP} \cap co\mathcal{NP}$ with propositional proof of disjointness, with security loss independent of the witness length. Furthermore, as an application of Theorem 2, we build a WE scheme for Turing machines for any language $L$ in $\mathcal{NP} \cap co\mathcal{NP}$ with $PV$ proof of disjointness. The latter scheme can support witnesses of *unbounded length*. The ciphertext size is independent of the witness length, but grows with the running time of the Turing machine $\overline{M}$.

**II. Succinct Non-Interactive Arguments.** A non-interactive argument system for an $\mathcal{NP}$ language $L$ is said to be *succinct* if the proof size is much smaller than the witness size. Gentry and Wichs (GW) [44] proved that such argument systems cannot be constructed with a black-box proof of *adaptive*[4] soundness to falsifiable assumptions. On the other hand, a *non-adaptively* sound construction based on $\mathsf{iO}$ was given by Sahai and Waters (SW) [67].

While $\mathsf{iO}$ is not a falsifiable assumption, one can instantiate the SW construction with a recent $\mathsf{iO}$ scheme (such as [49]) to obtain a scheme based on falsifiable assumptions. This resulting scheme, however, incurs a security loss exponential in the witness length due to the input-length barrier to $\mathsf{iO}$. This has two consequences: first, this means that the scheme bypasses the GW lower bound due to the fact that the security reduction is able to decide the language.[5] Second, the scheme can only handle witnesses of a priori bounded length, and in particular, the size of the common reference string (which contains the obfuscation) grows with the size of the witness.

We show how to overcome these limitations by constructing SNARGs that can support witnesses of *unbounded length* for any language $L \in \mathcal{NP} \cap co\mathcal{NP}$ with $PV$ proof of disjointness. The CRS size is independent of the witness length and only depends on the running time of the Turing machine $\overline{M}$ that defines $\overline{L}$. Our base scheme is non-adaptively sound, but by standard complexity leveraging over the instances, it can also achieve adaptive soundness.

An important step towards this obtaining this result is to build puncturable pseudorandom functions (PRFs) [16], [19], [50] with a $PV$-proof of functionality preservation. We show

[4]Adaptive (resp., non-adaptive) soundness refers to the setting where the adversary can choose the challenge instance after (resp., before) viewing the common reference string.

[5]Indeed, this scheme can also achieve adaptive security by standard complexity leveraging (over the instances) without further security degradation.

that puncturable PRFs based on the GGM PRFs [16], [19], [50], [67] satisfy this property (see Section I-C for further discussion).

### C. How to Use iO with Proofs of Equivalence

We provide some general guidance for building new applications using our results. We consider some tools that are commonly used within iO-based applications and demonstrate how one can formalize properties about such tools in propositional logic or theory $PV$. Such proofs can then be used to build proofs of equivalence of circuits or Turing machines involved in the desired application.

In Section I-C1, we consider puncturable PRFs that are used ubiquitously in constructions involving iO [67]. Specifically, we show that the functionality preservation property of GGM PRFs [16], [19], [50], [67] can be proven in theory PV. Next, in Section I-C2, we provide general guidance on proving properties of tools in group-based cryptography and lattice-based cryptography. As concrete examples, we demonstrate that the correctness of ElGamal encryption [32] and Regev's encryption [66] can be proven in theory $PV$ .

*1) Puncturable PRFs:* A *puncturable* PRF [16], [19], [50], [67] $\mathsf{PRF}_{punc}$ is a pseudorandom function with the additional property that allows one to puncture the PRF key $k$ at any point $x^*$ to obtain a punctured key $k \setminus \{x^*\}$. For each $x \neq x^*$, the functionality preservation property guarantees that $\mathsf{PRF}(k, x) = \mathsf{PRF}_{punc}(k \setminus \{x^*\}, x)$.

iO-based constructions that involve the use of puncturable PRFs require the functionality preservation property to establish the functional equivalence of the two programs being obfuscated. Since our constructions require proofs of equivalence in theory $PV$, this translates to requiring that the functionality preservation property of $\mathsf{PRF}_{punc}$ can be proven in theory $PV$. Formally, we say that a puncturable PRF $\mathsf{PRF}_{punc}$ has a $PV$ proof of functionality preservation if the algorithms $\mathsf{PRF}_{punc}$, $\mathsf{PRF}$ and the puncturing algorithm can be defined in $PV$ as function symbols and there exists a proof in $PV$ for $x \neq x^* \rightarrow \mathsf{PRF}(k, x) = \mathsf{PRF}_{punc}(k \setminus \{x^*\}, x)$.

We observe that the GGM-based construction of puncturable PRFs has a $PV$ proof of functionality preservation. We emphasize that we do not need to modify the GGM construction nor its natural mathematical proof of functionality preservation. All we need to do is formalize the existing mathematical proof of functionality preservation in theory $PV$. It is important to note that theory $PV$ does not allow general proof-by-induction rules. Instead, it only allows the following "polynomial-time induction" rule.

$$\text{If } \Phi(0) \text{ holds and } \Phi(x) \rightarrow (\Phi(2x) \wedge \Phi(2x+1)) \\ \text{holds for every } x, \text{ then } \Phi(x) \text{ holds for all } x,$$

where $\Phi(x)$ is a formula in $PV$.

Fortunately, the binary tree structure of the GGM construction is naturally compatible with the polynomial-time induction rule. Hence, the functionality preservation property can be naturally formalized in $PV$.

*2) Proving Arithmetic Properties in $PV$:* In addition to puncturable PRFs, iO-based applications often involve the use of cryptographic primitives such as commitment schemes and encryption schemes. In such cases, key properties of these primitives such as perfect binding or correctness of decryption are essential for establishing the functional equivalence of the programs being obfuscated. We now discuss how such properties can be proven in theory $PV$ when the cryptographic primitives are instantiated using group-based cryptography and lattice-based cryptography.

The general principle involves the following two steps:

- First, write a mathematical proof of such property in natural language.
- Second, examine the basic theorems and axioms used in the mathematical proof to ensure that they can be formalized in theory $PV$.

For illustration purposes, we demonstrate how to prove correctness of group-based and lattice-based public key encryption schemes in theory $PV$.

**Instantiation from Groups.** As an example in group-based cryptography, we show how to prove the correctness of ElGamal encryption [32] in theory $PV$.

Recall that the public key of ElGamal encryption is of the form $(g, g^s)$ where $s$ is the secret key, and $g \in \mathsf{G}$ is a group element. To encrypt a message $m \in \mathsf{G}$ with random coins $r$ under the public key, the ciphertext is $(g^r, (g^s)^r \cdot m)$.

