The Usefulness of Sparsifiable Inputs: How to Avoid Subexponential iO

Thomas Agrikola[1,⋆], Geoffroy Couteau[2,⋆], and Dennis Hofheinz[3,⋆]

[1] Karlsruhe Institute of Technology, Karlsruhe, Germany
[2] IRIF, Paris-Diderot University, CNRS, France
[3] ETH Zurich, Switzerland

Work done while all authors were at Karlsruhe Institute of Technology.

Abstract. We consider the problem of removing subexponential reductions to indistinguishability obfuscation (iO) in the context of obfuscating probabilistic programs. Specifically, we show how to apply complexity absorption (Zhandry, Crypto 2016) to the recent notion of probabilistic indistinguishability obfuscation (piO, Canetti et al., TCC 2015). As a result, we obtain a variant of piO which allows to obfuscate a large class of probabilistic programs, from polynomially secure indistinguishability obfuscation and extremely lossy functions. Particularly, our piO variant is able to obfuscate circuits with specific input domains regardless of the performed computation. We then revisit several (direct or indirect) applications of piO, and obtain

- a fully homomorphic encryption scheme (without circular security assumptions),
- a multi-key fully homomorphic encryption scheme with threshold decryption,
- an encryption scheme secure under arbitrary key-dependent messages,
- a spooky encryption scheme for all circuits,
- a function secret sharing scheme with additive reconstruction for all circuits,

all from polynomially secure iO, extremely lossy functions, and, depending on the scheme, also other (but polynomial and comparatively mild) assumptions. All of these assumptions are implied by polynomially secure iO and the (non-polynomial, but very well-investigated) exponential DDH assumption. Previously, all the above applications required to assume the *subexponential* security of iO (and more standard assumptions).

Keywords: indistinguishability obfuscation, extremely lossy functions, subexponential assumptions.

⋆ Supported by ERC Project PREP-CRYPTO 724307.

Table of Contents

1 Introduction

Obfuscation. Code obfuscation has been formalized already in the early 2000s as a cryptographic building block, by Hada [Had00] and Barak et al. [Bar+01], along with a number of early positive [Can97; LPS04; Wee05; Hoh+07; HMS07] and negative [Bar+01; GK05; Wee05] results. However, prior to the candidate obfuscation scheme of Garg et al. [Gar+13], only relatively few positive results on obfuscation were known.

The first candidate obfuscator from [Gar+13] changed things. Their work identified indistinguishability obfuscation (iO, cf. [Bar+01; GR07]) as an achievable *and* useful general notion of obfuscation: it presented a candidate indistinguishability obfuscator, along with a first highly non-trivial application. Since then, a vast number of applications have been proposed, ranging from functional [Gar+13], deniable [SW14], and fully homomorphic [Can+15] encryption, over multi-party computation (e.g., [Gar+14a]), to separation results (e.g., [HRW16]). In the process, powerful techniques like "puncturing" [SW14] have been discovered, which have found applications even beyond obfuscation (e.g., in multi-party computation [BL18; GS18], instantiating the Fiat-Shamir paradigm [Can+18], and verifiable random functions [Bit17; Goy+17]). Besides, the notion of iO itself has been refined, and related to other notions of obfuscation [Ana+13; BP13; BCP14; Bit+14; Can+15; IPS15], and various different constructions of obfuscators have been presented [PST14; Zim15; AJ15; BV15; AS17; Lin17; LT17].

Subexponential assumptions. It is currently hard to find a cryptographic primitive that can *not* be constructed from iO (in combination with another mild assumption such as the existence of one-way functions). However, some of the known iO-based constructions come only with *subexponential* reductions to iO. For instance, the only known iO-based constructions of fully homomorphic encryption [Can+15], spooky encryption [Dod+16], and graded encoding schemes [Far+18] suffer from reductions with a subexponential loss.

Hence, while iO has generally been recognized as an extremely powerful primitive (even to the extent being called a "central hub" for cryptography [SW14]), it is not at all clear if this also holds for *polynomially* secure iO. Indeed, it is conceivable that only polynomially secure iO exists, in which case much of iO's power stands in question.

