

# Public-Coin Differing-Inputs Obfuscation and Its Applications

Yuval Ishai  
Technion  
yuvali@cs.technion.ac.il

Omkant Pandey  
UIUC  
omkant@uiuc.edu

Amit Sahai  
UCLA  
sahai@cs.ucla.edu

## Abstract

*Differing inputs obfuscation* (diO) is a strengthening of indistinguishability obfuscation (iO) that has recently found applications to improving the efficiency and generality of obfuscation, functional encryption, and related primitives. Roughly speaking, a diO scheme ensures that the obfuscations of two efficiently generated programs are indistinguishable not only if the two programs are equivalent, but also if it is hard to find an input on which their outputs differ. The above “indistinguishability” and “hardness” conditions should hold even in the presence of an auxiliary input that is generated together with the programs.

The recent works of Boyle and Pass (ePrint 2013) and Garg et al. (Crypto 2014) cast serious doubt on the plausibility of general-purpose diO with respect to general auxiliary inputs. This leaves open the existence of a variant of diO that is plausible, simple, and useful for applications.

We suggest such a diO variant that we call *public-coin* diO. A public-coin diO restricts the original definition of diO by requiring the auxiliary input to be a public random string which is given as input to all relevant algorithms. In contrast to standard diO, we argue that it remains very plausible that current candidate constructions of iO for circuits satisfy the public-coin diO requirement.

We demonstrate the usefulness of the new notion by showing that several applications of diO can be obtained by relying on the public-coin variant instead. These include constructions of *succinct* obfuscation and functional encryption schemes for Turing Machines, where the size of the obfuscated code or keys is essentially independent of the input-length, running time and space.

# 1 Introduction

General-purpose obfuscation refers to the concept of transforming an arbitrary program so that its functionality is preserved, but otherwise rendering the program “unintelligible.” This concept has intrigued cryptographers for decades, and led to multiple attempts at formalization (most notably [BGI<sup>+</sup>12]). A critical goal in obfuscation research has been to identify the strongest notions of obfuscation that are *plausible* and have wide applicability. General-purpose obfuscation, however, has proven to be perched precariously between possibility and impossibility.

On the one extreme, *virtual black-box obfuscation* (VBB) is an ideal form of obfuscation that captures the intuitive notion of obfuscation and often can be directly used in applications. Unfortunately, this notion is impossible in the sense that it provably cannot be realized for certain contrived classes of programs [BGI<sup>+</sup>12], or for quite large classes of programs under contrived auxiliary inputs [GK05].

On the other extreme, the most liberal notion of general-purpose obfuscation is *indistinguishability obfuscation* (iO) [BGI<sup>+</sup>12, GR07]. An iO scheme for a class of “programs” is an efficient randomized algorithm that maps any program  $P$  into a functionally equivalent obfuscated program  $P'$  such that if  $P_1$  and  $P_2$  compute the same function then their obfuscations  $P'_1$  and  $P'_2$  are computationally indistinguishable.

The first plausible construction of a general-purpose iO scheme was given in 2013 by Garg et al. [GGH<sup>+</sup>13b]. This construction and similar ones from [BR14, BGK<sup>+</sup>14] render the existence of an iO scheme a *plausible assumption*, since there are currently no attacks or other evidence suggesting that these constructions fail to meet the iO requirements. In particular, no theoretical impossibility results are known for iO schemes even for contrived classes of programs and auxiliary inputs.

On the downside, the security guarantee of iO appears to be too weak for most natural applications of obfuscation. A recent line of work, originating from [GGH<sup>+</sup>13b, SW14], has made impressive progress on applying iO towards a wide array of cryptographic applications. However, these applications are still not as broad as one might expect, and the corresponding constructions and their analysis are significantly more complex than those that could be obtained from an ideal obfuscation primitive. Indeed, this may be the case because the definition of iO seems to capture only a quite minimal property of obfuscation.

**In search of the “strongest plausible assumption.”** The above limitations of iO motivate the search for stronger notions of obfuscation that support more applications and give rise to simpler constructions and security proofs. Such a stronger notion should be *plausible*, in the sense that current candidate obfuscation constructions can be conjectured to satisfy the stronger requirements without contradicting other theoretical or empirical evidence. Another important feature is *succinct description*, ruling out contrived notions whose security requirements refer separately to each application. This leads to the following question, which is at the center of our work:

*Is there a plausible, useful, and succinctly described notion of obfuscation that captures stronger security requirements than indistinguishability obfuscation?*

**Differing inputs obfuscation.** A seemingly promising positive answer to our question was given by the notion of *differing inputs obfuscation* (diO). First proposed in [BGI<sup>+</sup>12] and recently revisited in [ABG<sup>+</sup>13, BCP14], diO has found a diverse array of applications that do not seem to follow from traditional iO (see below for more details). Roughly speaking, a diO scheme ensures that the obfuscations of two efficiently generated programs  $P_1$  and  $P_2$  are indistinguishable not only if they compute the same function, but also if it is hard for an adversary to find an input  $x$  on which the two programs differ, namely  $x$  such that  $P_1(x) \neq P_2(x)$ . The above “indistinguishability” and “hardness” conditions should hold even in the presence of an auxiliary input  $\text{aux}$  that is generated together with

the programs and is given as input both to the adversary trying to find an input  $x$  as above and to the distinguisher who tries to distinguish between the two obfuscated programs. Indeed, different applications give rise to different distributions of  $\text{aux}$ .

However, the recent works of [BP13, GGHW14] cast serious doubts on the plausibility of general-purpose diO with respect to general auxiliary inputs. In particular, [GGHW14] showed that the existence of diO with respect to arbitrary auxiliary inputs contradicts a certain “special-purpose obfuscation” conjecture. At a high level, the impossibility result of [GGHW14] proceeds as follows: Consider a pair of programs  $P_1$  and  $P_2$  that produce different one-bit outputs only on inputs  $x = (m, \sigma)$  that consist of valid message-signature pairs with respect to a fixed unforgeable signature scheme verification key. Now we consider another program  $D$  which takes a program  $P$  as input, and then hashes  $P$  to compute  $m = h(P)$  together with a signature  $\sigma$  on  $m$ . It then feeds  $x = (m, \sigma)$  as input to  $P$ , and outputs the first bit of  $P(x)$ . Now, the auxiliary input given to the adversary will be a “special-purpose obfuscation” of this program  $D$ . The special-purpose obfuscation conjecture of [GGHW14] is that even given this auxiliary input, it is still hard for the adversary to obtain any valid message-signature pair. This assumption seems quite plausible, for instance if  $D$  is obfuscated using the obfuscators of [GGH<sup>+</sup>13b, BR14, BGK<sup>+</sup>14]. Now, it is evident that the adversary can distinguish between any obfuscations of  $P_1$  and  $P_2$  using the auxiliary input, and yet by the special-purpose assumption, the adversary cannot compute any valid message-signature pair, and therefore cannot find a differing input.

**What causes impossibility for diO?** If we would like to consider general notions of obfuscation that capture security requirements beyond indistinguishability obfuscation, it is imperative that we understand the roots of impossibility for diO. Indeed, it is not difficult to evade the impossibility results of [BP13, GGHW14] by simply assuming that diO only holds with respect to *specific* auxiliary input distributions, as has been suggested in [ABG<sup>+</sup>13, BCP14, BP13]. However, this approach would yield disparate special-purpose variants of the diO assumption for each potential application scenario, with little clarity on why any particular such assumption should be valid. This would defeat our goal of obtaining a general and *succinctly described* assumption. Therefore, we seek to understand the essential flaw that the works of [BP13, GGHW14], and others like it, can exploit using auxiliary inputs.

Our starting point is the suggestion, made in several previous works [BCP14, BP13, BCCT13, BCP14], to specifically consider an auxiliary input that is *uniformly random*, since at least some applications of diO and other suspect primitives seem to work with just uniformly random auxiliary inputs. This certainly seems a great deal safer, and does avoid the specific impossibility results known. However, our starting observation is that even a uniformly random auxiliary input could potentially be problematic in that the auxiliary input could be chosen so that it is the output of a one-way permutation – thus there would still be a secret about the auxiliary input that is hidden from the adversary. Although we don’t currently see a way to exploit this to obtain an impossibility result, could this eventually lead to trouble?

Indeed, in the negative results of [BP13, GGHW14], and similarly in other impossibility results using auxiliary inputs (e.g. [GK05]), it is critical that the auxiliary input can contain *trapdoor information*. In other words, secrets can be used to choose the auxiliary input, and these secrets are not themselves revealed to the adversary. (In the case of [GGHW14], this trapdoor information includes the secret signing key of the signature scheme, and the randomness used to obtain the obfuscation of the program  $D$ .) Our objective, then, is to formulate a notion of diO that avoids this possibility altogether.

**Public-coin differing inputs obfuscation.** Building upon the observations above, we introduce the notion of *public-coin* diO. A public-coin diO restricts the original definition of diO by requiring the auxiliary input  $\text{aux}$  to be the *actual random coins* that were given to the program that sampled  $P_1$  and  $P_2$ . Thus, in public-coin diO, the auxiliary input is not chosen by a sampling algorithm, but rather the auxiliary input is simply a set of public coins that are made available to all algorithms. In particular, this means that it must be hard to find an input  $x$  such that  $P_1(x) \neq P_2(x)$  even given all information about how  $P_1$  and  $P_2$  were generated. This rules out the possibility of planting a trapdoor in the auxiliary input, an option that was critical for proving the negative evidence against diO [BP13, GGHW14].

Indeed, we know of no evidence of impossibility for public-coin diO. The public coin restriction appears to cut a wide path around the impossibility result of [GGHW14]. Intuitively, public-coin diO requires that even “nature” – which is computationally bounded but all-seeing – cannot find any inputs on which the two programs  $P_1$  and  $P_2$  will differ. This is important because not only [BP13, GGHW14], but also all previous impossibility results on VBB obfuscation (e.g [BGI<sup>+</sup>12, GK05]) used the input/output behavior of the program to plant hidden inputs on which the output of the program is too revealing. But in public-coin diO, the existence of such planted inputs would automatically rule out any security guarantee from diO, since given knowledge of these planted inputs it is easy to find a differing input. Thus, intuitively speaking, this suggests that an impossibility result for public-coin diO would need to find actual weaknesses in the obfuscation mechanism itself – some way to distinguish obfuscations that does not use the input/output behavior of the underlying programs in any way. Existing security proofs in generic<sup>1</sup> models [BR14, BGK<sup>+</sup>14] offer strong evidence that such an impossibility result is unlikely to exist.

We also view our public coin restriction as being a natural limitation to place on diO. Indeed, while our notion is novel in the context of obfuscation, it is reminiscent of (but also quite different from) other scenarios in cryptography where the public-coin restriction was introduced in order to prevent the existence of trapdoor information. For example, in the context of trapdoor permutations, it was observed that allowing the input sampler to use general auxiliary information can lead to problematic constructions technically satisfying the definition of a trapdoor permutation but rendering applications of trapdoor permutations insecure [GR13]. To prevent this, the notion of enhanced trapdoor permutations limits the input samplers to be given only public coins as input. Separately, in the context of collision-resistant hash functions, the distinction between secret-coin and public-coin collision-resistant hash families was studied in [HR04], where it was noted that some applications of collision-resistant hashing require public coins, since secret coins may enable the party picking the key to know impermissible trapdoor information. While these other public-coin primitives are quite different in nature from ours, we view our notion of public-coin diO to be as natural a variant as enhanced trapdoor permutations or public-coin collision resistant hash functions.

Bellare, Stepanovs, and Tessaro [BST14] presented a definitional framework for diO where security of obfuscation is parameterized by a class of samplers (instead of applying for all circuits). This allows one to define and study restricted forms of diO by considering different types of samplers. The central object in this framework is then to identify appropriate types of samplers (which, for example, do not suffer from the negative results of [BP13, GGHW14]).

Our notion of public-coin diO can be cast in the framework of [BST14] by considering samplers that are *public-coin*. We put forward the case of public-coin samplers as an important notion worthy of further study. Our work demonstrates that the public-coin case is of general interest, evades the implausibility results of [BP13, GGHW14] at a fundamental level, and yields several applications which we discuss shortly.

---

<sup>1</sup>The idealized adversary model considered in [BR14, BGK<sup>+</sup>14] is a generic model for multilinear maps [GGH13a, CLT13].

**On non-uniformity.** Often, because auxiliary input can also capture non-uniformity, the issues of auxiliary input and non-uniformity are treated jointly in definitions. However, since we are introducing nontrivial constraints on auxiliary inputs, we deal with non-uniformity separately. We formulate our definitions to separate out the contributions of auxiliary input (which is a public coin input to the potentially non-uniform sampler), and non-uniform advice. Specifically, we take care to ensure that no secrets can be embedded in the non-uniform advice provided to the sampler, by allowing the non-uniform advice given to the differing-input finding algorithm to depend arbitrarily on the non-uniform advice given to the sampler. Thus, in particular, the non-uniform advice given to the differing-input finding algorithm can contain all secrets present in the non-uniform advice given to the sampler.

**Applications of public-coin diO.** While the public-coin limitation allows us to give a general definition that avoids all known techniques for proving impossibility, one may wonder whether this limitation also rules out known applications of diO. Indeed, at first glance, the situation may seem quite problematic, since an auxiliary input is typically used to capture the partial progress of a security reduction, which will almost always contain various secrets that must be kept from the adversary. Indeed, existing security proofs for applications of diO [ABG<sup>+</sup>13, BCP14] proceed along these lines, and therefore do not carry over for public-coin diO.

In order to make use of public-coin diO, we need to ensure that a stronger property is true in the application scenario and throughout the hybrids in the security proof where the diO property is being used: We need to ensure that whenever the diO property is used, the two programs  $P_1$  and  $P_2$  being considered have the property that it is infeasible to find a differing input between  $P_1$  and  $P_2$  even given all the randomness used in the entire description of the hybrid experiment (except for the random coins of the obfuscation itself). This is sufficient: When using the diO property across two hybrids, we need that the obfuscations are indistinguishable to an all-knowing adversary that is privy to all randomness in these hybrids (except for the random coins of the obfuscation itself). But if the obfuscations are indistinguishable to an all-knowing adversary, then they are also indistinguishable to a more ignorant adversary. Thus, even if some secrets need to be hidden from the adversary in other hybrid experiments, the proof of security can go through.

Despite the flexibility of the above approach, there are still applications of diO in the literature where we do not know how to use public-coin diO in a similar way, because of the nature of the programs being obfuscated. For example, in [BCP14], diO is used to obtain full security for functional encryption by obfuscating programs that deal explicitly with signatures, where a secret verification key of a signature scheme is hidden within obfuscated programs, and given the signing key it is possible to discover differing inputs. Since trapdoors are crucial in this approach, we do not know how to apply public-coin diO. The fact that public-coin diO does not generically replace diO for all applications illustrates the nontrivial nature of our restriction.

Nevertheless, we can use public-coin diO to obtain several interesting applications, as we now detail. Separate from the applications below, building on our work, public-coin diO has been used to replace the need for diO to achieve constant-round concurrent zero knowledge based on obfuscation [PPS15].