Following the general principle described above, we can prove the correctness in $PV$ as follows:

- We first write down the mathematical proof of correctness of ElGamal in natural language, as follows. If $(c_1, c_2)$ is the ciphertext, then $c_1 = g^r, c_2 = (g^s)^r \cdot m$. Hence, the decryption algorithm Dec computes
$$\begin{aligned} \mathsf{Dec}((c_1, c_2), s) &= c_2/{c_1}^s = (g^s)^r \cdot m/(g^r)^s \\ &= (g^s)^r \cdot m/(g^s)^r = m \cdot ((g^s)^r \cdot /(g^s)^r) = m. \end{aligned}$$
- **Formalization in $PV$:** The above mathematical proof only relies on some basic theorems in arithmetic such as commutative law and associative law of modular multiplication and $(g^s)^r = (g^r)^s$. All such basic theorems can be formalized and proven in $PV$ [30], [22]. Therefore, the above mathematical proof can be formalized in $PV$.

**Instantiation from Lattices.** Using the above ideas, one can also prove the correctness of Regev's public key encryption scheme [66] in $PV$. The main point is that the proof of correctness in natural language only uses some basic arithmetic theorems such as commutative law, distributive law, and some basic properties about inequalities to reason about rounding operations. By Buss's work [22], all such theorems can be proven in $PV$.

### D. Discussion and Future Directions

**On Propositional Logic and Theory $PV$.** Since proofs in propositional logic are central to our results, it is important to understand their expressiveness. If one does not care about the *proof length*, propositional logic is quite expressive due to the completeness theorem [23] which says that any semantically true formula[6] in propositional logic has a proof. Furthermore, we expect that most theorems proven in mathematical logic systems other than propositional logic (e.g. Peano Arithmetic) can also be represented in propositional logic if we set a bound on the number of digits in the natural numbers, and use truth variables in propositional logic to represent the digits of natural numbers.

Propositional logic is expressive enough for proving the equivalence of two Turing machines: for any two functionality equivalent polynomial-time Turing machines, we can set an upper bound on the input length that is super-polynomial in the security parameter. Then, by the completeness theorem of propositional logic, there always exists a propositional logic proof of equivalence for the two Turing machines under the given input bound. However, there is no guarantee that such proofs in propositional logic have *polynomial size*.

Our results crucially require the proof size to be a polynomial. Thus, it is important to understand what can be proven with *polynomial-size propositional proofs*. This question has been extensively studied in proof complexity. In [30], Cook introduced a theory $PV$ to formalize the intuition of "polynomial-time reasoning" and showed that any proof in $PV$ can be translated to a polynomial-size propositional logic proof. Later, a series of works [63], [22], [52] proposed other propositional translations. In this work, we use $PV$ since it is conceptually the simplest. $PV$ allows the definition of new function symbols using Cobham's characterization of polynomial time functions [26]. Basic arithmetic operations can be introduced in this way, and their related properties can be proved in $PV$.

On the positive side, Cook [30] suggested that a good part of elementary number theory can be formalized in $PV$ if the theorems are stated carefully. In the region of linear algebra, [68] showed that the Cayley–Hamilton theorem, basic properties of determinants, and basic matrix properties can be proven in $PV$. For theorems in complexity, it is known that the Cook-Levin theorem and PCP theorem can be formalized and proven in $PV$ [29], [65]. Indeed, Cook observed that the correctness of "natural" polynomial-time algorithms usually can be proven in $PV$ [28]. In this work, we show that a large part of the cryptographic algorithms fall in this category. They include functionality preservation of puncturable PRFs and the correctness of ElGamal Encryption [32] and Regev's encryption [66] (See Section I-C).

On the negative side, it is known that Fermat's little theorem is unlikely to be provable in $PV$ unless factoring can be solved in polynomial time, due to the witnessing theorem [22]. Because of the same reason, the correctness of any polynomial-time algorithm that decides primes is unlikely to be proven in $PV$. Moreover, assuming $\mathcal{NP} \neq co\mathcal{NP}$, there are tautologies that can not be proven with polynomial-size proofs in propositional logic, because $\mathsf{TAUT}$ is $co\mathcal{NP}$-complete [31].

[6]A propositional formula is semantically true if it always evaluates to true under any truth assignments.

**Beyond Theory $PV$.** As discussed earlier, our approach relies on polynomial-size proofs in propositional logic. To increase the scope of our approach, a future direction is to either handle *super-polynomial size* propositional proofs, or use more expressive logic systems. In the former direction, the main challenge is that in our present approach, the sizes of the intermediate circuits grows with the size of the propositional proofs, and thus the obfuscated program will be super-poly size if we naively rely on super-polynomial size propositional proofs. We hypothesize that a potential solution could be to restrict the logic system to "bounded-space reasoning" theories, and finding a more clever way to build the intermediate circuits from propositional logic proofs.

An alternate future direction is to generalize our idea to leverage the "local" property of proofs in more powerful logic systems such as Buss's theories $S_2^i, T_2^i$ [22] since more theorems can be proven in them. Ultimately, one might ask if we can build iO for programs whose equivalence is provable in Zermelo–Fraenkel set theory with the axiom of choice (ZFC). Since ZFC is the most common foundation of mathematics, such a result might be sufficient for most applications of iO. The main seeming difficulty towards this goal is that our current method crucially relies on the property that each line of the propositional proof is also a *circuit*, whereas a line in ZFC is naturally a *Turing machine* that evaluates the truthfulness of that line. Hence, an interesting future direction is to extend our "gate-by-gate" framework to "Turing-machine-by-Turing-machine" framework to support ZFC.

**Towards $\mathcal{NP} \cap co\mathcal{NP}$.** Our method of leveraging mathematical proofs limits us to the circuits whose equivalence can be *verified* in polynomial time. Since circuit equivalence is trivially in $co\mathcal{NP}$, ideally, we could hope to bypass the input-length barrier for the language of circuit pairs in $\mathcal{NP} \cap co\mathcal{NP}$ [59].

Our work makes an important attempt in this direction. We note, however, that not all pairs of circuits whose functionally equivalence is in $\mathcal{NP} \cap co\mathcal{NP}$ necessarily have short mathematical proofs. Therefore, fully realizing the above vision is an important goal for future work.

**Comparison with Decomposable $\mathsf{iO}$.** Liu and Zhandry [59] introduced the notion of decomposable $\mathsf{iO}$ to unify prior works [37], [39], [38] that attempt to avoid the use of sub-exponential hardness assumptions in some specific applications of $\mathsf{iO}$. In the same work [59], Liu and Zhandry proved that deciding whether two circuits are "decomposing-equivalent" is in $\mathcal{P}$. This naturally limits the applicability of their framework. For example, it cannot support the Sahai-Waters construction of public-key encryption from $\mathsf{iO}$ and pseudorandom generators [67]. This is because the security of the pseudorandom generator implies that the two circuits of consideration in the security proof *cannot* be "decomposing-equivalent" since the latter is in $\mathcal{P}$. Indeed, a similar issue arises in many other applications and for this reason, decomposable iO is only applicable when it is easy to check equivalence. (See Section 1.5 in [59] for more discussion.)