More generally, subexponential reductions (in particular to iO) are undesirable. Namely, the security of existing iO constructions is still not well-understood, and in particular current state-of-the-art constructions of iO schemes (such as [AS17; Lin17; LT17]) already require subexponential computational assumptions themselves. Hence, assuming subexponential iO is a particularly risky bet. This suspicion is confirmed in part by [Ps16], who separate polynomial and subexponential security for virtual black-box obfuscation.

Removing subexponential assumptions in general and from iO-based constructions in particular has already explicitly been considered in [LM16; GS16] and [GPS16; Gar+17; LZ17] respectively. These works offer general techniques and ideas to turn subexponential reductions into polynomial ones. For instance, [Gar+17; LZ17] offer ways to replace (subexponential) iO-based constructions with (polynomial) constructions based on functional encryption. Of course, this requires a special structure of the primitive to be implemented, and is demonstrated for several primitives, including non-interactive key exchange and short signature schemes.

Our contribution. In this work, we are also concerned with substituting subexponential with polynomial reductions in iO-based constructions. Unlike [Gar+17; LZ17], however, we do not follow the approach of using functional encryption directly in place of iO, but instead will employ extremely lossy functions (ELFs) [Zha16] to "absorb" subexponential complexity.[4]

We will implement a variant of probabilistic indistinguishability obfuscation (piO, introduced in [Can+15]) using polynomially secure iO (and ELFs). piO schemes can be used to obfuscate *probabilistic* (i.e., randomized) programs, and are currently the only way to obtain, e.g., fully homomorphic encryption (FHE) schemes without circular security assumptions [Can+15]. However, the only previous construction of piO schemes required subexponentially secure iO [Can+15]. Hence, our construction yields the first FHE scheme from polynomially secure iO (and ELFs). Similarly, we can turn the assumption of subexponentially secure iO into polynomially secure iO (plus ELFs) in the construction of spooky encryption from [Dod+16].

[4] That means that our final schemes depend on ELFs, which are currently only known to be instantiable from exponential assumptions. However, we stress that ELFs can be built from exponential variants of very standard assumptions, such as the decisional Diffie-Hellman (DDH) assumption.

Both FHE and spooky encryption are quite powerful primitives, and we obtain several "spin-off results" by revisiting their implications. For instance, when instantiating the piO-based FHE construction of [Can+15] with our piO scheme and a suitable public-key encryption scheme, we obtain a fully key-dependent message (KDM) secure public-key encryption scheme from (polynomially secure) iO and the exponentially secure DDH assumption (and no further assumptions). Under the same assumptions, we obtain multi-key FHE with threshold decryption and function secret sharing schemes from the spooky encryption construction from [Dod+16].

On the plausibility of ELFs. One could argue that we trade one exponential assumption for another, and it is not clear that assuming polynomial iO and exponential DDH is any better than assuming only subexponential iO in the first place. Seconding Zhandry [Zha16] here, we think that exponential DDH is a realistic assumption that is far more popular, better-investigated, and arguably more plausible than subexponential iO. Much of the currently deployed cryptography relies on (in fact a strong variant of) exponential DDH, because parameters are almost always chosen according to the best known attacks.

On the number of assumptions. Another natural observation is that iO for general circuits is already an exponential family of assumptions in itself (one for each obfuscated circuit). It might seem that this lets the challenge of relying on polynomially secure iO instead of subexponentially secure iO appear less appealing. We make two comments on that.

- First, being an exponential family of assumptions and assuming resistance against subexponential adversaries are orthogonal issues. Many cryptographic assumptions have several dimensions of strengths, and relaxing the assumption in any of these dimensions is desirable.[5] In this work, we make progress in one important dimension. By replacing subexponential iO by polynomial iO plus exponential DDH, we effectively trade an *exponential* number of subexponential hardness assumptions in exchange for a *single* (plausible, well-studied) exponential hardness assumption (plus an exponential family of polynomial hardness assumptions).
- Second, iO being an exponential family of assumptions can be considered an artificial consequence of working on the general notion of iO for *arbitrary circuits*. When using iO in concrete constructions (e.g. in all the constructions described in this paper), one almost never needs to assume iO for all circuits. It usually suffices to assume iO for a constant number of specific circuits (namely those being obfuscated in the construction and the analysis). Hence, iO is a small number of assumptions when used for building a cryptographic primitive.