**Obfuscating Turing Machines / RAMs with unbounded inputs.** Generally, obfuscation has been studied in the context of circuits [BGI<sup>+</sup>12, GGH<sup>+</sup>13b]. Of course, given a bound on both (1) the input length and (2) the running time, any Turing Machine or RAM program can be converted to an equivalent circuit. However, if either or both of these variables can be unbounded, then obfuscating Turing Machines presents new feasibility questions beyond obfuscating circuits.

Moreover, note that transforming the TM into an equivalent circuit results in a circuit whose size is proportional to the *worst case* running time of the TM. This leads to severe inefficiency since one would have to evaluate a rather large circuit for every input. Indeed, motivated by this issue, Goldwasser

et al. [GKP<sup>+</sup>13a, GKP<sup>+</sup>13b] introduced and studied the notion of *input-specific* run time in the context of several cryptographic primitives such as fully homomorphic encryption [Gen09], functional encryption [SW05, BSW11], and attribute-based encryption [SW05, GPSW06].

Using indistinguishability obfuscation alone, there has recently been exciting progress towards obfuscating Turing Machines directly (i.e., without first transforming it into a circuit). The recent works of Lin and Pass [LP14], Canetti, Holmgren, Jain, and Vaikuntanathan [CHJV14], and Bitansky, Garg, and Telang [BGT14], show how to obfuscate Turing machines or RAM programs directly when both the *input-length* and the overall *space* of the computation is a-priori bounded. More specifically, [LP14, CHJV14, BGT14] first construct garbling schemes for Turing machines (with bounded input-length and space) and use them to obtain obfuscation for Turing machines under same constraints. The size of the obfuscated program in these constructions only depends on the maximum input length *and* space used by the computation (as opposed to worst case running time of the original TM). However, obtaining obfuscation of TMs from garbling schemes introduces its own subtleties due to which current constructions additionally require cryptographic assumptions of sub-exponential hardness.

The recent work of Koppula, Lewko, and Waters [KLW14] presents a novel construction of indistinguishability obfuscation for Turing Machines with *bounded input length* (and unbounded space), based only on iO for circuits and standard assumptions. In other words, the size of the obfuscated TM in the [KLW14] construction is polynomial in the maximum *input length* to be accepted by the obfuscated TM, the description-size of the TM  $M$  to be obfuscated, and the security parameter. (Note that, in particular, it is independent of the maximum space of the computation.) While this is a remarkable result, the dependence upon the maximum input length is still a drawback of this work – a drawback that our work does not encounter. In applications of iO for TMs such as non-black-box simulation [PPS15], it is crucial that there is no a-priori polynomial upper bound on the input length of the obfuscated TM. Furthermore, we note that our construction is significantly simpler than the iO-based construction of [KLW14] and relies only on *polynomial* hardness assumptions; in contrast [KLW14] (as well as [LP14, CHJV14, BGT14]) require sub-exponential hardness assumptions.

In [BCP14, ABG<sup>+</sup>13], diO for circuits, together with SNARKs [BCCT12, BCCT13, BCC<sup>+</sup>14], was shown to imply diO for Turing Machines with *unbounded* inputs, running time, and space complexity (we will refer to this setting as the setting of *general* Turing Machines). However, given the evidence of impossibility for diO, prior to our work, there was no method known to bootstrap from obfuscation for circuits to obfuscation for general Turing Machines based on a plausible and general obfuscation definition. We show that the construction and proofs of [BCP14, ABG<sup>+</sup>13] can be adapted to the setting of public-coin diO: Specifically, we show that public-coin diO for  $NC^1$ , together with fully homomorphic encryption with decryption in  $NC^1$ , and public-coin SNARKs, imply diO for general Turing Machines. We note that our formulation of public-coin SNARK also avoids all known impossibility results for SNARKs and other extractability assumptions [BCPR14].

**Functional Encryption for Turing Machines with unbounded inputs.** We next tackle the problem of (selectively secure) functional encryption [SW05, BSW11] for Turing Machines with unbounded inputs. Here, we are able to show that public-coin diO for general Turing Machines together with standard cryptographic assumptions, implies selectively secure functional encryption for general Turing Machines. As mentioned above, the approach given in [BCP14] achieves full security for functional encryption, but does not adapt to the setting of public-coin diO. The starting point for our scheme is the functional encryption construction given by [ABG<sup>+</sup>13], however in the case of functional encryption, we must make several adjustments to both the construction and the proof of security in order to make use of public-coin diO, and avoid the need for security with respect to general auxiliary inputs. We note that for the case of single-key functional encryption [SS10], the problem of supporting Turing machines and achieving input-specific runtimes was previously introduced and resolved by Goldwasser et al. [GKP<sup>+</sup>13a] under cryptographic assumptions that are incomparable to our work,

but nevertheless, subject to the same criticism as the existence of diO.

Functional encryption is strict strengthening of many cryptographic notions including garbling schemes [Yao82, FKN94, BHR12] (also known as randomized encoding of functions [IK00, AIK06]). Thus, our results for functional encryption imply results for garbling schemes (as well as other notions that are implied by functional encryption) under the public-coin diO assumptions of only *polynomial* hardness. In particular, this applies to several applications of garbling schemes discussed in recent works of [LP14, CHJV14, BGT14] (under incomparable assumptions). We refer the reader to [App11] for a survey of applications of garbling schemes in different areas of cryptography.

**OTHER RELATED WORKS.** Another general and plausible notion of obfuscation that strengthens iO is *virtual gray box* (VGB) obfuscation [BC14, BCKP14]. While conceptually appealing, this notion does not seem as useful as diO for natural applications.

As briefly discussed above, current iO obfuscation candidates can be backed by security proofs in a generic multilinear model [BR14, BGK<sup>+</sup>14]. One can draw an analogy between the broad challenge addressed in the present work and earlier works on instantiating random oracles. Similarly to the practical use of the random oracle model [BR93], provable constructions in the generic multilinear model can give rise to heuristic real-world constructions by plugging in multilinear map candidates such as those from [GGH13a, CLT13]. This may be a reasonable heuristic leap of faith in the context of concrete natural applications. However, similarly to the negative results on instantiating random oracles [CGH04], this methodology is provably not sound in general. Thus, one is left with the challenge of formulating a *succinct* and *plausible* assumption that can be satisfied by an explicit random oracle instantiation and suffices for a wide array of applications. Despite some partial progress (e.g., [Can97, BHK13]), this challenge is still quite far from being fully met.

## 2 Our Definitions

**Notation.** We denote by  $\mathbb{N}$  the set of all natural numbers, and use  $n \in \mathbb{N}$  to denote the security parameter. An efficient non-uniform algorithm  $A$  is denoted by a family of circuits  $A = \{A_n\}_{n \in \mathbb{N}}$  and an associated polynomial  $s$  such the size of  $A_n$  is at most  $s(n)$  for all  $n \in \mathbb{N}$ .

We denote by  $\mathcal{C} = \{\mathcal{C}_n\}_{n \in \mathbb{N}}$  a parameterized *collection* of circuits such that  $\mathcal{C}_n$  is the set of all circuits of size at most  $n$ . Likewise, we denote by  $\mathcal{M} = \{\mathcal{M}_n\}_{n \in \mathbb{N}}$  a parameterized collection of *Turing machines* (TM) such that  $\mathcal{M}_n$  is the set of all TMs of size at most  $n$  which halt within polynomial number steps on all inputs. For  $x \in \{0, 1\}^*$ , if  $M$  halts on input  $x$ , we denote by  $\text{steps}(M, x)$  the number of steps  $M$  takes to output  $M(x)$ . Following [ABG<sup>+</sup>13], we adopt the convention that the output  $M(x)$  includes the number of steps  $M$  takes on  $x$ , in addition to the “official” output. When clear from the context, we drop  $n \in \mathbb{N}$  from the notation.

### 2.1 Circuits

We first define the notion of a public-coin differing-inputs sampler.

**Definition 2.1** (Public-Coin Differing-Inputs Sampler for Circuits). An efficient non-uniform sampling algorithm  $\text{Sam} = \{\text{Sam}_n\}$  is called a *public-coin differing-inputs sampler* for the parameterized collection of circuits  $\mathcal{C} = \{\mathcal{C}_n\}$  if the output of  $\text{Sam}_n$  is distributed over  $\mathcal{C}_n \times \mathcal{C}_n$  and for every efficient non-uniform algorithm  $A = \{A_n\}$  there exists a negligible function  $\varepsilon$  such that for all  $n \in \mathbb{N}$ :

$$\Pr_r [C_0(x) \neq C_1(x) : (C_0, C_1) \leftarrow \text{Sam}_n(r), x \leftarrow A_n(r)] \leq \varepsilon(n). \quad \square$$

The definition insists that the sampler and the attacker circuits both receive the same random coins as input. Therefore,  $\text{Sam}$  cannot keep any “secret” from  $A$ . We now define the notion of public-coin

differing-inputs obfuscator. The crucial change from existing diO definitions is that the distinguisher now gets the actual coins of the sampler as the auxiliary input.

**Definition 2.2** (Public-Coin Differing-Inputs Obfuscator for Circuits). A uniform PPT algorithm  $\mathcal{O}$  is a *public-coin differing-inputs obfuscator for the parameterized collection of circuits*  $\mathcal{C} = \{\mathcal{C}_n\}$  if the following requirements hold:

- **Correctness:**  $\forall n, \forall C \in \mathcal{C}_n, \forall x$  we have that  $\Pr[C'(x) = C(x) : C' \leftarrow \mathcal{O}(1^n, C)] = 1$ .
- **Security:** for every public-coin differing-inputs samplers  $\text{Sam} = \{\text{Sam}_n\}$  for the collection  $\mathcal{C}$ , every efficient non-uniform (distinguishing) algorithm  $\mathcal{D} = \{\mathcal{D}_n\}$ , there exists a negligible function  $\varepsilon$  s.t. for all  $n$ :

$$\left| \Pr[\mathcal{D}_n(r, C') = 1 : (C_0, C_1) \leftarrow \text{Sam}_n(r), C' \leftarrow \mathcal{O}(1^n, C_0)] - \Pr[\mathcal{D}_n(r, C') = 1 : (C_0, C_1) \leftarrow \text{Sam}_n(r), C' \leftarrow \mathcal{O}(1^n, C_1)] \right| \leq \varepsilon(n)$$

where the probability is *taken over  $r$  and the coins of  $\mathcal{O}$* . □

## 2.2 Turing machines

We now present our definitions for the case of Turing machines.

**Definition 2.3** (Public-Coin Differing-Inputs Sampler for TMs). An efficient non-uniform sampling algorithm  $\text{Sam} = \{\text{Sam}_n\}$  is called a *public-coin differing-inputs sampler* for the parameterized collection of TMs  $\mathcal{M} = \{\mathcal{M}_n\}$  if the output of  $\text{Sam}_n$  is always a pair of Turing machines  $(M_0, M_1) \in \mathcal{M}_n \times \mathcal{M}_n$  such that  $|M_0| = |M_1|$  and for all efficient non-uniform (attacker) algorithms  $A = \{A_n\}$  there exists a negligible function  $\varepsilon$  such that for all  $n \in \mathbb{N}$ :

$$\Pr_r \left[ \begin{array}{l} M_0(x) \neq M_1(x) \quad \wedge \\ \text{steps}(M_0, x) = \text{steps}(M_1, x) = t \end{array} : \begin{array}{l} (M_0, M_1) \leftarrow \text{Sam}_n(r), \\ (x, 1^t) \leftarrow A_n(r) \end{array} \right] \leq \varepsilon(n). \quad \square$$

**Remark.** By requiring  $A_n$  to output  $1^t$ , we rule out all inputs  $x$  for which  $M_0, M_1$  may take more than polynomial steps.

**Definition 2.4** (Public-Coin Differing-Inputs Obfuscator for TMs). A uniform PPT algorithm  $\mathcal{O}$  is a *public-coin differing-inputs obfuscator for the parameterized collection of TMs*  $\mathcal{M} = \{\mathcal{M}_n\}$  if the following requirements hold:

- **Correctness:**  $\forall n, \forall M \in \mathcal{M}_n, \forall x \in \{0, 1\}^*$  we have  $\Pr[M'(x) = M(x) : M' \leftarrow \mathcal{O}(1^n, M)] = 1$ .
- **Security:** for every public-coin differing-inputs samplers  $\text{Sam} = \{\text{Sam}_n\}$  for the collection  $\mathcal{M}$ , for every efficient non-uniform (distinguishing) algorithm  $\mathcal{D} = \{\mathcal{D}_n\}$ , there exists a negligible function  $\varepsilon$  s.t. for all  $n$ :

$$\left| \Pr[\mathcal{D}_n(r, M') = 1 : (M_0, M_1) \leftarrow \text{Sam}_n(r), M' \leftarrow \mathcal{O}(1^n, M_0)] - \Pr[\mathcal{D}_n(r, M') = 1 : (M_0, M_1) \leftarrow \text{Sam}_n(r), M' \leftarrow \mathcal{O}(1^n, M_1)] \right| \leq \varepsilon(n)$$

where the probability is *taken over  $r$  and the coins of  $\mathcal{O}$* .

- **Succinctness and input-specific running time:** there exists a (global) polynomial  $s'$  such that for all  $n$ , for all  $M \in \mathcal{M}_n$ , for all  $M' \leftarrow \mathcal{O}(1^n, M)$ , and for all  $x \in \{0, 1\}^*$ ,  $\text{steps}(M', x) \leq s'(n, \text{steps}(M, x))$ . □



**Remark.** The size of the obfuscated machine  $M'$  is always bounded by the running time of  $\mathcal{O}$  which is polynomial in  $n$ . More importantly, the size of  $M'$  is independent of the running time of  $M$ . This holds even if we consider TMs which always run in polynomial time. This is because the polynomial bounding the running time of  $\mathcal{O}$  is independent of the collection  $\mathcal{M}$  being obfuscated.

It is easy to obtain a uniform formulation from our current definitions.

### 3 Preliminaries

**Succinct non-interactive arguments.** The universal relation [BG02] is defined to be the set  $\mathcal{R}_{\mathcal{U}}$  of instance-witness pairs  $(y, w)$  such that  $y$  is of the form  $(M, x, t)$ ,  $|w| \leq t$ , and  $M$  is a TM which accepts  $(x, w)$  within  $t$  steps where  $t$  is an arbitrary number in  $\mathbb{N}$ . For constant  $c \in \mathbb{N}$ , we define  $\mathcal{R}_c$  to the subset of  $\mathcal{R}_{\mathcal{U}}$  consisting of those pairs  $\{(y, w) = ((M, x, t), w)\}$  for which  $t \leq |x|^c$ . The language corresponding to a relation  $\mathcal{R} \subseteq \mathcal{R}_{\mathcal{U}}$  will be denoted by  $L_{\mathcal{R}}$ .

We recall the definitions of *succinct non-interactive arguments* (SNARG) and *succinct non-interactive arguments of knowledge* (SNARK) below. We require that these systems be *publicly verifiable* and work in the common *random* string model where any uniformly random string of sufficient length can act as the CRS. Our definition follows the standard formulations [BCCT12, BCP14].