Our work does not suffer from this limitation since we do *not* require circuit equivalence to be decidable in $\mathcal{P}$. Instead, we only require the *existence* of a *witness* that allows us to verify the equivalence of two circuits, where the witness is a polynomial-size propositional logic proof. In general, deciding whether the equivalence of two circuits has a short propositional logic proof is not known to be in $\mathcal{P}$.

**On Our Gate-by-Gate Template for iO.** In this work, we develop a new "gate-by-gate" template for building iO for general circuits from iO for "small" circuits. In our template, the topology of the input circuit is preserved in the obfuscated circuit.

While this approach is crucial towards obtaining our results, we observe that it also yields some additional features that might be beneficial in specific use cases. Suppose after distributing an obfuscated circuit, one wishes to modify some gates in the underlying circuit [5], [36]. Instead of obfuscating the modified circuit from scratch (which might be costly), our "gate-by-gate" template allows for easy replacement of the relevant gates in the obfuscated circuit. We defer a formal treatment of this property to future work.

## II. Overview of our Results

We now provide an overview of our results. In Section II-A, we discuss how to establish $\Delta$-Equivalence starting from propositional proofs of equivalence of two circuits. In Section II-B, we describe our construction of iO for $\Delta$-equivalent circuits. We defer the technical overview of iO for unbounded input length Turing machines with $PV$ proofs of equivalence to the full version of the paper.

### A. $\Delta$-Equivalence from Propositional Proofs

We show that given two circuits $C_1, C_2$, if the proposition "$C_1(x) = C_2(x)$" can be proven in a propositional logic system with *extension axioms* such as *extended Frege system* ($\mathcal{EF}$), then $C_1, C_2$ are $\Delta$-equivalent up to some padding. That is, we can find a series of intermediate circuits $C'_1, C'_2, \ldots, C'_\ell$ with the same topology such that every two adjacent circuits $C'_i, C'_{i+1}$ only differ in a *logarithmic* number of gates, and the subcircuits formed by these gates in $C'_i, C'_{i+1}$ are functionality equivalent. Furthermore, the initial circuit $C'_1$ and the final circuit $C'_\ell$ are obtained by padding $C_1, C_2$, respectively, with some dummy gates.

**Background.** We first recall the definition of propositional proof systems with extension axioms. Such logic systems can be described as a set of variables and connectives including "$\to$", "$\leftrightarrow$", "$\wedge$", "$\vee$", and "$\neg$", which refers to "imply", "equal", "and", "or", and "negation", respectively. A *proof* in the propositional proof system is a series of propositional formulas, where each formula is derived from either of the following cases.

- **Axiom:** The formula is in one of the following forms: $P \to (Q \to P)$, $(P \to (Q \to R)) \to ((P \to Q) \to (P \to R))$, or $\neg\neg P \to P$ where $P, Q, R$ are formulas.
- **Modus Ponens:** The formula is in the form $Q$, and there are two previous formulas $P, P \to Q$ derived before the current formula.
- **Extension:** The formula is in the form $e \leftrightarrow Q$, where $e$ is a new variable that does not appear in $Q$ and all previous formulas. This rule is used to introduce intermediate variables and hence can shorten the proof size.

For any circuit, we can treat each of its wires as a variable in propositional logic, whose truth value represents the wire value. Then the mathematical statement "$C_1(x) = C_2(x)$" can be formalized in $\mathcal{EF}$ as a formula.

Assuming there exists a proof $\pi = (\theta_1, \theta_2, \ldots, \theta_k)$ in propositional logic for $C_1(x) = C_2(x)$, we now prove that $C_1, C_2$ are $\Delta$-equivalent. Equivalently, we only need to show that we can transform from $C_1$ to $C_2$ via a series of incremental changes, where each change replaces a logarithmic size subcircuit with a functionally equivalent new subcircuit. To illustrate our high-level ideas, we firstly ignore the topology of the circuits, and hence we can add gates and delete gates arbitrarily. Since we can always treat extension rules as introducing a new wire in the circuit, we also assume there are no extension rules for simplicity.

Our transformation is based on the following key observations.

- The proof $\pi$ is "local", i.e., the truthfulness of each $\theta_i$ follows from a constant number of previous formulas in $\theta_1, \ldots, \theta_{i-1}$.
- The propositional formulas $\theta_1, \theta_2, \ldots, \theta_k$ can also be regarded as boolean circuits, since the connectives including "$\to$" can be expressed as the combination of $\wedge, \vee$, and $\neg$ gates.

**A Sketch of the Transformation.** Based on these observations, our transformation from $C_1$ to $C_2$ proceeds in the following phases. We start with a circuit $C(x)$ that is the same as $C_1(x)$. After the following incremental changes to $C$, $C(x)$ will become $C_2(x)$.

- **Grow $C_2$.** We add the circuit $C_2(x)$ to $C$ in a gate-by-gate manner. Specifically, we add each gate of $C_2$ in the topological order to $C$, while the output wire of $C$ is still set to be the output wire of $C_1(x)$.
  We only change the circuit $C$ for a constant number of gates when we add a gate, since we can always assume such a gate has a constant arity without loss of generality.
- **Grow the Proof.** We add the formulas $\theta_1, \theta_2, \ldots \theta_k$ in the proof $\pi$ one by one to $C$ as follows. Note that each formula $\theta_i$ can be regarded as a circuit that computes the truth values of $\theta_i$ from its variables.
  Firstly, we add $\theta_1$ to $C$ by modifying the output of $C$ as $C_1(x) \wedge \theta_1$. Similarly, to add $\theta_2$, we further modify the output of $C$ to be $C_1(x) \wedge \theta_1 \wedge \theta_2$. We continue this process until all $\theta_1, \theta_2, \ldots, \theta_k$ are added. Then the output of $C$ becomes $C_1(x) \wedge \theta_1 \wedge \theta_2 \wedge \ldots \wedge \theta_k$.

We now show that we only change a small subcircuit in each step of the above process. There are three cases for each $i$, depending on how $\theta_i$ is derived.