1.1 Technical overview

The piO construction of Canetti et al. To describe our ideas, it will be helpful to briefly review the work of Canetti et al. [Can+15]. In a nutshell, they define the notion of piO as a way to obfuscate probabilistic programs, and show how to use piO to implement the first FHE scheme without any circular security assumption. Intuitively, where the notion of iO captures that the obfuscation $\mathsf{iO}(P)$ of a *deterministic* program P does not leak anything beyond the functionality of P, piO captures the same for *probabilistic* programs P.[6]

They also show how to implement piO with an indistinguishability obfuscator iO and a pseudorandom function (PRF) F. Namely, in order to obfuscate a probabilistic program P, Canetti et al. obfuscate the *deterministic* program P' that, on input x, runs $P(x)$ with random coins $r = F(K, x)$. Here, K is a PRF key hardcoded into P'. The security proof uses "puncturing" techniques [SW14] and a hybrid argument over all possible P-inputs x. More specifically, for each P-input x, separate reductions to the security of iO and F show that the execution of $P'(x)$ is secure.[7]

[5] For example, if a protocol relies on the subexponential hardness of LWE with exponential modulus-to-noise ratio, it would be desirable to achieve the same while relying only on polynomially secure LWE, even if the modulus-to-noise ratio remains exponential.

[6] This is of course an oversimplification. Also, [Can+15] define several types of piO security that provide a tradeoff between security and achievability.

[7] Again, we are not very specific about the form of desired or assumed security. However, we believe that for this exposition, these specifics do not matter.

This proof strategy is very general and does not need to make any specific assumptions about the structure of P. (In fact, this strategy can be viewed as a specific form of "complexity leveraging", technically similar to the conversion of selective security into adaptive security, e.g., [BB04a].) However, the price to pay is a reduction loss which is linear in the size of the input domain (which usually is exponentially large). In particular, even after scaling security parameters suitably, Canetti et al. still require subexponentially secure iO and PRFs.

More on previous works to remove subexponentiality. There are a number of known ways to deal with subexponential reduction losses due to complexity leveraging (or related techniques). For instance, various semi-generic (pre-iO) techniques seek to achieve adaptive security (for different primitives) by establishing an algebraic or combinatorial structure on the used inputs [BB04b; Wat05; HK08; HW09], and can sometimes be adapted to the iO setting [HSW14]. But like the already-mentioned, somewhat more general approaches [Gar+17; LZ17], these works make specific assumptions about the structure of the involved computations.

A somewhat more general approach (that works for more general classes of programs) was outlined by Zhandry [Zha16], who introduces the notion of "extremely lossy functions" (ELFs). Intuitively, an ELF is an injective function G that can be switched into an "extremely lossy mode", in which its range is polynomially small. Such an ELF can sometimes be used to "preprocess" inputs in a cryptographic scheme, with the following benefit: a security reduction can switch the ELF to extremely lossy mode, so that only a polynomial number of (preprocessed) inputs $G(x)$ need to be considered. This simplifies a potential hybrid argument over all (preprocessed) inputs $G(x)$, and can lead to a polynomial (instead of a subexponential) reduction.

However, trying to apply this strategy to the construction and reduction of Canetti et al. (as sketched above) directly fails. Namely, in their application, inputs will be inputs x to an arbitrary (probabilistic) program P; preprocessing them with an ELF will destroy their structure, and it is not clear how to run P on ELF-preprocessed inputs $G(x)$. Indeed, applying ELFs to realize piO requires fundamentally different techniques.

Main idea: piO with sparsifiable inputs. Instead, we will restrict ourselves to programs P that take as input an element x from a small number of (arbitrary but efficiently samplable) distributions. In other words, all possible inputs x need to be in the range of one of a small number of efficient samplers S_i. As an example, for $i \in \{0,1\}$, sampler S_i could sample ciphertexts C that encrypt plaintext i. Moreover, we require that all inputs to a program P to be obfuscated are at some point actually sampled from some S_i according to a certain process.