**Definition 3.1** (SNARG). A pair of algorithms  $(P, V)$  is a (publicly verifiable) SNARG for a relation  $\mathcal{R} \subseteq \mathcal{R}_{\mathcal{U}}$  in the common *random* string model if there exist polynomials  $p, q, \ell$  (independent of  $\mathcal{R}$ ) such that the following conditions are satisfied:

- **Completeness:**  $\forall (y, w) \in \mathcal{R}$ ,  $\Pr [V(\text{crs}, y, \pi) = 1 : \text{crs} \leftarrow \{0, 1\}^{\text{poly}(n)}, \pi \leftarrow P(\text{crs}, y, w)] = 1$ , and for every  $\text{crs}$ ,  $P(\text{crs}, y, w)$  halts within  $p(n, |y|, t)$  where  $y = (M, x, t)$ .
- **Succinctness:** for every  $(\text{crs}, y, w) \in \{0, 1\}^{\text{poly}(n)} \times \mathcal{R}$  the size of  $\pi \leftarrow P(\text{crs}, y, w)$  is bounded by  $\ell(n, \log t)$  and the running time of  $V(\text{crs}, y, \pi)$  is bounded by  $q(n + |y|) = q(n + |M| + |x| + \log t)$ .
- **Adaptive soundness:** for every polynomial-size prover  $P^* = \{P_n^*\}$ , there exists a negligible function  $\varepsilon$  such that for all  $n$ :

$$\Pr \left[ V(\text{crs}, y, \pi) = 1 \wedge y \notin L_{\mathcal{R}} : \text{crs} \leftarrow \{0, 1\}^{\text{poly}(n)}, (y, \pi) \leftarrow P_n^*(\text{crs}) \right] \leq \varepsilon(n). \quad \square$$

Observe that the soundness condition is not required to hold with respect to *common auxiliary input* of any kind. This notion suffices for the restricted cases where obfuscation size grows with the maximum supported input length (a.k.a. bounded-input case). To deal with inputs of unbounded polynomial length, we need the following stronger notion.

**Definition 3.2** (SNARK). A pair of algorithms  $(P, V)$  is a (publicly verifiable) SNARK for the relation  $\mathcal{R} \subseteq \mathcal{R}_{\mathcal{U}}$  in the common *random* string model if it satisfies the *completeness* and *succinctness* conditions of definition 3.1 (above) and the following argument-of-knowledge property:

- **Adaptive argument of knowledge:** for every polynomial-size prover  $P^* = \{P_n^*\}$ , there exists a polynomial-size extractor  $\mathcal{E}_{P^*} = \{\mathcal{E}_n\}$  and a negligible function  $\varepsilon$  such that for all  $n$ :

$$\Pr \left[ (V(\text{crs}, y, \pi) = 1) \wedge ((y, w) \notin \mathcal{R}) : \left\{ \begin{array}{ll} \text{crs} \leftarrow \{0, 1\}^{\text{poly}(n)}, & z \leftarrow \{0, 1\}^{\text{poly}(n)}, \\ (y, \pi) \leftarrow P_n^*(\text{crs}, z), & (y, w) \leftarrow \mathcal{E}_n(\text{crs}, z) \end{array} \right\} \right] \leq \varepsilon(n). \quad \square$$

Observe that in this definition a uniformly distributed auxiliary input  $z$  is allowed. As noted in [BCCT12], none of the existing implausibility results regarding the existence of SNARKs or extractable one-way/collision-resistant-hash functions apply to the case where auxiliary input is a uniformly random string. A candidate construction (and perhaps the only one at this time) for such SNARKs is Micali’s CS proof system [Mic94].

We remark that the above definition requires the extraction to succeed with probability almost 1. Our results do not require this strong form of extraction, and work with a weaker notion as well where extraction probability is only required to be negligibly close to the success probability of the cheating prover.

We shall also use fully homomorphic encryption and non-interactive strong witness indistinguishable proofs, e.g., [FLS99]. We discuss them in appendix A.

## 4 Bootstrapping Obfuscation from $\text{NC}^1$ to Turing Machines

In this section, we show that given a public-coin differing-inputs obfuscator for the class  $\text{NC}^1$ , we can construct a public-coin differing-inputs obfuscator for the parameterized collection  $\mathcal{M}_n$  of all polynomial-time TMs. The construction is a slightly simplified version of [ABG<sup>+</sup>13] where we get rid of the hash functions. We shall prove the following theorem.

**Theorem 4.1.** *If there exists a public-coin differing-inputs obfuscator for circuits in the class  $\text{NC}^1$ , a fully homomorphic encryption scheme with decryption in  $\text{NC}^1$ , and a public-coin SNARK for  $\mathcal{R}_{\mathcal{U}}$  in the common random string model, there exists a public-coin differing-inputs obfuscator for the class of all polynomial-time Turing machines accepting inputs of unbounded polynomial length.*

We first present the construction, and then prove the theorem. Let  $\mathcal{M} = \{\mathcal{M}_n\}_{n \in \mathbb{N}}$  be a parameterized collection of polynomial-time TMs that accepts inputs of unbounded polynomial length, i.e., there exists a constant  $c \in \mathbb{N}$  such that every  $M \in \mathcal{M}_n$  is of size  $n$ , takes inputs of length at most  $n^c$ , and halts within  $n^c$  steps. We adopt the convention that  $c$  is included in the description of  $M$ . Let  $\text{FHE} = (\text{Gen}, \text{Enc}, \text{Dec}, \text{Eval})$  be a fully homomorphic encryption scheme with decryption in  $\text{NC}^1$  and  $\Pi = (P, V)$  be a SNARK for the relation  $\mathcal{R}_{\mathcal{U}}$  defined earlier. The description of our obfuscator for  $\mathcal{M}$ , and its evaluation algorithm, are as follows.

**Obfuscator  $\mathcal{O}(1^n, M \in \mathcal{M}_n)$ :** By convention, description of  $M$  includes a constant  $c$  bounding the running time of  $M$  on all inputs by  $n^c$ . Let  $U_n$  be an *oblivious* universal TM which on input the description of a TM  $B$ , and a string  $x$  executes  $B$  on  $x$  for no more than  $n^c$  steps. The obfuscator proceeds in the following steps:

1. Generate two FHE public-keys  $(pk_1, sk_1) \leftarrow \text{Gen}(1^n; u_1)$  and  $(pk_2, sk_2) \leftarrow \text{Gen}(1^n; u_2)$ .
2. Encrypt  $M$  under both FHE public-keys:  $g_1 \leftarrow \text{Enc}_{pk_1}(M; v_1)$  and  $g_2 \leftarrow \text{Enc}_{pk_2}(M; v_2)$ . Here  $M$  is assumed to be encoded as a bit string of length  $n$  for use by the universal TM  $U_n$ .
3. Uniformly sample  $\text{crs} \leftarrow \{0, 1\}^{\text{poly}(n)}$  of sufficient length (for the SNARK  $\Pi$ ).
4. Generate an obfuscation of the  $\text{NC}^1$ -program  $P1_{sk_1, g_1, g_2}^{\text{crs}}$  given in figure 1:

$$P' \leftarrow \mathcal{O}_{\text{NC}^1}(1^n, P1_{sk_1, g_1, g_2}^{\text{crs}}).$$

5. Output  $M' = (P', \text{crs}, pk_1, pk_2, g_1, g_2)$ .

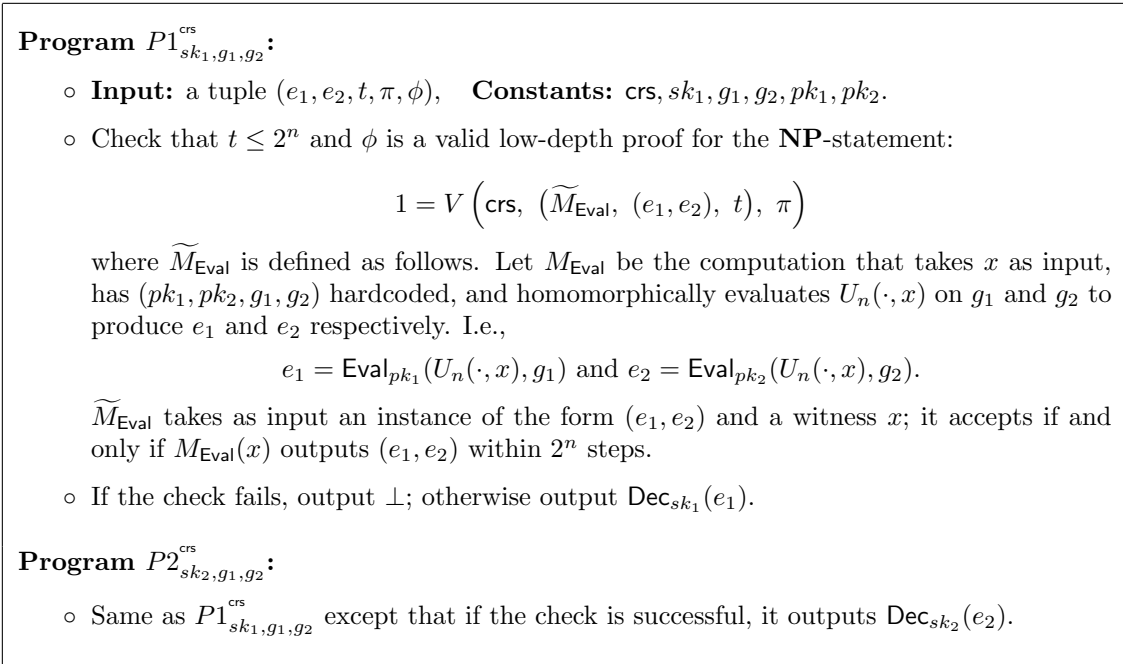


Figure 1: The programs  $P1$  and  $P2$

**Evaluation of  $M'$ :** Evaluate  $M' = (P', crs, pk_1, pk_2, g_1, g_2)$  on input  $x$  as follows:

1. Compute  $(e_1, e_2) = M_{Eval}(x)$ . This takes at most  $n^{2c}$  steps. See fig. 1 for  $M_{Eval}$ .
2. Compute a SNARK proof  $\pi$  using  $x$  as the witness and  $t = n^{4c}$  as the time-bound:

$$\pi \leftarrow P\left(crs, (\widetilde{M}_{Eval}, (e_1, e_2), t), x\right)$$

3. Compute a low-depth proof  $\phi$  for the **NP**-statement  $1 = V(crs, (\widetilde{M}_{Eval}, (e_1, e_2), t), \pi)$ . This can be done by providing the entire computation of  $V$  on these inputs.
4. Execute  $P'(e_1, e_2, t, \pi, \phi)$  and output the result.

The construction is now analyzed in the proof below. We denote by  $a||b$  the concatenation of two bit strings  $a$  and  $b$ .

**Proof of theorem 4.1.** The correctness and succinctness of this construction are relatively straightforward to verify, and in particular, closely follow the analyses in [ABG<sup>+</sup>13, BCP14]. We analyze its security.

**Security.** Fix any public-coin differing-inputs sampler  $\text{Sam} = \{\text{Sam}_n\}$  for the family  $\mathcal{M}$  and any efficient distinguisher  $\mathcal{D} = \{\mathcal{D}_n\}$ . For a bit  $b$ , let  $\mathcal{X}_n(b)$  denote the output of the following experiment over a random choice of  $r$  and the coins of  $\mathcal{O}$ :

$$\mathcal{X}_n(b) := \{(M_0, M_1) \leftarrow \text{Sam}_n(r), M' \leftarrow \mathcal{O}(1^n, M_b), \text{output } \mathcal{D}_n(r, M')\}$$

We need to show that  $\mathcal{X}_n(0) \approx_c \mathcal{X}_n(1)$ . Consider the following sequence of hybrid experiments.

- $H_0$ : This hybrid corresponds to an honest sampling of  $\mathcal{X}_n(0)$ . In this case,  $M'$  creates two FHE encryptions of  $M_0$ , namely  $g_1$  and  $g_2$  (where  $M_0$  is the first output of  $\text{Sam}_n$ ).

- $H_1$ : Same as  $H_0$  except that the second FHE ciphertext is now generated as an encryption of  $M_1$ , i.e.,  $g_2 = \text{Enc}_{pk_2}(M_1)$  (where  $M_1$  is the second output of  $\text{Sam}_n$ ).
- $H_2$ : Same as  $H_1$  except that the obfuscated program  $P'$  is now generated as an obfuscation of  $P2_{sk_2, g_1, g_2}^{\text{crs}}$  which decrypts the *second* ciphertext using  $sk_2$ , i.e.,  $P' \leftarrow \mathcal{O}_{\text{NC}^1}(1^n, P2_{sk_2, g_1, g_2}^{\text{crs}})$ .
- $H_3$ : Same as  $H_2$  except that the *first* FHE ciphertext  $g_1$  is now also generated as an encryption of  $M_1$ , i.e.,  $g_1 \leftarrow \text{Enc}_{pk_1}(M_1)$ .
- $H_4$ : Same as  $H_3$  except that the obfuscated program  $P'$  is once again generated as an obfuscation of  $P1_{sk_1, g_1, g_2}^{\text{crs}}$ , i.e.,  $P' \leftarrow \mathcal{O}_{\text{NC}^1}(1^n, P1_{sk_1, g_1, g_2}^{\text{crs}})$ . Note that  $H_4$  is identical to  $\mathcal{X}_n(1)$ .

We now prove that each hybrid in this sequence is indistinguishable from the previous one.

**Step 1:**  $H_0 \approx_c H_1$ . This follows from the IND-CPA security of FHE. Formally, consider an adversary  $A_{\text{FHE}}$ , who receives a challenge public-key  $pk$ , then samples  $(M_0, M_1) \leftarrow \text{Sam}_n(r)$  for a random  $r$ , and receives an honestly generated ciphertext  $g$  to either  $M_0$  or  $M_1$  under  $pk$ .  $A_{\text{FHE}}$  then generates an obfuscation of  $M_0$  following the instruction of  $\mathcal{O}$  except that it sets  $pk_2 = pk$  and  $g_2 = g$ . Note that all instructions of  $\mathcal{O}$  can indeed be performed efficiently knowing only  $(pk_2, g_2)$ . Let  $M'$  denote the resulting obfuscation which includes an  $\text{NC}^1$ -obfuscation  $P'$  of program  $P1_{sk_1, g_1, g_2=g}^{\text{crs}}$ .  $A_{\text{FHE}}$  outputs whatever  $\mathcal{D}_n(r, M')$  outputs. The output of  $A_{\text{FHE}}$  is distributed identically to that of  $H_b$  when  $g$  is an encryption of  $M_b$  where  $b \in \{0, 1\}$ . Because  $\text{Sam}_n$  and  $\mathcal{D}_n$  are of polynomial-size, it follows that  $A_{\text{FHE}}$  is a polynomial-size circuit violating IND-CPA security of FHE unless  $H_0 \approx_c H_1$ .