- **Axiom:** In this case $\theta_i$ is one of the axioms, for example, $\theta_i$ is in the form $P \to (Q \to P)$. We can assume without loss of generality that $P, Q$ are constant size formulas, as we can always reduce the size of $P, Q$ by assigning their subformulas to new variables using the extension rule.
  In this case the change from $C_1(x) \wedge \theta_1 \wedge \ldots \wedge \theta_{i-1}$ to $C_1(x) \wedge \theta_1 \wedge \ldots \wedge \theta_{i-1} \wedge \theta_i$ can be regarded as replacing a subcircuit that always outputs 1 with a new subcircuit $\theta_i$. The functionality equivalence between the two subcircuits follows from the fact that axioms must be tautologies.
- **Modus Ponens:** For this case, there exists some $P, Q$ such that $P, P \to Q$ are the formulas derived in the first $(i-1)$ formulas, and the current formula $\theta_i$ is $Q$. Similar to the case of axioms, we can assume $P, Q$ are constant-size formulas.
  In this case the change from $C_1(x) \wedge \ldots \wedge P \wedge \ldots \wedge (P \to Q) \wedge \ldots \wedge \theta_{i-1}$ to $C_1(x) \wedge \ldots \wedge P \wedge \ldots \wedge (P \to Q) \wedge \ldots \wedge \theta_{i-1} \wedge Q$ can be regarded as replacing a subcircuit $P \wedge (P \to Q)$ with a new subcircuit $P \wedge (P \to Q) \wedge Q$. The functionality equivalence can be proved by enumerating all possible truth assignment to $P$ and $Q$.

– **Change the Output.** Let $o_1, o_2$ be the output wires of $C_1, C_2$ respectively. Then a proof of "$C_1(x) = C_2(x)$" ends with $o_1 \leftrightarrow o_2$. Namely, $\theta_k$ is the formula $o_1 \leftrightarrow o_2$. Hence, we can replace the output of $C$, which is $o_1 \wedge \theta_1 \wedge \ldots \wedge \theta_k$, with $o_2 \wedge \theta_1 \wedge \ldots \wedge \theta_k$. This step is an incremental change, since it can be regarded as replacing the subcircuit $o_1 \wedge (o_1 \leftrightarrow o_2)$ with $o_2 \wedge (o_1 \leftrightarrow o_2)$.

– **Shrink the Proof.** This phase deletes $\theta_1, \theta_2, \ldots, \theta_k$ in the circuit $C$. Specifically, we remove $\theta_k, \theta_{k-1}, \ldots \theta_1$ one by one in the reversing order that they are added.
This process is a series of incremental changes for the same reason as the "Grow the Proof" phase.

– **Shrink $C_1$.** At the beginning of this phase, the circuit $C$ outputs $o_2$, which is the output wire of $C_2(x)$. The circuit $C_1$ is still in $C$, but its output wire $o_1$ is not used anywhere. Then we delete the gates of $C_1$ in $C$ one by one in the reverse topological order. Finally, we obtain the circuit $C = C_2$.
Deleting a gate of $C_1$ in this phase is an incremental change for the same reason as the "Grow $C_2$" phase.

The reader may already notice that the above sketch over-simplifies many details. For example, the output of the circuit $C$ is computed as a series of $\wedge$-gates, i.e. $C(x) = o_1 \wedge \theta_1 \wedge \theta_2 \ldots$ in the "Grow the Proof" phase, and we argue that we change the subcircuit $P \wedge (P \to Q)$ to $P \wedge (P \to Q) \wedge Q$. However, in the reality, we need to use the arity-2 $\wedge$-gates to implement the series of $\wedge$-gates in $C(x)$. Then $P \wedge (P \to Q)$ and $P \wedge (P \to Q) \wedge Q$ may not be subcircuits, since the positions of $P, Q$ may not be consecutive in the circuit.

**Building An AND Tree.** We resolve this issue by implementing the series of $\wedge$-gates as a binary tree of $\wedge$-gates. Initially, on every leaves there is a gate that always outputs 1. Then in the "Grow the Proof" phase, we replace the leaves with $\theta_i$'s one by one. Now, for each $\theta_i = Q$ obtained from modus ponens, the subcircuit consists of the root-to-leaf paths of $P, P \to Q$ and $\theta_i$. This subcircuit contains only $O(\log k)$ gates, which is logarithmic.

**Handling Extension Rules.** Another issue is how to handle the extension rules. Indeed, there is an additional phase "Grow the Extension" between the "Grow $C_2$" phase and the "Grow the Proof" phase, where we handle all the extensions by introducing new wires in the circuit. Specifically, for any extension of the form $e \leftrightarrow Q$, we add a new wire $e$ and set it as the output wire of a circuit that computes $Q$. Here we can also assume $Q$ is only constant size for the same reason as the "Grow the Proof" phase. Also, between the "Shrink the Proof" phase and the "Shrink $C_1$" phase, we add a phase "Shrink the Extension" to delete the wires in the reverse order that they are introduced.

More technical issues raise when we build iO leveraging the series of incremental changes above. As we will show later, our construction of iO for $\Delta$-equivalent circuits does not hide the *topology* of the input circuit. As a result, in our $\Delta$-equivalence definition, we require the circuits $C_1, C_2$ and their intermediate circuits $C'_1, \ldots, C'_\ell$ have the *same topology*. To further preserve the topology of the circuit, we pad them to the same topology. We defer the details to the full version of the paper.

*B. Construction of iO for $\Delta$-equivalent Circuits*

We now describe our construction of iO for $\Delta$-equivalent circuits. Our high-level strategy is as follows:

– We first consider a notion of $\delta$iO, namely, iO for circuits that only differ by a small subcircuit. Specifically, we build $\delta$iO for any two circuits that only differ by two logarithmic-size functionally equivalent subcircuits.
– Next, we use $\delta$iO to obfuscate $\Delta$-equivalent circuits as follows. Recall that for any $\Delta$-equivalent circuits $C_1, C_2$, there is a polynomial number of intermediate circuits $C_1 = C'_1, C'_2, \ldots, C'_\ell = C_2$, and each two adjacent circuits $C'_i, C'_{i+1}$ only differ by two functionality equivalent logarithmic subcircuits. From the first step, it follows that for every $i$, $\delta\mathsf{iO}(C'_i)$ and $\delta\mathsf{iO}(C'_{i+1})$ are indistinguishable. By a hybrid argument, we can now establish the indistinguishability of $\delta\mathsf{iO}(C_1)$ and $\delta\mathsf{iO}(C_2)$.

Let us explain why this approach overcomes the input-length barrier. Whether two circuits only differ by two functionality equivalent subcircuits of *logarithmic* size can be decided in polynomial-time, since we only need to check all inputs to the *subcircuit* instead of all inputs to the *entire circuit*. Hence, the input-length barrier does not apply to $\delta$iO. Therefore, we can hope to build $\delta$iO without a security loss that is exponential in the input length.

Thus, the main task towards our goal is to build $\delta$iO. Towards this end, we present a new template for obfuscation that preserves the *topology* of the input circuit. This feature is crucial to proving security without incurring a loss exponential in the input length. In particular, it allows us to make "local" changes to leverage the fact the input pair of circuits only differ by a logarithmic-size functionally equivalent subcircuit. To the best of our understanding, this property is not satisfied by prior templates for obfuscation (see, e.g., [6], [14], [24], [12], [51], [40]).

**Our Gate-by-Gate iO Template.** Our first attempt is to mimic the gate-by-gate construction of garbled circuits [70] that preserves the structure of the input circuit. Specifically, for each gate $g$ in an input circuit $C$, we use a "small" iO to obfuscate the gate functionality. Note that the input and output wires need to be encrypted since otherwise, an adversary can run the obfuscated program on arbitrary inputs and observe the truth table of the gate $g$. Towards this end, we associate a puncturable PRF key to each wire of the circuit, and use it to encrypt the wire value. Then, for each gate $g$, we obfuscate the following circuit $\mathsf{Gate}_g(\cdot,\cdot)$: it takes as input two ciphertexts that correspond to encryptions of $g$'s input wires. It first decrypts the ciphertexts, computes the functionality of the gate $g$, and then encrypts the output wire value. In order to perform the decryption and encryption steps, $\mathsf{Gate}_g$ contains the puncturable PRF keys for the input and output wires of $g$ hardwired in its description. The obfuscated circuit consists of the obfuscation of $\mathsf{iO}(\mathsf{Gate}_g)$'s for every gate $g$ in $C$.

In order to prove security, the main idea is to only modify the obfuscation of the gates that correspond to the logarithmic-size subcircuit where the input circuits differ. Note that our use of existing iO schemes (that incur security loss exponential in the input length) does not pose a problem towards bypassing the input-length barrier because the input length of each $\mathsf{Gate}_g$ is much smaller than the input length of the entire circuit $C$.

**Mix-and-Match Attacks.** This initial attempt, unfortunately, suffers from "mix-and-match" attacks. An adversary can run the obfuscated program for several different inputs, and keep the ciphertexts of the intermediate wires. Later, the adversary can provide the "mixed" input ciphertexts *sourced from different inputs* to some gate $\mathsf{iO}(\mathsf{Gate}_g)$. Then the adversary might learn more input-output pairs of $\mathsf{Gate}_g$ than the functionality of the circuit $\mathsf{Gate}_g$ should have provided, and thus we have no hope to prove the security of the above construction.

To prevent such attacks, we can modify the construction as follows: let $\mathsf{ct}_l, \mathsf{ct}_r$ denote the "left" and "right" input ciphertexts to $\mathsf{Gate}_g$. The modified $\mathsf{Gate}_g$ additionally takes the *entire* inputs $x_l, x_r$ to $C$ that lead to the input ciphertexts $\mathsf{ct}_l, \mathsf{ct}_r$ and checks whether $x_l = x_r$. In order to "tie" the entire input with a ciphertext, we use another puncturable PRF to compute a message-authenticate code (MAC) over the pair $(\mathsf{ct}_l, x_l)$ and similarly $(\mathsf{ct}_r, x_r)$

It is not difficult to see that this modified construction prevents mix-and-match' attacks. Intuitively, only the ciphertexts generated by $\mathsf{Gate}_g$ can have a valid MAC, and hence the mix-and-match attacks can be caught by the consistency check over the inputs $x_l, x_r$. Unfortunately, however, the input length of $\mathsf{Gate}_g$ is now as large as the input length of $C$. This means that this construction will incur a security loss exponential in the input length of $C$.

**An Intermediate Step.** Towards overcoming this problem, we first describe a modified construction that improves upon the above but only for specific circuits, namely, ones in $\mathsf{NC}^0$. As we will see shortly, it serves as a useful basis towards our final solution for general circuits.

Our starting idea is to leverage the fact that each gate in $C$ might not depend on the *entire* input of $C$. Hence, we can modify $\mathsf{Gate}_g$ such that it only takes as input the input wire values of $C$ that $g$ depends upon. To characterize *dependency*, we introduce the notation $\mathsf{dep}(w)$ to denote the set containing *all the intermediate wires* that a wire $w$ depends upon, excluding itself. Note that $\mathsf{dep}(w)$ includes not only the input wires but also the internal wires of $C$.

The security loss incurred by this modified construction is exponential in the input length of $\mathsf{Gate}_g$. This loss is small when $C$ is in $\mathsf{NC}^0$ since any output bit of an $\mathsf{NC}^0$ circuit only depends on a constant number of input bits. However, for general circuits, $\mathsf{dep}(l)$ and $\mathsf{dep}(r)$ may contain the entire input in the worst case. In such a scenario, the security loss is still exponential in the input length of $C$.

**Shrinking Input Length via Hashing.** To resolve this issue, we observe that in the above security proof, $\mathsf{Gate}_g^{\mathsf{direct}}$ does not even need to know every ciphertext in $\mathsf{dep}(l) \cup \mathsf{dep}(r)$ to compute the wire value of $o$. Instead, the wire $o$ only depends on the wires in $\mathsf{dep}(o)$ that are also the input wires of the subcircuit $S$. For ease of representation, we use $\mathsf{inp}(S)$ to denote the input to $S$. Since the size of $\mathsf{dep}(o) \cap \mathsf{inp}(S)$ is only logarithmic, if we modify $\mathsf{Gate}_g$ to take as input the ciphertexts in $\mathsf{dep}(o) \cap \mathsf{inp}(S)$ instead, then we significantly shorten the input length of $\mathsf{Gate}_g$.

However, we can not provide the above set as an explicit input to $\mathsf{Gate}_g$ since $S$ is not known in the *construction* of $\delta$iO; instead, it is only available in the security reduction. If we hardwire $S$ in (the public description of) $\mathsf{Gate}_g$ in an intermediate hybrid of the security proof, then we can not hope to argue indistinguishability. Hence, we need to *hide* the set $S$ and at the same time also provide the above set of ciphertexts in $\mathsf{dep}(o) \cap \mathsf{inp}(S)$ as an input to $\mathsf{Gate}_g$.

To achieve these two properties simultaneously, we use a somewhere extractable hash function (SEH) [46] to hash the ciphertexts in $\mathsf{dep}(l)$ and the ciphertexts in $\mathsf{dep}(r)$. We set the hash function to be extractable for the ciphertexts in $\mathsf{dep}(l) \cap \mathsf{inp}(S)$ and $\mathsf{dep}(r) \cap \mathsf{inp}(S)$. The key indistinguishability property of SEH guarantees that the extraction locations are hidden in the hash key. Moreover, the size of the SEH hash value grows linearly in $|S|$.