**Step 2:**  $H_1 \approx_c H_2$ . We use the soundness of SNARK and diO-security of  $\mathcal{O}_{\text{NC}^1}$  to argue that  $H_1 \approx_c H_2$ . Suppose that  $H_1$  and  $H_2$  are not computationally indistinguishable. We use  $\text{Sam}_n$  and  $\mathcal{D}_n$  to construct a public-coin differing-inputs sampler  $\text{Sam}_n^{\text{NC}^1}$  along with a distinguisher  $\mathcal{D}_n^{\text{NC}^1}$  such that  $\text{Sam}_n^{\text{NC}^1}$  outputs circuits in  $\text{NC}^1$  and  $\mathcal{D}_n^{\text{NC}^1}$  violates the security of  $\mathcal{O}_{\text{NC}^1}$  w.r.t.  $\text{Sam}_n^{\text{NC}^1}$ . We start by constructing  $\text{Sam}_n^{\text{NC}^1}$ .

**Sampler  $\text{Sam}_n^{\text{NC}^1}(\rho)$ .**

1. Parse  $\rho$  as  $\rho = (r, \rho_1, u_1, u_2, v_1, v_2)$ .
2. Sample  $(M_0, M_1) \leftarrow \text{Sam}_n(r)$ . // comment: this is the given sampler.
3. Set  $\text{crs} = \rho_1$ ,  $(pk_1, sk_1) \leftarrow \text{Gen}(1^n; u_1)$ ,  $(pk_2, sk_2) \leftarrow \text{Gen}(1^n; u_2)$ .
4. Set  $g_1 \leftarrow \text{Enc}_{pk_1}(M_0; v_1)$  and  $g_2 \leftarrow \text{Enc}_{pk_2}(M_1; v_2)$ .
5. Output  $(C_0, C_1)$  corresponding to the programs  $(P1_{sk_1, g_1, g_2}^{\text{crs}}, P2_{sk_2, g_1, g_2}^{\text{crs}})$ .

Note that input to the circuits  $C_0, C_1$  above are of the form  $m = (e_1, e_2, t, \pi, \phi)$ .

**Claim 4.2.**  $\forall n \in \mathbb{N}$   $\text{Sam}_n^{\text{NC}^1}$  is a public-coin differing-inputs sampler for a family  $\mathcal{C} \in \text{NC}^1$ .

**Proof.** We have to show that every non-uniform PPT attacker  $\{A_n^{\text{NC}^1}\}$  fails to find a differing-input for circuits sampled by  $\text{Sam}_n^{\text{NC}^1}$ . Given an attacker  $A_n^{\text{NC}^1}$  which succeeds against our sampler, we construct an attacker  $A_n$  which will succeed against the given sampler  $\text{Sam}_n$ . We shall rely on the soundness of SNARK to prove this.

Formally, suppose that the claim is false, and there exists a polynomial-size attacker family  $\{A_n^{\text{NC}^1}\}$ , a polynomial  $p$ , and infinitely many  $n$  s.t.

$$\Pr_{\rho} \left[ C_0(x) \neq C_1(x) : (C_0, C_1) \leftarrow \text{Sam}_n^{\text{NC}^1}(\rho), x \leftarrow A_n^{\text{NC}^1}(\rho) \right] \geq 1/p(n).$$

We start by defining a prover family which receives a uniformly random auxiliary input, denoted by  $z$ , and then use it to define an attacker  $\tilde{A}_n$  which also receives a uniform auxiliary input. Later on, this auxiliary input will be completely removed from  $\tilde{A}_n$ .

**Prover  $P_n^*(\text{crs}, z)$ :** String  $z$  is of the form  $(r, u_1, u_2, v_1, v_2)$  and the circuit has adversary  $A_n^{\text{NC}^1}$  hardcoded in it. The circuit proceeds as follows:

1. Define  $\rho := (r, \text{crs}, u_1, u_2, v_1, v_2)$  using  $z$  and  $\text{crs}$ .
2. Compute  $m \leftarrow A_n^{\text{NC}^1}(\rho)$  which is of the form  $m = (e_1, e_2, t, \pi, \phi)$ . Recall that  $A_n^{\text{NC}^1}(\rho)$  defines a TM  $\tilde{M}_{\text{Eval}}$  and  $\pi$  is a SNARK proof for the statement  $y := (\tilde{M}_{\text{Eval}}, (e_1, e_2), t)$ .
3. Output  $(y, \pi)$ .

Let  $\{\mathcal{E}_n^*\}$  be a family of extractor circuits w.r.t. the prover family  $\{P_n^*\}$  defined above. Now we define the following attacker circuit  $\tilde{A}_n$  which receives a uniformly random auxiliary input  $z^*$  and outputs a differing input for the given sampler  $\text{Sam}_n$ . Later we will choose an appropriate  $z^*$  and hardcode it as part of the circuit description to achieve an attacker circuit without auxiliary input.

**Circuit  $\tilde{A}_n(r, z^*)$ :** String  $z^*$  is of the form  $(\rho_1, u_1, u_2, v_1, v_2)$  and extractor circuit  $\mathcal{E}_n^*$  is hardcoded in this circuit. The circuit computes as follows:

1. Define  $\text{crs} = \rho_1$  and  $z = (r, u_1, u_2, v_1, v_2)$  using  $r$  and  $z^*$ .
2. Compute  $(y, w) \leftarrow \mathcal{E}_n^*(\text{crs}, z)$  where  $y$  is of the form  $y := (\tilde{M}_{\text{Eval}}, (e_1, e_2), t)$ .
3. Output  $x = w$  as the differing input.

For any given  $r, z^*$ , the concatenation  $\rho = r \| z^*$  is of the form  $(r, \rho_1, u_1, u_2, v_1, v_2)$ , and defines a valid random string for  $A_n^{\text{NC}^1}$ . We say that a fixed string  $z^*$  is **good** if, the success probability of  $A_n^{\text{NC}^1}(\rho)$  is at least  $\frac{1}{2p}$  over the choice of  $r$ . Formally, a string  $z^*$  is **good** if for a randomly chosen  $r$ , defining the tape  $\rho = r \| z^*$ , the probability that  $A_n^{\text{NC}^1}(\rho)$  outputs  $m$  such that  $C_0(m) \neq C_1(m)$  where  $(C_0, C_1) \leftarrow \text{Sam}_n^{\text{NC}^1}(\rho)$  is at least  $1/2p$ . By simple averaging, at least  $\frac{1}{2p}$  fraction of  $z^*$  are **good**.

Now let us define **sound** strings. Roughly speaking, we say that  $z^*$  is **sound** if the probability that the output of  $A_n^{\text{NC}^1}(\rho)$  contains a valid proof  $\pi$  but the output of the extractor (in step 2 above) is not a valid witness, is less than  $1/4p$ .

Formally, we say that a fixed string  $z^* = (\rho_1, u_1, u_2, v_1, v_2)$  is **sound** if for a randomly chosen  $r$ , defining the tape  $\rho = r \| z^*$ , the probability of the following event, taken over  $r$ , is at most  $1/4p$ : the output of  $\mathcal{E}_n^*(\text{crs}, z)$  (in step 2 of  $\tilde{A}_n(r, z^*)$ ) is  $(y, w)$  and output of  $A_n^{\text{NC}^1}(\rho)$  is  $m = (e_1, e_2, t, \pi, \phi)$  such that  $V$  accepts the proof  $\pi$  for the statement  $y$  but  $w$  is not a valid witness, i.e.  $(y, w) \notin \mathcal{R}_{2c}$ .

A randomly chosen  $z^*$  contains a uniformly distributed  $\text{crs}$  string; therefore, it follows that at least  $1 - \epsilon'$  fraction of  $z^*$  are **sound** where  $\epsilon'$  is the soundness error of SNARK.

Therefore, at least  $\frac{1}{2p} - \epsilon' \geq \frac{1}{4p}$  fraction of  $z^*$  are both **good** and **sound**. Fix such a  $z^*$ . By definition of **good** it follows that w.r.t. this  $z^*$ , at least  $1/2p$  fraction of inputs  $r$  are such that  $A_n^{\text{NC}^1}(r \| z^*)$  outputs a differing-input for  $C_0, C_1$ . Further, by definition of **sound**, at most a  $1/4p$  fraction of such inputs  $r$  are such that the extractor  $\mathcal{E}_n^*$  (in step 2 above) will not output a valid witness  $w$ . Therefore, at least  $\frac{1}{2p} - \frac{1}{4p} \geq \frac{1}{4p}$  of inputs  $r$  result in a differing-input where the extractor's output is a valid witness. We call such inputs **nice**.

By construction, for nice  $r$ , we have that:

$$C_0(m) \neq C_1(m) \implies P1_{sk_1, g_1, g_2}^{\text{crs}=\rho_1}(m) \neq P2_{sk_2, g_1, g_2}^{\text{crs}=\rho_1}(m) \implies M_0(x) \neq M_1(x)$$

where  $x = w$  is the differing-input output by  $\tilde{A}_n$ , and the last implication follows because  $x$  is a valid witness, i.e. if  $m$  contains ciphertexts  $e_1, e_2$  then the values in these ciphertexts will indeed be  $M_0(x)$  and  $M_1(x)$  respectively. Here  $M_0, M_1$  are the TMs sampled in (step 2 of) the execution of  $\text{Sam}_n^{\text{NC}^1}(r||z^*)$ . We now observe that, by construction,  $(M_0, M_1)$  are also the output of  $\text{Sam}_n(r)$ . Therefore, the output of  $\tilde{A}_n$  outputs a differing input  $x$  for the outputs of  $\text{Sam}_n$  whenever  $z^*$  is good and sound and  $r$  is nice.

To have a deterministic attacker  $A_n$  which on input  $r$  outputs a differing-input  $x$ , we choose a  $z^*$  that is good and sound, and hardcode it in the description of  $\tilde{A}_n$ . It follows that, since the fraction of nice strings  $r$  is at least  $1/4p$ ,  $A_n$  violates the public-coin differing-input property of  $\text{Sam}_n$  with noticeable property. The proof is completed by observing that circuits output by  $\text{Sam}_n^{\text{NC}^1}$  are indeed in the complexity class  $\text{NC}^1$ . ■

We now present distinguisher  $\mathcal{D}_n^{\text{NC}^1}$  which violates the security of  $\mathcal{O}_{\text{NC}^1}$  w.r.t. sampler  $\text{Sam}_n^{\text{NC}^1}$ .

**Distinguisher**  $\mathcal{D}_n^{\text{NC}^1}(\rho, C')$ . The input consists a string  $\rho$  and an obfuscated circuit  $C'$ .  $C'$  is an obfuscation of either  $C_0$  or  $C_1$  which are output by  $\text{Sam}_n^{\text{NC}^1}(\rho)$ . The distinguisher attempts to create a valid obfuscation  $M'$  of the TM implicitly present in  $C'$ . Since entire string  $\rho$  is available, it can be efficiently done as follows.

1. Parse  $\rho$  as  $\rho = (r, \rho_1, u_1, u_2, v_1, v_2)$ , and set  $\text{crs} = \rho_1$ ,  $(pk_1, sk_1) \leftarrow \text{Gen}(1^n; u_1)$ ,  $(pk_2, sk_2) \leftarrow \text{Gen}(1^n; u_2)$ ,  $g_1 \leftarrow \text{Enc}_{pk_1}(M_0; v_1)$  and  $g_2 \leftarrow \text{Enc}_{pk_2}(M_1; v_2)$ , where  $(M_0, M_1) \leftarrow \text{Sam}(1^n; r)$ .
2. Define  $M' = (C', \text{crs}, pk_1, pk_2, g_1, g_2)$ , and output whatever  $\mathcal{D}_n(r, M')$  outputs. (Recall that  $\mathcal{D}_n$  is the given distinguisher).

By construction of  $\text{Sam}_n^{\text{NC}^1}$ , we can see that if  $C'$  is a correctly generated obfuscation of  $C_b$ , then  $M'$  is distributed as in hybrid  $H_{b+1}$ . It follows that if outputs of  $H_1$  and  $H_2$  are distinguishable then  $\mathcal{D}_n^{\text{NC}^1}$  is a good distinguisher against  $\mathcal{O}_{\text{NC}^1}$  w.r.t.  $\text{Sam}_n^{\text{NC}^1}$ .

**Final step:**  $H_2 \approx_c H_3$  and  $H_3 \approx_c H_4$ . Proof for the claim  $H_2 \approx_c H_3$  is nearly identical to step 1. The proof for  $H_3 \approx_c H_4$  is nearly identical to step 2. We omit the details. ■

## 5 Functional Encryption for Turing Machines

In this section, we shall construct a functional encryption scheme. The scheme can encrypt messages of arbitrary polynomial length. The secret key  $SK_M$  is given corresponding to a TM  $\mathcal{M}$  of polynomial size which can accept inputs of arbitrary polynomial length and halts in polynomial time. The holder of  $SK_M$  can learn the value of  $M(x)$  given an encryption of  $x$ . The size of the public-parameters of our scheme is polynomial in the security parameter, and the size of secret-keys, say  $SK_M$ , is polynomial in the security parameter,  $|M|$ , and  $\log t$  where  $t$  is an arbitrary polynomial bounding the worst case running time of  $M$ .

We assume familiarity with the definition of functional encryption (FE) schemes. Our scheme will satisfy indistinguishability based notion of security in the selective model of security which we recall here. In this model, we consider the following experiment  $\text{Expt}$  between an attacker  $A$  and a challenger. The experiment takes a bit  $b$  as input, and proceeds as follows:

**Init**  $A$  sends two messages  $x_0^*, x_1^*$  such that  $|x_0^*| = |x_1^*|$ .

**Phase 1** The challenger samples  $(\text{pp}, \text{msk}) \leftarrow \text{F.Setup}(1^n)$  and sends  $\text{pp}$  to  $A$ .

**Phase 2**  $A$  adaptively asks polynomially secret-key queries where in each query it sends the description of a TM  $M \in \mathcal{M}$  such that  $M(x_1^*) = M(x_2^*)$ .<sup>2</sup> The challenger responds with  $SK_M \leftarrow \text{F.KeyGen}(\text{pp}, \text{msk}, M)$ .

**Challenge** The challenger sends an encryption  $e = \text{F.Enc}(\text{pp}, x_b^*)$ .

**Phase 3** Phase 2 is repeated.

**Output** The output  $A$  is the output of the experiment, which is a bit without loss of generality.

The scheme is said to be selectively secure if  $\text{Adv}_A$  is negligible in  $n$  where we define  $\text{Adv}_A := |\Pr[\text{Expt}(0) = 1] - \Pr[\text{Expt}(1) = 1]|$ .

**Our construction.** Let  $\mathcal{O}$  be a public-coin differing-inputs obfuscation for the class of polynomial-size and polynomial-time TMs taking inputs of arbitrary polynomial length. Let  $\Pi = (\text{CRSGen}, P, V)$  be a statistically sound, non-interactive, *strong* witness-indistinguishable proof system for  $\mathbf{NP}$  where  $\text{CRSGen}$  simply outputs its own random coins—therefore, we are in the common *random* string model where any random string of sufficient length can act as the CRS. Let  $\mathcal{H} = \{\mathcal{H}_n\}$  be a family of collision-resistant hash functions such that every  $h \in \mathcal{H}_n$  maps strings from  $\{0, 1\}^*$  to  $\{0, 1\}^n$ .

Let  $\text{PKE} = (\text{Gen}, \text{Enc}, \text{Dec})$  be an ordinary, semantically secure public-key encryption scheme, and  $\text{com}$  be a statistically binding commitment scheme.<sup>3</sup> We assume that  $\text{PKE}$  (resp.,  $\text{com}$ ) encrypts (resp., commits) to a string of unbounded polynomial length by individually encrypting (resp., committing) to each bit. We assume w.l.o.g. that  $\text{PKE}$  (resp.,  $\text{com}$ ) uses randomness of length  $n$  to encrypt (resp., commit) to a single bit (and therefore  $sn$  random bits for a string of length  $s$  will be needed).

The algorithms of our functional encryption scheme are as follows. Recall that  $a\|b$  denotes the concatenation of two bit strings  $a$  and  $b$ .

- $\text{F.Setup}(1^n)$ : Generate  $(pk_1, sk_1) \leftarrow \text{Gen}(1^n)$ ,  $(pk_2, sk_2) \leftarrow \text{Gen}(1^n)$ , and  $(pk_3, sk_3) \leftarrow \text{Gen}(1^n)$ . Generate two commitments  $\alpha_1 = \text{com}(0^n; u_1)$  and  $\alpha_2 = \text{com}(0^n; u_2)$ . Sample  $\text{crs} \leftarrow \text{CRSGen}(1^n)$  and  $h \leftarrow \mathcal{H}_n$ . Output  $(\text{pp}, \text{msk})$  where:

$$\text{pp} := (pk_1, pk_2, pk_3, \alpha_1, \alpha_2, \text{crs}, h), \quad \text{msk} := sk_1.$$

- $\text{F.Enc}(\text{pp}, x)$ : On input a message  $x \in \{0, 1\}^*$  of arbitrary polynomial length, generate two ciphertexts  $c_1 = \text{Enc}_{pk_1}(x; r_1)$  and  $c_2 = \text{Enc}_{pk_2}(x; r_2)$ . Define string  $a := x\|r_1\|0^{n^2}\|x\|r_2\|0^{n^2}$  and encrypt it under the third public-key to get ciphertext  $c_3 = \text{Enc}_{pk_3}(a; r_3)$ . Finally, compute a proof  $\pi$  for the statement that  $y \in L_{\text{FE}}$  using  $w = (a, r_3)$  as the witness where  $y = (c_1, c_2, c_3, pk_1, pk_2, pk_3, \alpha_1, \alpha_2)$ : i.e.,  $\pi \leftarrow P(\text{crs}, y, w)$ .<sup>4</sup> Here  $L_{\text{FE}}$  is the language corresponding to the relation  $R_{\text{FE}}$  defined below.

**Relaton**  $R_{\text{FE}}$ :

Instance:  $y' = (c'_1, c'_2, c'_3, pk'_1, pk'_2, pk'_3, \alpha'_1, \alpha'_2)$

Witness:  $w' = (a', r'_3)$  where  $a' = x'_1\|r'_1\|u'_1\|x'_2\|r'_2\|u'_2$

$R_{\text{FE}}(y', w') = 1$  if and only if the following condition holds:

1.  $c'_3 = \text{Enc}_{pk'_3}(a'; r'_3)$ ; AND

<sup>2</sup>Recall that according to our convention, the output of  $M$  on an input  $x$  includes its running time as well.

<sup>3</sup>We view  $\text{com}$  as a non-interactive algorithm; we can also use two-round schemes where the first message is fixed as part of the public-parameters by the setup algorithm and then  $\text{com}$  is viewed w.r.t. such a fixed message.

<sup>4</sup>Observe that here no a-priori bound is known on  $|x|$ ; any multi-theorem proof system such as [FLS99] is capable of proving statements of unbounded polynomial length.

2. The OR of the following two statements is true:

- (a)  $c'_1, c'_2$  encrypt the same message which is one of  $x'_1$  or  $x'_2$ , i.e.:  
 $(c'_1 = \text{Enc}_{pk'_1}(x'_1; r'_1) \text{ AND } c'_2 = \text{Enc}_{pk'_2}(x'_1; r'_2)); \text{ OR}$   
 $(c'_1 = \text{Enc}_{pk'_1}(x'_2; r'_1) \text{ AND } c'_2 = \text{Enc}_{pk'_2}(x'_2; r'_2));$
- (b)  $c'_1, c'_2$  encrypt  $x'_1, x'_2$  respectively, which may be different but then the hash of one them is committed in  $\alpha'_1, \alpha'_2$ ; i.e.,
  - i.  $(c'_1 = \text{Enc}_{pk'_1}(x'_1; r'_1) \text{ AND } c'_2 = \text{Enc}_{pk'_2}(x'_2; r'_2)); \text{ AND}$
  - ii.  $(\alpha'_1 = \text{com}(h(x'_1); u'_1) \text{ OR } \alpha'_1 = \text{com}(h(x'_2); u'_1)); \text{ AND}$
  - iii.  $(\alpha'_2 = \text{com}(h(x'_1); u'_2) \text{ OR } \alpha'_2 = \text{com}(h(x'_2); u'_2)).$

Proof  $\pi$  is computed for the AND of statements 1 and 2(a) of  $R_{\text{FE}}$ . The algorithm outputs  $e = (c_1, c_2, c_3, \pi)$  as the ciphertext.

- $\text{F.KeyGen}(\text{pp}, \text{msk}, M)$ : The secret-key  $SK_M$  corresponding to a TM  $M$  is an obfuscation of the program  $\text{Prog}_{M, \text{msk}}$ , i.e.,  $SK_M \leftarrow \mathcal{O}(1^n, \text{Prog}_{M, \text{msk}})$ , where  $\text{Prog}_{M, \text{msk}}$  is the following program.

**Program  $\text{Prog}_{M, \text{msk}}$ :**

- Input: a ciphertext  $e$  of the form  $e = (c_1, c_2, c_3, \pi)$ .
  - Constants:  $\text{msk} = sk_1$  and  $\text{pp} = (pk_1, pk_2, pk_3, \alpha_1, \alpha_2, \text{crs}, h)$ .
  - The program checks that  $1 = V(\text{crs}, y, \pi)$  where  $y = (c_1, c_2, c_3, pk_1, pk_2, pk_3, \alpha_1, \alpha_2)$ .
  - If the check fails, output  $\perp$ ; otherwise output  $M(\text{Dec}_{sk_1}(c_1))$ .
- $\text{F.Dec}(SK_M, e)$ : Evaluate the program  $SK_M$  on input  $e$  and output whatever it outputs.

**Theorem 5.1.** *Let  $\mathcal{M}$  be the class of all polynomial-time Turing machines accepting inputs of unbounded polynomial length. If there exists a public-coin differing-inputs obfuscator for the class  $\mathcal{M}$ , a non-interactive zero-knowledge proof system (i.e., with statistical soundness) for  $\mathbf{NP}$  in the common random string model, a public-key encryption scheme, a non-interactive perfectly-binding commitment scheme, and a family of collision-resistant hash functions with publicly samplable index, then there exists a selectively-secure functional encryption scheme with indistinguishability-based security for Turing machines in the class  $\mathcal{M}$ .*

**Proof of theorem 5.1.** The correctness and succinctness of our scheme is easy to verify, and in particular, is similar to analyses in [GGH<sup>+</sup>13b, ABG<sup>+</sup>13]. We shall provide this analysis in the full version. We now analyze the security of this construction. We prove that the scheme satisfies indistinguishability based security for FE in the selective security model. We prove this by considering the following sequence of hybrid experiments:

- Hybrid  $H_0$  : This hybrid is identical to experiment  $\text{Expt}(0)$ . The public-parameters  $\text{pp}$  in Phase 1 are of the form  $\text{pp} := (pk_1, pk_2, pk_3, \alpha_1, \alpha_2, \text{crs}, h)$  where  $\alpha_1 = \text{com}(0^n; u_1)$  and  $\alpha_2 = \text{com}(0^n; u_2)$ .
- Hybrid  $H_1$  : This hybrid is identical to  $H_0$  except that  $\alpha_1$  and  $\alpha_2$  are computed as commitments to  $h(x_0^*)$  and  $h(x_1^*)$  respectively:  $\alpha_1 = \text{com}(h(x_0^*); u_1)$  and  $\alpha_2 = \text{com}(h(x_1^*); u_2)$ . We recall that the challenge ciphertext is of the form  $(c_1, c_2, c_3, \pi)$  where both  $c_1, c_2$  encrypt  $x_0^*$ ,  $c_3$  encrypts  $a_0 = x_0^* \| r_1 \| 0^{n^2} \| x_0^* \| r_2 \| 0^{n^2}$  using randomness  $r_3$ , and  $\pi$  is computed using  $w = (a_0, r_3)$  as the witness. This is identical to how these values were computed in the previous hybrid.
- Hybrid  $H_2$  : Identical to  $H_1$  except that string  $a_0$  is now changed to  $a^* = x_0^* \| r_1 \| u_1 \| x_1^* \| r_2 \| u_2$ . Consequently, ciphertext  $c_3$  is an encryption of  $a^*$  which we denote by  $c_3^*$ . Since  $a^*$  has changed, the witness used in computing the proof  $\pi$  has also changed, and we shall denote the new proof by  $\pi^*$ . The challenge ciphertext is therefore  $(c_1, c_2, c_3^*, \pi^*)$  where both  $c_1, c_2$  still encrypt  $x_0^*$ .



- Hybrid  $H_3$  : Same as  $H_2$  except that  $c_2$  now encrypts  $x_1^*$ . Furthermore,  $\pi^*$  is computed w.r.t. the AND of statements 1 and 2(b) (see the description of  $R_{\text{FE}}$ ). That is, the witness corresponding to condition 2(b.i) will now be  $(x_0^*, r_1, x_1^*, r_2)$ , and for 2(b.ii) and 2(b.iii) it will be  $(x_0^*, u_1)$  and  $(x_1^*, u_2)$  respectively. Note that  $a^*, c_3^*$  and everything else remains the same.
- Hybrid  $H_4$  : Same as  $H_3$  except that the keys are generated differently. The challenger sets  $\text{msk} = sk_2$  and answers the secret-key queries corresponding to a TM  $M$  by obfuscating the following program  $\text{Prog}_{M, \text{msk}}^*$ :

**Program  $\text{Prog}_{M, \text{msk}}^*$** : The program is identical to  $\text{Prog}_{M, \text{msk}}$  except that it decrypts the second ciphertext using  $\text{msk} = sk_2$  if the check succeeds. That is, the input to the program is a ciphertext  $e = (c_1, c_2, c_3, \pi)$ , the constants are  $(\text{msk}, \text{pp})$ . The program outputs  $\perp$  if  $\pi$  is not a valid proof; otherwise it outputs  $M(\text{Dec}_{sk_2}(c_2))$ .

- Hybrid  $H_5$  Same as  $H_4$  except that  $c_1$  is now changed to encrypt  $x_1^*$ . Furthermore,  $\pi^*$  is computed by using the witness corresponding to condition 2(a), i.e., using  $(x_1^*, r_1, r_2)$ .
- Hybrid  $H_6$  : Same as  $H_5$  except that all secret-key queries are now switched back to using  $\text{msk} = sk_1$  and the key for TM  $M$  is an obfuscation of the program  $\text{Prog}_{M, \text{msk}}$ .
- Hybrid  $H_7$ : Same as  $H_6$  except that  $a^*$  is changed to string  $a_1 = x_1^* \| r_1 \| 0^{n^2} \| x_1^* \| r_2 \| 0^{n^2}$ . The witness corresponding to 2(a) does not change, but corresponding to statement in 1 changes (see  $R_{\text{FE}}$ ). Therefore, proof  $\pi$  also changes.
- Hybrid  $H_8$ : Same as  $H_7$  except that  $\alpha_1, \alpha_2$  are switched back to the commitments of  $0^n$ . Observe that  $H_8$  is identical to the experiment  $\text{Expt}(1)$ .

We now prove the indistinguishability of every two consecutive hybrids in this experiment.

**Step 1:**  $H_0 \approx_c H_1$ . This follows from computational hiding of the commitment scheme. Formally, we consider the following adversary  $A_{\text{com}}$ , which internally executes the hybrid  $H_0$  except that it does not generate commitments  $(\alpha_1, \alpha_2)$  on its own. Instead, after receiving values  $(x_1^*, x_2^*)$  during **Init** phase from  $A$ , it sends two sets of strings, namely  $(0^n, 0^n)$  and  $(h(x_1^*), h(x_2^*))$ , to the outside challenger and receives in return two commitments  $(\alpha_1, \alpha_2)$  corresponding to either the first or the second set of strings. It is clear that  $A_{\text{com}}$  is a polynomial time machine, and violates the hiding of **com** unless  $H_0 \approx_c H_1$ .

**Step 2:**  $H_1 \approx_c H_2$ . The proof of this claim relies on the semantic security of PKE and the *strong* witness indistinguishability of the proof system  $\Pi$  for polynomially many statements.<sup>5</sup> Recall that strong WI asserts the following: let  $\mathcal{D}_0$  and  $\mathcal{D}_1$  be distributions which output an instance-witness pair for an **NP**-relation  $\mathbf{R}$  and suppose that the first components of these distributions are computationally indistinguishable, i.e.,  $\{y : (y, w) \leftarrow \mathcal{D}_0(1^n)\} \approx_c \{y : (y, w) \leftarrow \mathcal{D}_1(1^n)\}$ ; then  $\mathcal{X}_0 \approx_c \mathcal{X}_1$  where  $\mathcal{X}_b : \{(crs, y, \pi) : crs \leftarrow \text{CRSGen}(1^n); (y, w) \leftarrow \mathcal{D}_b(1^n); \pi \leftarrow P(crs, y, w)\}$  for  $b \in \{0, 1\}$ . Strong WI for polynomially many statements is implied by any multi-theorem NIZK proof such as [FLS99].

Suppose that  $H_1$  and  $H_2$  can be distinguished with noticeable advantage  $\delta$ . Observe that both distribution internally sample the following values in an identical manner:  $z := (h, pk_1, pk_2, pk_3, c_1, c_2, \alpha_1, \alpha_2)$

<sup>5</sup>Strictly speaking, we only need strong WI w.r.t. a single statement  $y$  of *unbounded polynomial* length. Any multi-theorem NIZK proof system such as [FLS99] generically yields a strong WI proof system for unbounded polynomial length statements.

which is all but  $\text{crs}, c_3$  and  $\pi$ . By simple averaging, there are at least  $\delta/2$  fraction of string  $st$  s.t. the two hybrids can be distinguished with advantage at least  $\delta/2$  when  $z = st$ . Call such a  $z$  **good**. Fix one such  $z$ , and denote the resulting hybrids by  $H_1^{(z)}, H_2^{(z)}$ . Note that the hybrids have inbuilt into them all other values used to sample  $z$  namely:  $(x_0^*, x_1^*)$  received from  $A$ , randomness  $u_1, u_2, r_1, r_2$  for  $(\alpha_1, \alpha_2, c_1, c_2)$  respectively, and  $\text{msk} = sk_1$ .