Next, we modify the circuit $\mathsf{Gate}_g$ to take $\mathsf{Hash}(CT_l), \mathsf{Hash}(CT_r)$ as additional inputs, where $CT_l$ (resp. $CT_r$) contains all ciphertexts that the wire $l$ (resp. $r$) depends on. Then in the security proof, for each two adjacent intermediate circuits $C_i', C_{i+1}'$, we first switch the set $S$ to be the subcircuit that $C_i'$ and $C_{i+1}'$ differ on. Then, we replace $\mathsf{Gate}_g$ with a new $\mathsf{Gate}_g^{\mathsf{direct}'}$ that extracts the sets of ciphertexts in $\mathsf{dep}(o) \cap \mathsf{inp}(S)$ from the hash values and computes the output wires $o$ directly from them.

However, an issue arises in arguing security since we need to enforce the consistency check of the ciphertexts in $\mathsf{dep}(l)$ and the ciphertexts in $\mathsf{dep}(r)$ given only their hash values. A natural idea is to further attach a *succinct* non-interactive proof that proves that the two hash values are consistent. Note that we seemingly need such a proof to be *statistically sound*; such proofs, however, are unlikely to exist.

Our key observation is that we in fact do not need a succinct proof with full statistical soundness. Instead, we only succinct non-interactive arguments (SNARGs) with the following *somewhere statistical soundness* property: for two hash values computed as above, the extracted ciphertexts are consistent. Namely, given the hash values $h_l, h_r, h_o$ with respect to $\mathsf{dep}(l), \mathsf{dep}(r), \mathsf{dep}(o)$, respectively, if the extracted ciphertexts in $\mathsf{dep}(l) \cap \mathsf{inp}(S)$ and $\mathsf{dep}(o) \cap \mathsf{inp}(S)$ are inconsistent, or the extracted ciphertexts in $\mathsf{dep}(r) \cap \mathsf{inp}(S)$ and $\mathsf{dep}(o) \cap \mathsf{inp}(S)$ are inconsistent, then any proof computed by an unbounded cheating prover must be rejected.

We build such somewhere statistically sound SNARGs with only *poly-logarithmic* size proof and verification time from the polynomial hardness of learning with errors (LWE) by relying on the techniques in the recent work of [25]. In [25], the authors constructed SNARGs for the so-called *batch index* language with (semi-adaptive) somewhere extraction property from LWE, where an index language is an $\mathcal{NP}$ language where the instances are treated as indices that can be described in a logarithmic number of bits. We observe that a minor modification of their construction achieves (semi-adaptive) somewhere *statistical* soundness.

Armed with somewhere statistically sound SNARGs for the batch index language, we show how to build an SEH with consistency proofs. We start with the somewhere statistical binding hash construction of [46]. Their construction also allows extraction of the binding positions, and hence is also an SEH. Moreover, their construction has a Merkle tree structure, and thus supports succinct local openings. Namely, one can use a root-to-leaf in the Merkle tree to serve as a small-size opening for each bit in the string being hashed. To hash the index set $CT_l$, we first assign a unique integer to each wire. Then we arrange the elements in $CT_l$ as an array. At the index $w$, if $w$ index is non-empty in $CT_l$ then we put $\mathsf{ct}_w$ at the $w$-th index. Otherwise, we put a special symbol $\perp$ at the $w$-th index. To generate a consistency proof for $h_l$ and $h_o$, we use a SNARG for batch-index language to prove that for each wire $w$, there exists valid local openings to $h_l$ and $h_o$ at the index $w$, and if $CT_l$ has a non-empty element at the index $w$, then $CT_o$ also has the same element at the index $w$. Then the somewhere statistical soundness of SNARGs for batch-index implies the property we want from the consistency proof.

It looks like we have bypassed the input-length barrier, since the input length to $\mathsf{Gate}_g$ seems to be independent of the input length of $C$. However, a careful examination reveals that this is not the case. Specifically, the bit-length of the hash value $h_o$ is at least the size of one ciphertext plus a $\mathsf{poly}(\lambda)$ term, and the size of one ciphertext is at least the size of $h_l$ or $h_r$. Hence, we have $|h_o| \geq |h_l| + \mathsf{poly}(\lambda)$. Therefore, the size of the hash value $h_o$ grows at least linearly in the depth of the circuit. This leads to a linear dependence on the depth of the circuit in the input length of $\mathsf{Gate}_g$.

**Removing the Depth Dependence.** To overcome this issue, we need to further shrink the hash values. Towards this end, our key observation is that we only require a weaker extraction property from SEH: instead of extracting the ciphertexts, we only need to extract the underlying messages. Hence, we use a fully homomorphic encryption scheme to encrypt the SEH extraction trapdoor together with the puncturable PRF keys for the wires whose values we wish to extract from SEH. Then we homomorphically extract the underlying messages.

**Full Version.** A formal presentation of all our results is deferred to the full version of the paper.

## References


[1] Shweta Agrawal. Indistinguishability obfuscation without multilinear maps: New methods for bootstrapping and instantiation. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part I*, volume 11476 of *Lecture Notes in Computer Science*, pages 191–225, Darmstadt, Germany, May 19–23, 2019. Springer, Heidelberg, Germany. 1

[2] Shweta Agrawal and Alice Pellet-Mary. Indistinguishability obfuscation without maps: Attacks and fixes for noisy linear FE. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020, Part I*, volume 12105 of *Lecture Notes in Computer Science*, pages 110–140, Zagreb, Croatia, May 10–14, 2020. Springer, Heidelberg, Germany. 1

[3] Prabhanjan Ananth, Dan Boneh, Sanjam Garg, Amit Sahai, and Mark Zhandry. Differing-inputs obfuscation and applications. Cryptology ePrint Archive, Report 2013/689, 2013. https://eprint.iacr.org/2013/689. 3

[4] Prabhanjan Ananth, Aayush Jain, Huijia Lin, Christian Matt, and Amit Sahai. Indistinguishability obfuscation without multilinear maps: New paradigms via low degree weak pseudorandomness and security amplification. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 284–332, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany. 1

[5] Prabhanjan Ananth, Aayush Jain, and Amit Sahai. Robust transforming combiners from indistinguishability obfuscation to functional encryption. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017, Part I*, volume 10210 of *Lecture Notes in Computer Science*, pages 91–121, Paris, France, April 30 – May 4, 2017. Springer, Heidelberg, Germany. 7

[6] Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 308–326, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany. 1, 3, 9