Define distribution  $\mathcal{D}_0^{(z)}$  as follows: set  $a_0 = (x_0^* \| r_1 \| 0^{n^2} \| x_0^* \| r_2 \| 0^{n^2})$ , compute  $c_3 = \text{Enc}_{pk_3}(a_0; r_3)$ , and let statement  $y = (c_1, c_2, c_3, pk_1, pk_2, pk_3, \alpha_1, \alpha_2)$ , witness  $w_0 = (a_0, r_3)$ ; output  $(y, w_0)$ . Note that  $y$  has identical to  $z$  except that  $h$  has been removed and  $c_3$  has been added. Now define a second distribution  $\mathcal{D}_1^{(z)}$  which is identical to  $\mathcal{D}_0^{(z)}$  except that instead of string  $a_0$ , it uses string  $a^* = (x_0^* \| r_1 \| u_1 \| x_0^* \| r_2 \| u_2)$ , sets  $c_3 = \text{Enc}_{pk_3}(a^*; r_3)$ , and  $w = (a^*, r_3)$ . It follows from the security of the encryption scheme that the distribution of  $y$  sampled by  $\mathcal{D}_0^{(z)}$  is computationally indistinguishable from when it is sampled by  $\mathcal{D}_1^{(z)}$ . Therefore, we must have that  $\mathcal{X}_0 \approx \mathcal{X}_1$  w.r.t. these distributions. We show that this is not the case unless  $H_1^{(z)} \approx_c H_2^{(z)}$ .

Consider an adversary for strong WI who incorporates  $A$  and  $z$  (along with  $sk_1$  and all values for computing  $z$  described above), and receives a challenge  $(\text{crs}, y, \pi)$  distributed according to either  $\mathcal{D}_0^{(z)}$  or  $\mathcal{D}_1^{(z)}$ ; here  $y$  has a component  $c_3$  (and all other parts of  $y$  are identical to the respective parts of  $z \setminus \{h\}$ ). The adversary uses  $\text{crs}$  to completely define  $\text{pp}$  and feeds it to  $A$ ; it uses  $sk_1$  to complete phase 2 and 4, and  $(c_3, \pi)$  to define the challenge ciphertext  $e = (c_1, c_2, c_3)$ . The adversary outputs whatever  $A$  outputs. We observe that the output of this adversary is distributed according to  $H_1^{(z)}$  (resp.,  $H_2^{(z)}$ ) when it receives a tuple from distribution  $\mathcal{X}_0$  (resp.,  $\mathcal{X}_1$ ). A randomly sampled  $z$  is **good** with probability at least  $\delta/2$ , and therefore it follows that with probability at least  $\delta^2/4$  the strong WI property will be violated unless  $\delta$  is negligible.

**Step 3:**  $H_2 \approx_c H_3$ . The proof of this part follows exactly the same ideas as in step 2, and relies on the semantic security of encryption and strong WI property of  $\Pi$ . Roughly speaking, changing  $c_2$  to encrypt  $x_1^*$  results in a computationally indistinguishable distribution over the statement (to be proven by proof  $\pi$ ). Due to this, although the resulting proof  $\pi$  will use a different witness, strong WI guarantees that the joint distribution of statement and proof (present in the challenge ciphertext) remains computationally indistinguishable in these two hybrids. The details are omitted.

**Step 4:**  $H_3 \approx_c H_4$ . This is the key part of our proof where we shall rely on the indistinguishability security of public-coin diO. Suppose that the claim is false and  $A$ 's output in  $H_3$  is noticeably different from its output in  $H_4$ . Suppose that  $A$ 's running time is bounded by a polynomial  $t$  so that there are at most  $t$  secret-key queries it can make in phase 2 and 3 combined. We consider a sequence of  $t$  hybrid experiments between  $H_3$  and  $H_4$  such that hybrid  $H_3^i$ , for  $i \in [t]$  is as follows.

**Hybrid  $H_3^i$**  is identical to  $H_3$  except that it answers the secret-key queries as follows. For  $j \in [t]$ , if  $j \leq i$ , the secret-key corresponding to the  $j$ -th query, denoted  $M_j$ , is an obfuscation of program  $\text{Prog}_{M_j, sk_1}$ ; otherwise, for  $j > i$ , it is an obfuscation of program  $\text{Prog}_{M_j, sk_2}^*$ . We define  $H_3^0$  to be  $H_3$  and observe that  $H_3^t$  is the same as  $H_4$ .

By simple calculation, we see that if  $A$ 's advantage in distinguishing  $H_3$  and  $H_4$  is  $\delta$ , then there exists an  $i \in [t]$  such that  $A$  distinguishes between  $H_3^{i-1}$  and  $H_3^i$  with advantage at least  $\delta/t$ . We show that if  $\delta$  is not negligible than we can use  $A$  to violate the indistinguishability of  $\mathcal{O}$ . To do so, we define a sampling algorithm  $\text{Sam}_A^i$  and a distinguishing algorithm  $\mathcal{D}_A^i$  and prove that  $\text{Sam}_A^i$  is a public-coin

differing-input sampler outputting a pair of differing-input TMs yet  $\mathcal{D}_A^i$  can distinguish an obfuscation of left TM from that of right output by  $\text{Sam}_A^i$ . The description of these two algorithms is as follows:

**Sampler  $\text{Sam}_A^i(\rho)$ :**

1. Receive  $(x_0^*, x_1^*)$  from  $A$ .
2. Parse  $\rho$  as  $(\text{crs}, h, \tau)$ .
3. Proceed identically to  $H_4$  using  $\tau$  as randomness for all tasks except for sampling the hash function which is set to  $h$ , and the CRS, which is set to  $\text{crs}$ . This involves the following steps:
  - (a) Parse  $\tau = (\tau_1, \tau_2, \tau_3, r_1, r_2, r_3, u_1, u_3)$ .
  - (b) Use  $\tau_i$  as randomness to generate  $(pk_i, sk_i) \leftarrow \text{Gen}(1^n; \tau_i)$  for  $i \in [3]$ ,  $r_1, r_2$  to generate  $c_1 = \text{Enc}_{pk_1}(x_0^*; r_1)$ ,  $c_2 = \text{Enc}_{pk_1}(x_1^*; r_2)$ , and  $u_1, u_2$  to generate  $\alpha_1 = \text{com}(h(x_0^*); u_1)$ ,  $\alpha_2 = \text{com}(h(x_1^*); u_2)$ .
  - (c) Use  $a^* = x_0^* \| r_1 \| u_1 \| x_1^* \| r_2 \| u_2$  and  $r_3$  to compute  $c_3^* = \text{Enc}_{pk_3}(a^*; r_3)$ , and then use  $w = (a^*, r_3)$  to compute proof  $\pi^*$  corresponding to conditions 1 and 2(b) in  $R_{\text{FE}}$ .
4. Define  $\text{pp} = (pk_1, pk_2, pk_3, \alpha_1, \alpha_2, \text{crs}, h)$  and challenge  $e = (c_1, c_2, c_3^*, \pi^*)$ .
5. Send  $\text{pp}$  to  $A$  and answer its secret-key queries as follows. For all queries  $M_j$  until  $j < i$ , send an obfuscation of  $\text{Prog}_{M, sk_1}$ .
6. If  $i$ -th secret-key query comes in phase 2, send ciphertext  $e$  in the challenge phase.
7. Upon receiving the  $i$ -th secret-key query  $M_i$ , output  $(M_0, M_1)$  and halt where:

$$M_0 := \text{Prog}_{M_i, sk_1}, \quad M_1 := \text{Prog}_{M_i, sk_2}^*.$$

**Distinguisher  $\mathcal{D}_A^i(\rho, M')$ :** on input a random tape  $\rho$  and an obfuscated TM  $M'$ , the distinguisher simply executes all steps of the sampler  $\text{Sam}_A^i(\rho)$ , answering secret-keys for all  $j < i$ , as described above. The distinguisher, however, does not halt when  $i$ -th query is sent, and continues the execution of  $A$  answering secret-key queries for  $M_j$  as follows:

- if  $j = i$ : send  $M'$  (which is an obfuscation of either  $M_0$  or  $M_1$ )
- if  $j > i$ : send an obfuscation of  $\text{Prog}_{M_j, sk_2}^*$

The distinguisher outputs whatever  $A$  outputs.

It is straightforward to see that if  $M'$  is an obfuscation of  $M_1$ , the output of  $\mathcal{D}_A^i$  is identical to  $A$ 's output in  $H_3^{i-1}$ ; and if  $M'$  is an obfuscation of  $M_0$ , it is identical to  $A$ 's output in  $H_3^i$ . We have that  $\mathcal{D}_A^i$  distinguishes  $H_3^{i-1}$  and  $H_3^i$  with at least  $\delta/t$  advantage. All that remains to prove now is that  $\text{Sam}_A^i$  is a public-coin differing-inputs sampler.

**Claim 5.2.**  $\text{Sam}_A^i$  is a public-coin differing-inputs sampler.

**Proof.** We show that if there exists an adversary  $\mathcal{B}$  who can find differing-inputs to the pair of TMs sampled by  $\text{Sam}_A^i$  with noticeable probability, say  $\mu$ , we can use  $\mathcal{B}$  and  $\text{Sam}_A^i$  to construct an efficient algorithm  $\text{CollFinder}_{\mathcal{B}, \text{Sam}_A^i}$  which finds collisions in  $h$  with probability  $\mu - \text{negl}(n)$ . The algorithm works as follows:

**CollFinder $_{\mathcal{B}, \text{Sam}_A^i}(h)$ :**

The algorithm incorporates  $\mathcal{B}, \text{Sam}_A^i$ . On input a random hash function  $h \leftarrow \mathcal{H}_n$ , the algorithm works as follows:

- sample uniformly random strings  $(\text{crs}, \tau)$  to define a random tape  $\rho := (\text{crs}, h, \tau)$ .
- sample  $(M_0, M_1) \leftarrow \text{Sam}_A^i(\rho)$  and  $e \leftarrow \mathcal{B}(\rho)$ .
- recall that  $e$  is of the form  $(c_1, c_2, c_3, \pi)$  where  $c_3$  is an encryption under  $pk_3$  where  $(pk_3, sk_3)$  are sampled using parts of the randomness  $\tau$ .
- let  $x_0^* \neq x_1^*$  be the strings output by  $A$  during the **Init** phase in the execution of  $\text{Sam}_A^i$ .
- if  $\pi$  is a valid proof, compute  $a = \text{Dec}_{sk_3}(c_3)$  and let  $a = x'_1 \| r'_1 \| u'_1 \| x'_2 \| r'_2 \| u'_2$ .
- if  $h(x_0^*) = h(x_1^*)$ , output  $(x_0^*, x_1^*)$  as the collisions; otherwise, if  $\exists m_1 \in \{x'_1, x'_2\}$  and  $\exists m_2 \in \{x_0^*, x_1^*\}$  s.t.  $m_1 \neq m_2$  and  $h(m_1) = h(m_2)$  output  $(m_1, m_2)$  as collisions; if none of the two conditions hold, output  $\perp$ .

Let us now analyze the success probability of this algorithm. Since  $h$  is uniformly sampled,  $\rho$  is a uniform random tape, and therefore with probability  $\mu$ ,  $\mathcal{B}$  outputs an  $e$  such that  $M_0(e) \neq M_1(e)$ . Recall that  $M_0 = \text{Prog}_{M_i, sk_1}^*$  and  $M_1 = \text{Prog}_{M_i, sk_2}^*$  for some TM  $M_i$  such that  $M_i(x_0^*) = M_i^*(x_1^*)$ . Furthermore, both of these programs output  $\perp$  if proof  $\pi$  is not valid. Since the output these two programs differ on  $e$ , it must be that  $\pi$  is a valid proof so that  $M_0(e) = M_i(\text{Dec}_{sk_1}(c_1))$  and  $M_1(e) = M_i(\text{Dec}_{sk_2}(c_2))$ . By construction, since  $\pi$  is a statistically sound proof, except with negligible probability it holds that  $x'_1 = \text{Dec}_{sk_1}(c_1)$  and  $x'_2 = \text{Dec}_{sk_2}(c_2)$  where  $x'_1, x'_2$  are part of the string  $a$  obtained by the collision finding algorithm by decrypting  $c_3$  above. Therefore, we have that  $M_0(e) = M_i(x'_1)$  and  $M_1(e) = M_i(x'_2)$ . However, we also have that  $M_0(e) \neq M_1(e) \implies M_i(x'_1) \neq M_i(x'_2) \implies x'_1 \neq x'_2$ . Since  $M_i(x_0^*) = M_i(x_1^*)$  it holds that the sets  $\{x'_1, x'_2\} \neq \{x_0^*, x_1^*\}$ .

Since  $\pi$  is valid, and  $c_1, c_2$  are encryptions of (unequal strings)  $x'_1, x'_2$ , from the statistical soundness of  $\pi$  statements 2(b.ii) and 2(b.iii) must be true. That is,  $\alpha_1$  (likewise  $\alpha_2$ ) must be a commitment to one of  $h(x'_1)$  or  $h(x'_2)$ . But  $\alpha_1$  is a commitment to  $h(x_0^*)$  and  $\alpha_2$  is a commitment to  $h(x_1^*)$  and commitment is statistically binding. Since at least one of  $x'_1, x'_2$  is not equal to any of  $x_0^*, x_1^*$  the collision must occur on one of the four possible pairs of these strings. ■

**Step 5: Indistinguishability of  $H_4$ – $H_8$ .** Hybrids  $H_4$  to  $H_8$  are applying changes very similar to the first four hybrids except in the reverse order. The proof of their indistinguishability can be obtained by following previous proofs in a near identical fashion. In particular we can prove  $H_4 \approx_c H_5$  by relying on the security of encryption and strong WI (following the proof in step 2 or 3),  $H_5 \approx_c H_6$  following the proof in step 4,  $H_6 \approx_c H_7$  following the proof in step 2, and  $H_7 \approx_c H_8$  following the proof in step 1.

This completes the proof of security of our functional encryption scheme. ■

## References

- [ABG<sup>+</sup>13] Prabhakaran Ananth, Dan Boneh, Sanjam Garg, Amit Sahai, and Mark Zhandry. Differing-inputs obfuscation and applications. *IACR Cryptology ePrint Archive*, 2013, 2013.
- [AIK06] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in  $\text{nc}^0$ . *SIAM J. Comput.*, 36(4):845–888, 2006.
- [App11] Benny Applebaum. Randomly encoding functions: A new cryptographic paradigm - (invited talk). In *ICITS*, pages 25–31, 2011.
- [BC14] Nir Bitansky and Ran Canetti. On strong simulation and composable point obfuscation. *J. Cryptology*, 27(2):317–357, 2014.

- [BCC<sup>+</sup>14] Nir Bitansky, Ran Canetti, Alessandro Chiesa, Shafi Goldwasser, Huijia Lin, Aviad Rubinfeld, and Eran Tromer. The hunting of the SNARK. *IACR Cryptology ePrint Archive*, 2014:580, 2014.
- [BCCT12] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012*, pages 326–349, 2012.
- [BCCT13] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for SNARKS and proof-carrying data. In *Symposium on Theory of Computing Conference, STOC’13, Palo Alto, CA, USA, June 1-4, 2013*, pages 111–120, 2013.
- [BCKP14] Nir Bitansky, Ran Canetti, Yael Tauman Kalai, and Omer Paneth. On virtual grey box obfuscation for general circuits. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II*, pages 108–125, 2014.
- [BCP14] Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability (a.k.a. differing-inputs) obfuscation. In *TCC*, 2014. Preliminary version on Eprint 2013: <http://eprint.iacr.org/2013/650.pdf>.
- [BCPR14] Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. On the existence of extractable one-way functions. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 505–514, 2014.
- [BG02] Boaz Barak and Oded Goldreich. Universal arguments and their applications. In *Annual IEEE Conference on Computational Complexity (CCC)*, volume 17, 2002. Preliminary full version available as Cryptology ePrint Archive, Report 2001/105.
- [BGI<sup>+</sup>12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6, 2012.
- [BGK<sup>+</sup>14] Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks. In *EUROCRYPT*, 2014.
- [BGT14] Nir Bitansky, Sanjam Garg, and Sidharth Telang. Succinct randomized encodings and their applications. Cryptology ePrint Archive, Report 2014/771, 2014. <http://eprint.iacr.org/>.
- [BHK13] Mihir Bellare, Viet Tung Hoang, and Sriram Keelveedhi. Instantiating random oracles via uces. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, pages 398–415, 2013.
- [BHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In *the ACM Conference on Computer and Communications Security, CCS’12, Raleigh, NC, USA, October 16-18, 2012*, pages 784–796, 2012. Cryptology Eprint Archive Report 2012/265.
- [BP13] Elette Boyle and Rafael Pass. Limits of extractability assumptions with distributional auxiliary input. Cryptology ePrint Archive, Report 2013/703, 2013. <http://eprint.iacr.org/2013/703.pdf>.

- [BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.
- [BR14] Zvika Brakerski and Guy N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. In *TCC*, 2014. Preliminary version on Eprint at <http://eprint.iacr.org/2013/563.pdf>.
- [BST14] Mihir Bellare, Igors Stepanovs, and Stefano Tessaro. Poly-many hardcore bits for any one-way function and a framework for differing-inputs obfuscation. In *ASIACRYPT*, pages 102–121, 2014. Earlier version: IACR Cryptology ePrint Archive 2013:873 (December 2013).
- [BSW11] Elette Boyle, Gil Segev, and Daniel Wichs. Fully leakage-resilient signatures. In *EUROCRYPT*, pages 89–108, 2011.
- [Can97] Ran Canetti. Towards realizing random oracles: Hash functions that hide all partial information. In *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings*, pages 455–469, 1997.
- [CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004.
- [CHJV14] Ran Canetti, Justin Holmgren, Abhishek Jain, and Vinod Vaikuntanathan. Indistinguishability obfuscation of iterated circuits and ram programs. Cryptology ePrint Archive, Report 2014/769, 2014. <http://eprint.iacr.org/>.
- [CLT13] Jean-Sebastien Coron, Tancrede Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In *CRYPTO*, pages 476–493, 2013.
- [FKN94] Uriel Feige, Joe Kilian, and Moni Naor. A minimal model for secure computation. In *STOC*, pages 554–563, 1994.
- [FLS99] Feige, Lapidot, and Shamir. Multiple noninteractive zero knowledge proofs under general assumptions. *SIAM Journal on Computing*, 29, 1999.
- [Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.
- [GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *EUROCRYPT*, pages 1–17, 2013.
- [GGH<sup>+</sup>13b] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, pages 40–49, 2013.
- [GGHW14] Sanjam Garg, Craig Gentry, Shai Halevi, and Daniel Wichs. On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, pages 518–535, 2014.
- [GK05] Shafi Goldwasser and Yael Tauman Kalai. On the impossibility of obfuscation with auxiliary input. In *FOCS*, pages 553–562, 2005.

- [GKP<sup>+</sup>13a] Shafi Goldwasser, Yael Kalai, Raluca Ada Popa, Vinod Vaikuntanathan, , and Nickolai Zeldovich. How to run turing machines on encrypted data. In *CRYPTO*, 2013.
- [GKP<sup>+</sup>13b] Shafi Goldwasser, Yael Kalai, Raluca Ada Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Succinct functional encryption and applications: Reusable garbled circuits and beyond. In *STOC*, 2013.
- [Gol01] Oded Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, 2001. Earlier version available on <http://www.wisdom.weizmann.ac.il/~oded/frag.html> .
- [GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *CCS*, 2006.
- [GR07] Shafi Goldwasser and Guy N. Rothblum. On best-possible obfuscation. In *TCC*, pages 194–213, 2007.
- [GR13] Oded Goldreich and Ron D. Rothblum. Enhancements of trapdoor permutations. *J. Cryptology*, 26(3):484–512, 2013.
- [HR04] Chun-Yuan Hsiao and Leonid Reyzin. Finding collisions on a public road, or do secure hash functions need secret coins? In *CRYPTO*, pages 92–105, 2004.
- [IK00] Yuval Ishai and Eyal Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*, pages 294–304, 2000.
- [KLW14] Venkata Koppula, Allison Bishop Lewko, and Brent Waters. Indistinguishability obfuscation for turing machines with unbounded memory. Cryptology ePrint Archive, Report 2014/925, 2014.
- [LP14] Huijia Lin and Rafael Pass. Succinct garbling schemes and applications. Cryptology ePrint Archive, Report 2014/766, 2014. <http://eprint.iacr.org/>.
- [Mic94] S. Micali. CS proofs. In *FOCS*, pages 436–453, 1994.
- [PPS15] Omkant Pandey, Manoj Prabhakaran, and Amit Sahai. Obfuscation-based non-black-box simulation and four message concurrent zero knowledge for NP. In *TCC*, 2015. Earlier version: IACR Cryptology ePrint Archive 2013:754.
- [SS10] Amit Sahai and Hakan Seyalioglu. Worry-free encryption: functional encryption with public keys. In *CCS*, pages 463–472, 2010.
- [SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, pages 457–473, 2005.
- [SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *STOC*, pages 475–484, 2014.
- [Yao82] Andrew C. Yao. Theory and applications of trapdoor functions. In *Proc. 23rd FOCS*, pages 80–91, 1982.

## A Other primitives

**Fully homomorphic encryption with decryption in  $\text{NC}^1$ .** A fully homomorphic encryption (FHE) scheme is a public-key encryption scheme with an additional evaluation algorithm  $\text{Eval}$ . Formally, given a public-key  $pk$ , ciphertexts  $c_1, \dots, c_m$  corresponding to the bits  $b_1, \dots, b_m$  (under  $pk$ ), and a circuit  $f : \{0, 1\}^m \rightarrow \{0, 1\}$ , algorithm  $\text{Eval}$  outputs a ciphertext  $c'$  such that except with negligible probability over the randomness of all algorithms, the decryption of  $c'$  is the bit  $f(b_1, \dots, b_m)$  where  $m = m(n)$  is an arbitrary polynomial.

The encryption of a long message  $x \in \{0, 1\}^n$  under  $pk$  consists of encrypting each bit of  $x$  under  $pk$ , and will be denoted by  $c = \text{Enc}_{pk}(x)$ . Given  $c$ , the homomorphic evaluation of an *oblivious Turing machine*  $M$  with known running time  $t$  consists of  $t$  homomorphic evaluations of the the circuit corresponding to the transition function of  $M$  where in the  $i$ -th iteration the transition function is applied on the contents of the encrypted input/work tape (containing  $x$  at the start) and an encrypted state; it results in a new encrypted state as well as new encrypted contents on the work tape.

A FHE scheme has decryption in  $\text{NC}^1$  if there exists a constant  $c \in \mathbb{N}$  such that for all  $n \in \mathbb{N}$  the depth of the circuit corresponding to the decryption function  $\text{Dec}(1^n, pk, \cdot)$  is at most  $c \log n$ .

**Strong non-interactive witness indistinguishable proofs for  $\text{NP}$ .** As a tool for our functional encryption application, we need non-interactive proofs for  $\text{NP}$  in the common random string model. We require that the proof system be capable of proving statements of unbounded polynomial length. In terms of soundness, we require the system to be a *proof* system where the soundness guarantee is statistical: i.e., even unbounded provers cannot prove a false statement with noticeable probability. In terms of prover security, we only require the proof system to satisfy *strong* witness indistinguishability [Gol01] which is implied by the zero-knowledge property. The NIZK proof system of Feige, Lapidot, and Shamir [FLS99] satisfies all of these requirements.

## B Bounded-Input Case

In this section we consider the  *$\ell$ -bounded-input* case, in which we consider the class of TMs whose input is bounded by a polynomial  $\ell$ , and the size of the obfuscation is allowed to depend on  $\ell$ ; however it does not depend on the running time of TMs in the class, which could be much larger than  $\ell$ . To emphasize that a class is a bounded-input TM class, we will explicitly include  $\ell$  in the notation.

**Definition B.1** (Public-Coin Differing-Inputs Obfuscator for  $\ell$ -Bounded-Input TMs). For every polynomial  $\ell : \mathbb{N} \rightarrow \mathbb{N}$ , let  $\mathcal{M} = \{\mathcal{M}_n^\ell\}_{n \in \mathbb{N}}$  denote the class of all TMs such that every  $M \in \mathcal{M}_n^\ell$  is of size  $n$ , accepts inputs of length at most  $\ell(n)$ , and halts within a polynomial, say  $t(n)$ , number of steps on all inputs. A uniform PPT algorithm  $\mathcal{O}$  is a *public-coin differing-inputs obfuscator for the class of all bounded-input TMs* if it satisfies the *correctness* and *security* conditions of definition 2.4 and the following modified succinctness condition: there exists a (global) polynomial  $s' : \mathbb{N} \rightarrow \mathbb{N}$  such that for all  $n$ , for all  $M \in \mathcal{M}_n$ , and for all  $M' \leftarrow \mathcal{O}(1^n, M)$ , the size of  $M'$  is bounded by  $s'(n, \ell(n))$  and its running time is bounded by  $s'(n, t(n))$  for all  $x \in \{0, 1\}^{\leq \ell(n)}$ .

We show that given a public-coin differing-inputs obfuscator for the class  $\text{NC}^1$ , we can construct a public-coin differing-inputs obfuscator for *bounded-input Turing machines*' class.

**Theorem B.2.** *Suppose that there exists a public-coin differing-inputs obfuscator for circuits in the class  $\text{NC}^1$ . Then, assuming the existence of a fully homomorphic encryption scheme with decryption in  $\text{NC}^1$  and a (publicly verifiable) SNARG system for  $\mathbf{P}$  (alternatively, a  $\mathbf{P}$ -certificate) in the common random string model, there exists a public-coin differing-inputs obfuscator for bounded-input Turing machines as defined in B.1.*



We first present the construction, and then prove the theorem. Let  $\ell$  and  $t$  be polynomials, and let  $\mathcal{M} = \{\mathcal{M}_n^\ell\}_{n \in \mathbb{N}}$  be the family of bounded-input TMs where every  $M \in \mathcal{M}_n^\ell$  is of size  $n$ , accepts inputs of length at most  $\ell(n)$  and halts within  $t(n)$  steps on every  $x$ . Let  $\text{FHE} = (\text{Gen}, \text{Enc}, \text{Dec}, \text{Eval})$  be a fully homomorphic encryption scheme with decryption in  $\text{NC}^1$ , and  $\Pi = (P, V)$  be a SNARG for the relation  $\mathcal{R}_c$  defined earlier where  $c$  is a constant such that  $t(n) \leq n^c$  for all  $n$ . The description of our obfuscator and its evaluation algorithm, are as follows.

**Obfuscator**  $\mathcal{O}(1^n, M \in \mathcal{M}_n^\ell)$ : By convention, description of  $M$  includes the bounds  $t$  and  $\ell$ . Let  $U_n$  be an *oblivious* universal TM which on input the description of a TM  $B$ , and a string  $x \in \{0, 1\}^{\leq \ell(n)}$  executes  $B$  on  $x$  for no more than  $t(n)$  steps. The obfuscator proceeds in the following steps:

1. Generate two FHE public-keys  $(pk_1, sk_1) \leftarrow \text{Gen}(1^n; u_1)$  and  $(pk_2, sk_2) \leftarrow \text{Gen}(1^n; u_2)$ .
2. Encrypt  $M$  under both FHE public-keys:  $g_1 \leftarrow \text{Enc}_{pk_1}(M; v_1)$  and  $g_2 \leftarrow \text{Enc}_{pk_2}(M; v_2)$ . Here  $M$  is assumed to be encoded as a bit string of length  $n$  for use by the universal TM  $U_n$ .
3. Uniformly sample  $\text{crs} \leftarrow \{0, 1\}^{\text{poly}(n)}$  of sufficient length for SNARG  $\Pi$ .
4. Generate an obfuscation of the  $\text{NC}^1$ -program  $P1_{sk_1, g_1, g_2}^{\text{crs}}$  given in figure 2:

$$P' \leftarrow \mathcal{O}_{\text{NC}^1}(1^n, P1_{sk_1, g_1, g_2}^{\text{crs}}).$$

5. Output  $M' = (P', \text{crs}, pk_1, pk_2, g_1, g_2)$ .

**Program**  $P1_{sk_1, g_1, g_2}^{\text{crs}}$ :

- **Input:** a tuple  $(x, e_1, e_2, \pi, \phi)$ ,    **Constants:**  $\text{crs}, sk_1, g_1, g_2, pk_1, pk_2$ .
- Check that  $\phi$  is a valid low-depth proof for the **NP**-statement:
 
$$1 = V\left(\text{crs}, (\widetilde{M}_{\text{Eval}}, \langle x, e_1, e_2 \rangle, t^2), \pi\right)$$

where  $\widetilde{M}_{\text{Eval}}$  simply checks that computation  $M_{\text{Eval}}(x)$  outputs  $(e_1, e_2)$  in  $\leq 2t \log t$  steps.  $M_{\text{Eval}}(x)$  is defined as follows: it has  $(pk_1, pk_2, g_1, g_2)$  hardcoded, and homomorphically evaluates  $U_n(\cdot, x)$  on  $g_1$  and  $g_2$  to produce  $e_1$  and  $e_2$  respectively. I.e.,  $e_1 = \text{Eval}_{pk_1}(U_n(\cdot, x), g_1)$  and  $e_2 = \text{Eval}_{pk_2}(U_n(\cdot, x), g_2)$ .
- If the check fails, output  $\perp$ ; otherwise output  $\text{Dec}_{sk_1}(e_1)$ .

**Program**  $P2_{sk_2, g_1, g_2}^{\text{crs}}$ :

- Same as  $P1_{sk_1, g_1, g_2}^{\text{crs}}$  except that if the check is successful, it outputs  $\text{Dec}_{sk_2}(e_2)$ .