[7] Prabhanjan Ananth and Amit Sahai. Projective arithmetic functional encryption and indistinguishability obfuscation from degree-5 multilinear maps. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017, Part I*, volume 10210 of *Lecture Notes in Computer Science*, pages 152–181, Paris, France, April 30 – May 4, 2017. Springer, Heidelberg, Germany. 1

[8] 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, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*,

pages 1–18, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Heidelberg, Germany. 1

[9] Mihir Bellare, Igors Stepanovs, and Brent Waters. New negative results on differing-inputs obfuscation. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 792–821, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany. 3

[10] Nir Bitansky, Ran Canetti, Sanjam Garg, Justin Holmgren, Abhishek Jain, Huijia Lin, Rafael Pass, Sidharth Telang, and Vinod Vaikuntanathan. Indistinguishability obfuscation for RAM programs and succinct randomized encodings. *SIAM J. Comput.*, 47(3):1123–1210, 2018. 1

[11] Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. On the existence of extractable one-way functions. In David B. Shmoys, editor, *46th Annual ACM Symposium on Theory of Computing*, pages 505–514, New York, NY, USA, May 31 – June 3, 2014. ACM Press. 3

[12] Nir Bitansky, Sanjam Garg, Huijia Lin, Rafael Pass, and Sidharth Telang. Succinct randomized encodings and their applications. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th Annual ACM Symposium on Theory of Computing*, pages 439–448, Portland, OR, USA, June 14–17, 2015. ACM Press. 3, 9

[13] Nir Bitansky, Omer Paneth, and Alon Rosen. On the cryptographic hardness of finding a Nash equilibrium. In Venkatesan Guruswami, editor, *56th Annual Symposium on Foundations of Computer Science*, pages 1480–1498, Berkeley, CA, USA, October 17–20, 2015. IEEE Computer Society Press. 1

[14] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In Venkatesan Guruswami, editor, *56th Annual Symposium on Foundations of Computer Science*, pages 171–190, Berkeley, CA, USA, October 17–20, 2015. IEEE Computer Society Press. 1, 3, 9

[15] Dan Boneh and Ramarathnam Venkatesan. Breaking RSA may not be equivalent to factoring. In Kaisa Nyberg, editor, *Advances in Cryptology – EUROCRYPT'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 59–71, Espoo, Finland, May 31 – June 4, 1998. Springer, Heidelberg, Germany. 1

[16] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology – ASIACRYPT 2013, Part II*, volume 8270 of *Lecture Notes in Computer Science*, pages 280–300, Bengalore, India, December 1–5, 2013. Springer, Heidelberg, Germany. 4, 5

[17] Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 480–499, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany. 1

[18] Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability obfuscation. In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 52–73, San Diego, CA, USA, February 24–26, 2014. Springer, Heidelberg, Germany. 3

[19] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In Hugo Krawczyk, editor, *PKC 2014: 17th International Conference on Theory and Practice of Public Key Cryptography*, volume 8383 of *Lecture Notes in Computer Science*, pages 501–519, Buenos Aires, Argentina, March 26–28, 2014. Springer, Heidelberg, Germany. 4, 5

[20] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Candidate iO from homomorphic encryption schemes. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020, Part I*, volume 12105 of *Lecture Notes in Computer Science*, pages 79–109, Zagreb, Croatia, May 10–14, 2020. Springer, Heidelberg, Germany. 1

[21] Mark Bun and Mark Zhandry. Order-revealing encryption and the hardness of private learning. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A: 13th Theory of Cryptography Conference, Part I*, volume 9562 of *Lecture Notes in Computer Science*, pages 176–206, Tel Aviv, Israel, January 10–13, 2016. Springer, Heidelberg, Germany. 1

[22] Samuel Buss. *Bounded Arithmetic*. Bibliopolis, Naples, Italy, 1986. 3, 5, 6

[23] Samuel R. Buss. Chapter i - an introduction to proof theory. In Samuel R. Buss, editor, *Handbook of Proof Theory*, volume 137 of *Studies in Logic and the Foundations of Mathematics*, pages 1–78. Elsevier, 1998. 2, 3, 4, 6

[24] Ran Canetti, Justin Holmgren, Abhishek Jain, and Vinod Vaikuntanathan. Succinct garbling and indistinguishability obfuscation for RAM programs. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th Annual ACM Symposium on Theory of Computing*, pages 429–437, Portland, OR, USA, June 14–17, 2015. ACM Press. 3, 9

[25] Arka Rai Choudhuri, Abhishek Jain, and Zhengzhong Jin. Snargs for $\mathcal{P}$ from lwe. In *FOCS*, 2021. https://ia.cr/2021/808. 3, 10

[26] 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. 6

[27] Aloni Cohen, Justin Holmgren, Ryo Nishimaki, Vinod Vaikuntanathan, and Daniel Wichs. Watermarking cryptographic capabilities. In Daniel Wichs and Yishay Mansour, editors, *48th Annual ACM Symposium on Theory of Computing*, pages 1115–1127, Cambridge, MA, USA, June 18–21, 2016. ACM Press. 1

[28] Stephen Cook. Connecting complexity classes, weak formal theories, and propositional proof systems. https://www.cs.toronto.edu/~sacook/slidesPrint.pdf. page 10-15. 6

[29] Stephen Cook and Jan Krajíček. Consequences of the provability of NP $\subseteq$ P/poly. *Journal of Symbolic Logic*, 72(4):1353 – 1371, 2007. 6

[30] 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, 4, 5, 6

[31] Stephen A. Cook and Robert A. Reckhow. The relative efficiency of propositional proof systems. *Journal of Symbolic Logic*, 44(1):36–50, 1979. 6

[32] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31:469–472, 1985. 5, 6

[33] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th Annual Symposium on Foundations of Computer Science*, pages 40–49, Berkeley, CA, USA, October 26–29, 2013. IEEE Computer Society Press. 1

[34] Sanjam Garg, Craig Gentry, Shai Halevi, and Daniel Wichs. On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 518–535, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany. 3

[35] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 467–476, Palo Alto, CA, USA, June 1–4, 2013. ACM Press. 1, 2

[36] Sanjam Garg and Omkant Pandey. Incremental program obfuscation. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 193–223, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany. 7

[37] Sanjam Garg, Omkant Pandey, and Akshayaram Srinivasan. Revisiting the cryptographic hardness of finding a nash equilibrium. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 579–604, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany. 2, 6

[38] Sanjam Garg, Omkant Pandey, Akshayaram Srinivasan, and Mark Zhandry. Breaking the sub-exponential barrier in obfustopia. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017, Part III*, volume 10212 of *Lecture Notes in Computer Science*, pages 156–181, Paris, France, April 30 – May 4, 2017. Springer, Heidelberg, Germany. 2, 6