Figure 2: The programs  $P1$  and  $P2$

**Evaluation of  $M'$ .** Evaluate  $M' = (P', \text{crs}, pk_1, pk_2, g_1, g_2)$  on input  $x$  as follows:

1. Let  $(e_1, e_2) = M_{\text{Eval}}(x)$ . Recall that (fig. 1):  $e_1 = \text{Eval}_{pk_1}(U_n(\cdot, x), g_1)$ ,  $e_2 = \text{Eval}_{pk_2}(U_n(\cdot, x), g_2)$
2. W.l.o.g, the running time of  $M_{\text{Eval}}(x)$  is at most  $2t \log t \leq t^2$ . Compute the proof  $\pi$ :<sup>6</sup>

$$\pi \leftarrow P\left(\text{crs}, (\widetilde{M}_{\text{Eval}}, \langle x, e_1, e_2 \rangle, t^2), \perp\right)$$

<sup>6</sup>No witness is necessary as this is computation in  $\mathbf{P}$ .

3. Compute a low-depth proof  $\phi$  that  $\pi$  is a valid SNARG, i.e.,  $V(\text{crs}, (\widetilde{M}_{\text{Eval}}, \langle \sigma, e_1, e_2 \rangle, t^2), \pi) = 1$ . This can be done by providing the entire computation of  $V$  on these inputs.
4. Execute  $P'(x, e_1, e_2, \pi, \phi)$  and output the result.

The construction is now analyzed in the proof below.

**Proof of theorem B.2.** The correctness and succinctness of this construction are relatively straightforward to verify, and in particular, are identical to the analyses in [ABG<sup>+</sup>13, BCP14].

**Security.** Fix any public-coin differing-inputs sampler  $\text{Sam} = \{\text{Sam}_n\}$  for the family  $\mathcal{M}$  and any efficient distinguisher  $\mathcal{D} = \{\mathcal{D}_n\}$ . For a bit  $b$ , let  $\mathcal{X}_n(b)$  denote the output of the following experiment over a random choice of  $r$  and the coins of  $\mathcal{O}$ :

$$\mathcal{X}_n(b) := \{(M_0, M_1) \leftarrow \text{Sam}_n(r), M' \leftarrow \mathcal{O}(1^n, M_b), \text{output } \mathcal{D}_n(r, M')\}$$

We need to show that  $\mathcal{X}_n(0) \approx_c \mathcal{X}_n(1)$ . Consider the following sequence of hybrid experiments.

- $H_0$ : This hybrid corresponds to an honest sampling of  $\mathcal{X}_n(0)$ . In this case,  $M'$  creates two FHE encryptions of  $M_0$ , namely  $g_1$  and  $g_2$  (where  $M_0$  is the first output of  $\text{Sam}_n$ ).
- $H_1$ : Same as  $H_0$  except that the second FHE ciphertext is now generated as an encryption of  $M_1$ , i.e.,  $g_2 = \text{Enc}_{pk_2}(M_1)$  (where  $M_1$  is the second output of  $\text{Sam}_n$ ).
- $H_2$ : Same as  $H_1$  except that the obfuscated program  $P'$  is now generated as an obfuscation of  $P2_{sk_2, g_1, g_2}^{\text{crs}}$  which decrypts the *second* ciphertext using  $sk_2$ , i.e.,  $P' \leftarrow \mathcal{O}_{\text{NC}^1}(1^n, P2_{sk_2, g_1, g_2}^{\text{crs}})$ .
- $H_3$ : Same as  $H_2$  except that the *first* FHE ciphertext  $g_1$  is now also generated as an encryption of  $M_1$ , i.e.,  $g_1 \leftarrow \text{Enc}_{pk_1}(M_1)$ .
- $H_4$ : Same as  $H_3$  except that the obfuscated program  $P'$  is once again generated as an obfuscation of  $P1_{sk_1, g_1, g_2}^{\text{crs}}$ , i.e.,  $P' \leftarrow \mathcal{O}_{\text{NC}^1}(1^n, P1_{sk_1, g_1, g_2}^{\text{crs}})$ . Note that  $H_4$  is identical to  $\mathcal{X}_n(1)$ .

We now prove that each hybrid in this sequence is indistinguishable from the previous one.

**Step 1:**  $H_0 \approx_c H_1$ . This follows from the IND-CPA security of FHE. Formally, consider an adversary  $\mathcal{A}_{\text{FHE}}$ , who receives a challenge public-key  $pk$ , then samples  $(M_0, M_1) \leftarrow \text{Sam}_n(r)$  for a random  $r$ , and receives an honestly generated ciphertext  $g$  to either  $M_0$  or  $M_1$  under  $pk$ .  $\mathcal{A}_{\text{FHE}}$  then generates an obfuscation of  $M_0$  following the instruction of  $\mathcal{O}$  except that it sets  $pk_2 = pk$  and  $g_2 = g$ . Note that all instructions of  $\mathcal{O}$  can indeed be performed efficiently knowing only  $(pk_2, g_2)$ . Let  $M'$  denote the resulting obfuscation which includes an  $\text{NC}^1$ -obfuscation  $P'$  of program  $P1_{sk_1, g_1, g_2=g}^{\text{crs}}$ .  $\mathcal{A}_{\text{FHE}}$  outputs whatever  $\mathcal{D}_n(r, M')$  outputs. The output of  $\mathcal{A}_{\text{FHE}}$  is distributed identically to that of  $H_b$  when  $g$  is an encryption of  $M_b$  where  $b \in \{0, 1\}$ . Because  $\text{Sam}_n$  and  $\mathcal{D}_n$  are of polynomial-size, it follows that  $\mathcal{A}_{\text{FHE}}$  is a polynomial-size circuit violating IND-CPA security of FHE unless  $H_0 \approx_c H_1$ .

**Step 2:**  $H_1 \approx_c H_2$ . We use the soundness of SNARG and diO-security of  $\mathcal{O}_{\text{NC}^1}$  to argue that  $H_1 \approx_c H_2$ . Suppose that  $H_1$  and  $H_2$  are not computationally indistinguishable. We use  $\text{Sam}_n$  and  $\mathcal{D}_n$  to construct a public-coin differing-inputs sampler  $\text{Sam}_n^{\text{NC}^1}$  along with a distinguisher  $\mathcal{D}_n^{\text{NC}^1}$  such that  $\text{Sam}_n^{\text{NC}^1}$  outputs circuits in  $\text{NC}^1$  and  $\mathcal{D}_n^{\text{NC}^1}$  violates the security of  $\mathcal{O}_{\text{NC}^1}$  w.r.t.  $\text{Sam}_n^{\text{NC}^1}$ . We start by constructing  $\text{Sam}_n^{\text{NC}^1}$ .

**Sampler  $\text{Sam}_n^{\text{NC}^1}(\rho)$ .**

1. Parse  $\rho$  as  $\rho = (r, \rho_1, u_1, u_2, v_1, v_2)$ .
2. Sample  $(M_0, M_1) \leftarrow \text{Sam}_n(r)$ . // comment: this is the given sampler.
3. Set  $\text{crs} = \rho_1$ ,  $(pk_1, sk_1) \leftarrow \text{Gen}(1^n; u_1)$ ,  $(pk_2, sk_2) \leftarrow \text{Gen}(1^n; u_2)$ .
4. Set  $g_1 \leftarrow \text{Enc}_{pk_1}(M_0; v_1)$  and  $g_2 \leftarrow \text{Enc}_{pk_2}(M_1; v_2)$ .
5. Output  $(C_0, C_1)$  corresponding to the programs  $(P1_{sk_1, g_1, g_2}^{\text{crs}}, P2_{sk_2, g_1, g_2}^{\text{crs}})$ .

Note that input to the circuits  $C_0, C_1$  above are of the form  $m = (x, e_1, e_2, \pi, \phi)$ .

**Claim B.3.**  $\forall n \in \mathbb{N} \text{Sam}_n^{\text{NC}^1}$  is a public-coin differing-inputs sampler for a family  $\mathcal{C} \in \text{NC}^1$ .

**Proof.** We have to show that every non-uniform PPT attacker  $\{\mathcal{A}_n^{\text{NC}^1}\}$  fails to find a differing-input for circuits sampled by  $\text{Sam}_n^{\text{NC}^1}$ . We prove this by using (any) given  $\mathcal{A}_n^{\text{NC}^1}$  to construct an attacker-family  $\{\mathcal{A}_n\}$  against the given sampler  $\{\text{Sam}_n\}$ . We then use the soundness of SNARG to argue that if  $\mathcal{A}_n^{\text{NC}^1}$  “succeeds” against our sampler then  $\mathcal{A}_n$  succeeds against the given sampler.

Formally, suppose that the claim is false, and there exists a polynomial-size attacker family  $\{\mathcal{A}_n^{\text{NC}^1}\}$ , a polynomial  $p$ , and infinitely many  $n$  s.t.

$$\Pr_{\rho} \left[ C_0(x) \neq C_1(x) : (C_0, C_1) \leftarrow \text{Sam}_n^{\text{NC}^1}(\rho), x \leftarrow \mathcal{A}_n^{\text{NC}^1}(\rho) \right] \geq 1/p(n).$$

For every sufficiently large string  $z_n = (\rho_1, u_1, u_2, v_1, v_2)$  define the following circuit  $\mathcal{A}_n^{(z_n)}$ :

**Circuit  $\mathcal{A}_n^{(z_n)}(r)$ :** String  $z_n$  and adversary  $\mathcal{A}_n^{\text{NC}^1}$  are hardcoded in this circuit.

1. Define  $\rho := r \circ z_n = (r, \rho_1, u_1, u_2, v_1, v_2)$ .
2. Compute  $m \leftarrow \mathcal{A}_n^{\text{NC}^1}(\rho)$  which is of the form  $m = (x, e_1, e_2, \pi, \phi)$ .
3. Output  $x$ .

We say that a (fixed) string  $z_n$  is **good** if, the success probability of  $\mathcal{A}_n^{\text{NC}^1}$  in step 2 above is at least  $\frac{1}{2p}$  (over the choice of  $r$ ). I.e., in step 2,  $\mathcal{A}_n^{\text{NC}^1}$  outputs  $m$  such that  $C_0(m) \neq C_1(m)$  where  $(C_0, C_1) \leftarrow \text{Sam}_n^{\text{NC}^1}(\rho)$ . By simple averaging, at least  $\frac{1}{2p}$  fraction of  $z_n$  are **good**.

Now let us define **sound** strings. Roughly speaking, we say that  $z_n$  is **sound** if the the output of  $\mathcal{A}_n^{\text{NC}^1}$  (in step 2 above) results in a valid proof for a false statement with less than  $1/4p$  probability w.r.t.  $\text{crs} = \rho_1$  contained in  $z_n$ . Formally, we say that a fixed string  $z_n$  is **sound** if the probability that  $\mathcal{A}_n^{\text{NC}^1}$  outputs  $m = (x, e_1, e_2, \pi, \phi)$  (in step 2 above) such that  $V$  accepts the proof  $\pi$  w.r.t.  $\text{crs} = \rho_1$  but  $(e_1, e_2)$  is not the output of computation  $M_{\text{Eval}}$  within  $2t \log t$  steps is at most  $1/4p$  where the probability is taken over  $r$ . Since a random choice of  $z_n$  results in a uniformly distributed  $\text{crs}$  string, it follows that at least  $1 - \varepsilon'$  fraction of  $z_n$  are **sound** where  $\varepsilon'$  is the soundness error of SNARG.

Therefore, at least  $\frac{1}{2p} - \varepsilon' \geq \frac{1}{4p}$  fraction of  $z_n$  are both **good** and **sound**. Fix such a  $z_n$ . By definition of **good** it follows that w.r.t. such a fixed  $z_n$ , at least  $1/2p$  fraction of inputs  $r$  are such that a differing-input will be found in step 2. Further, by definition of **sound**, at most a  $1/4p$  fraction of such inputs  $r$  can ever lead to a valid SNARG proof for a false statement. Therefore, at least  $\frac{1}{2p} - \frac{1}{4p} \geq \frac{1}{4p}$  of inputs  $r$  result in a differing-input where the statement for SNARG is actually true. We call such inputs **nice**.

By construction, for nice  $r$ , we have that:  $C_0(m) \neq C_1(m) \implies P1_{sk_1, g_1, g_2}^{\text{crs}=\rho_1}(m) \neq P2_{sk_2, g_1, g_2}^{\text{crs}=\rho_1}(m) \implies M_0(x) \neq M_1(x)$ . Here  $M_0, M_1$  are the TMs sampled in (step 2 of) the execution of  $\text{Sam}_n^{\text{NC}^1}$  (on randomness  $\rho = r \circ z_n$ ). But, again by construction,  $(M_0, M_1)$  are also output of  $\text{Sam}_n(r)$ . Therefore,  $\mathcal{A}_n^{(z_n)}$  outputs a differing input  $x$  for the outputs of  $\text{Sam}_n$  whenever  $z_n$  is **good** and **sound** and  $r$  is **nice**. This happens with probability at least  $1/4p \times 1/4p = 1/16p^2$  which is non-negligible.

Finally, observe that circuits output by  $\text{Sam}_{\text{NC}^1}$  are indeed in the complexity class NC. ■

We now present distinguisher  $\mathcal{D}_n^{\text{NC}^1}$  which violates the security of  $\mathcal{O}_{\text{NC}^1}$  w.r.t. sampler  $\text{Sam}_n^{\text{NC}^1}$ .

**Distinguisher  $\mathcal{D}_n^{\text{NC}^1}(\rho, C')$ .** The input consists a string  $\rho$  and an obfuscated circuit  $C'$ .  $C'$  is an obfuscation of either  $C_0$  or  $C_1$  which are output by  $\text{Sam}_n^{\text{NC}^1}(\rho)$ . The distinguisher attempts to create a valid obfuscation  $M'$  of the TM implicitly present in  $C'$ . Since entire string  $\rho$  is available, it can be efficiently done as follows.

1. Parse  $\rho$  as  $\rho = (r, \rho_1, u_1, u_2, v_1, v_2)$ , and set  $\text{crs} = \rho_1$ ,  $(pk_1, sk_1) \leftarrow \text{Gen}(1^n; u_1)$ ,  $(pk_2, sk_2) \leftarrow \text{Gen}(1^n; u_2)$ ,  $g_1 \leftarrow \text{Enc}_{pk_1}(M_0; v_1)$  and  $g_2 \leftarrow \text{Enc}_{pk_2}(M_1; v_2)$ , where  $(M_0, M_1) \leftarrow \text{Sam}(1^n; r)$ .
2. Define  $M' = (C', \text{crs}, pk_1, pk_2, g_1, g_2)$ , and output whatever  $\mathcal{D}_n(r, M')$  outputs. (Recall that  $\mathcal{D}_n$  is the given distinguisher).

By construction of  $\text{Sam}_n^{\text{NC}^1}$ , we can see that if  $C'$  is a correctly generated obfuscation of  $C_b$ , then  $M'$  is distributed as in hybrid  $H_{b+1}$ . It follows that if outputs of  $H_1$  and  $H_2$  are distinguishable then  $\mathcal{D}_n^{\text{NC}^1}$  is a good distinguisher against  $\mathcal{O}_{\text{NC}^1}$  w.r.t.  $\text{Sam}_n^{\text{NC}^1}$ .

**Final step:  $H_2 \approx_c H_3$  and  $H_3 \approx_c H_4$ .** Proof for the claim  $H_2 \approx_c H_3$  is nearly identical to step 1. The proof for  $H_3 \approx_c H_4$  is nearly identical to step 2. We omit the details. ■