[39] Sanjam Garg and Akshayaram Srinivasan. Single-key to multi-key functional encryption with polynomial loss. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B: 14th Theory of Cryptography Conference, Part II*, volume 9986 of *Lecture Notes in Computer Science*, pages 419–442, Beijing, China, October 31 – November 3, 2016. Springer, Heidelberg, Germany. 2, 6

[40] Sanjam Garg and Akshayaram Srinivasan. A simple construction of iO for turing machines. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018: 16th Theory of Cryptography Conference, Part II*, volume 11240 of *Lecture Notes in Computer Science*, pages 425–454, Panaji, India, November 11–14, 2018. Springer, Heidelberg, Germany. 9

[41] Romain Gay, Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from simple-to-state hard problems: New assumptions, new techniques, and simplification. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021, Part III*, volume 12698 of *Lecture Notes in Computer Science*, pages 97–126, Zagreb, Croatia, October 17–21, 2021. Springer, Heidelberg, Germany. 1

[42] Romain Gay and Rafael Pass. Indistinguishability obfuscation from circular security. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 736–749. ACM, 2021. 1

[43] Craig Gentry, Allison Bishop Lewko, Amit Sahai, and Brent Waters. Indistinguishability obfuscation from the multilinear subgroup elimination assumption. In Venkatesan Guruswami, editor, *56th Annual Symposium on Foundations of Computer Science*, pages 151–170, Berkeley, CA, USA, October 17–20, 2015. IEEE Computer Society Press. 1

[44] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd Annual ACM Symposium on Theory of Computing*, pages 99–108, San Jose, CA, USA, June 6–8, 2011. ACM Press. 2, 4

[45] Satoshi Hada. Zero-knowledge and code obfuscation. In Tatsuaki Okamoto, editor, *Advances in Cryptology – ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 443–457, Kyoto, Japan, December 3–7, 2000. Springer, Heidelberg, Germany. 1

[46] Pavel Hubacek and Daniel Wichs. On the communication complexity of secure function evaluation with long output. In Tim Roughgarden, editor, *ITCS 2015: 6th Conference on Innovations in Theoretical Computer Science*, pages 163–172, Rehovot, Israel, January 11–13, 2015. Association for Computing Machinery. 3, 9, 10

[47] Yuval Ishai, Omkant Pandey, and Amit Sahai. Public-coin differing-inputs obfuscation and its applications. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 668–697, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany. 3

[48] Aayush Jain, Huijia Lin, Christian Matt, and Amit Sahai. How to leverage hardness of constant-degree expanding polynomials overa $\mathbb{R}$ to build $i\mathcal{O}$. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part I*, volume 11476 of *Lecture Notes in Computer Science*, pages 251–281, Darmstadt, Germany, May 19–23, 2019. Springer, Heidelberg, Germany. 1

[49] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 60–73. ACM, 2021. 1, 4

[50] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013: 20th Conference on Computer and Communications Security*, pages 669–684, Berlin, Germany, November 4–8, 2013. ACM Press. 4, 5

[51] Venkata Koppula, Allison Bishop Lewko, and Brent Waters. Indistinguishability obfuscation for turing machines with unbounded memory. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th Annual ACM Symposium on Theory of Computing*, pages 419–428, Portland, OR, USA, June 14–17, 2015. ACM Press. 3, 9

[52] Jan Krajíček and Pavel Pudlák. Quantified propositional calculi and fragments of bounded arithmetic. *Mathematical Logic Quarterly*, 36(1):29–46, 1990. 6

[53] Huijia Lin. Indistinguishability obfuscation from constant-degree graded encoding schemes. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part I*, volume 9665 of *Lecture Notes in Computer Science*, pages 28–57, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany. 1

[54] Huijia Lin. Indistinguishability obfuscation from SXDH on 5-linear maps and locality-5 PRGs. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 599–629, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany. 1

[55] 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: 19th International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 9615 of *Lecture Notes in Computer Science*, pages 447–462, Taipei, Taiwan, March 6–9, 2016. Springer, Heidelberg, Germany. 1

[56] 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: 13th Theory of Cryptography Conference, Part I*, volume 9562 of *Lecture Notes in Computer Science*, pages 96–124, Tel Aviv, Israel, January 10–13, 2016. Springer, Heidelberg, Germany. 1, 3

[57] Huijia Lin and Stefano Tessaro. Indistinguishability obfuscation from trilinear maps and block-wise local PRGs. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 630–660, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany. 1

[58] Huijia Lin and Vinod Vaikuntanathan. Indistinguishability obfuscation from DDH-like assumptions on constant-degree graded encodings. In Irit Dinur, editor, *57th Annual Symposium on Foundations of Computer Science*, pages 11–20, New Brunswick, NJ, USA, October 9–11, 2016. IEEE Computer Society Press. 1

[59] Qipeng Liu and Mark Zhandry. Decomposable obfuscation: A framework for building applications of obfuscation from polynomial hardness. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part I*, volume 10677 of *Lecture Notes in Computer Science*, pages 138–169, Baltimore, MD, USA, November 12–15, 2017. Springer, Heidelberg, Germany. 1, 2, 6, 7

[60] Silvio Micali. Computationally sound proofs. *SIAM Journal on Computing*, 30(4):1253–1298, 2000. 2

[61] Moni Naor. On cryptographic assumptions and challenges (invited talk). In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 96–109, Santa Barbara, CA, USA, August 17–21, 2003. Springer, Heidelberg, Germany. 2

[62] Rohit Parikh. Existence and feasibility in arithmetic. *The Journal of Symbolic Logic*, 36(3):494–508, 1971. 3

[63] J. Paris and A. Wilkie. Counting problems in bounded arithmetic. In Carlos Augusto Di Prisco, editor, *Methods in Mathematical Logic*, pages 317–340, Berlin, Heidelberg, 1985. Springer Berlin Heidelberg. 6

[64] Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation from semantically-secure multilinear encodings. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 500–517, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany. 1

[65] 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. 6

[66] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 84–93, Baltimore, MA, USA, May 22–24, 2005. ACM Press. 5, 6

[67] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: Deniable encryption, and more. *SIAM J. Comput.*, 50(3):857–908, 2021. 1, 2, 4, 5, 6

[68] 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). 6

[69] Hoeteck Wee and Daniel Wichs. Candidate obfuscation via oblivious LWE sampling. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021, Part III*, volume 12698 of *Lecture Notes in Computer Science*, pages 127–156, Zagreb, Croatia, October 17–21, 2021. Springer, Heidelberg, Germany. 1

[70] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science*, pages 162–167, Toronto, Ontario, Canada, October 27–29, 1986. IEEE Computer Society Press. 9