Obfuscating a given probabilistic program P (that takes as inputs one or more x as above) now consists of two steps:

1. First, we *encode* all inputs x, in the sense that we compile S_i to attach a "certificate" aux to x. This certificate aux guarantees that x has really been sampled using S_i. Furthermore, the compiled sampler S_i uses preprocessed random coins of the form $G(r)$ (instead of r) for an ELF G. (When G is in injective mode, this does not affect the distribution of sampled x.) The certificate aux additionally guarantees this choice of random coins.[8]
2. Second, we produce the actual obfuctation of the probabilistic program P as follows. We use an indistinguishability obfuscator iO to obfuscate the following (deterministic) variant P' of P: on inputs $x_1, \ldots, x_\ell$ with certificates $\mathsf{aux}_1, \ldots, \mathsf{aux}_\ell$, P' first checks the certificates aux_i and aborts if one of them is invalid. Next, P' runs $P(x_1, \ldots, x_\ell)$, with random coins $F(K, (x_i)_{i=1}^{\ell})$ for a PRF F and a hardcoded PRF key K. Finally, P' outputs P's output.

Maybe the most important property of this setup is that now the sets of inputs x_i are "sparsifiable" in the following sense. If we set G to extremely lossy mode, then only a polynomial number of different random coins r can occur. Hence, each S_i will output one of only a small number of possible samples (e.g., encryptions C generated with random coins from a small set). In that sense, the set of possible inputs x_i to P has been "sparsified", and a hybrid argument over all possible inputs as in [Can+15] is possible with polynomial loss.

We stress that our technique of applying ELFs fundamentally differs from [Zha16]. In [Zha16], the constructed primitive itself ensures that G is applied on all inputs. When approaching the challenge of constructing piO, however, the input to the primitive must externally be sampled using random coins that are preprocessed with G. This process is not under the control of the primitive and therefore requires a mechanism

[8] Looking ahead, this "certificate" will be implemented using a NIZK in our construction.

certifying that inputs are generated according to this specific process. We implement this mechanism using the combination of compiling the sampler for the input distribution into a "certifying sampler" (step 1) and restricting correctness of the obfuscated program (step 2).

Surprisingly, our piO scheme achieves the notion of "dynamic-input piO" [Can+15], a very strong variant of piO security. On a high level, dynamic-input piO guarantees indistinguishability between obfuscations of probabilistic programs as long as their output distributions on adversarially chosen inputs are indistinguishable. This constitutes a very strong requirement and, in fact, implies differing-inputs obfuscation [Bar+01; Ana+13], a notion for which strong impossibility results exist [Gar+14b; BSW16]. However, our obfuscator produces circuits which are only required to work on inputs certifiably generated according to a specific process. Hence, our piO scheme enjoys a restricted form of correctness. This enables us to circumvent the impossibility results [Gar+14b; BSW16].

Applications. One obvious question is of course how restrictive our assumption on input domains really is. We show that our assumptions apply to two existing piO-based constructions, with a number of interesting consequences.

First, we revisit the piO-based construction of fully homomorphic encryption from [Can+15]. Here, piO is used to obfuscate the FHE evaluation algorithm that takes two ciphertexts (say, of two bit plaintexts b_0 and b_1) as input, and outputs a ciphertext of the NAND of the two plaintexts (i.e., $b_0 \overline{\wedge} b_1$). If we set S_b to be a sampler that samples an encryption of b, this setting perfectly fits our scheme. Hence, we obtain first a leveled homomorphic encryption (LHE) scheme, and from this an FHE scheme using the high-level strategy from [Can+15]. Hence, putting this together with our piO construction, we obtain an FHE scheme from polynomially secure iO and an ELF (and no further assumptions).

We note that the above FHE scheme is also fully key-dependent message (KDM, see [BRS03]) secure when implemented with a suitable basic public-key encryption scheme (such as the DDH-based scheme of [Bon+08]). In that case, the FHE is secure even when an encryption of its own secret key $C_{\mathsf{sk}} = \mathsf{Enc}(\mathsf{pk}, \mathsf{sk})$ is public. However, such an encryption C_{sk} can be transformed into an encryption $\mathsf{Enc}(\mathsf{pk}, f(\mathsf{sk}))$ of an arbitrary function of sk thanks to the fully homomorphic properties of the FHE scheme. This leads to a conceptually very simple fully KDM-secure encryption scheme from polynomial assumptions (and ELFs). (We stress that we do not claim novelty for this observation. The connection between FHE and KDM security has already been observed in [Bar+10], and [Dod+16] have observed that the FHE construction of Canetti et al. preserves interesting properties of the underlying encryption scheme. However, [Dod+16] do not explicitly mention KDM security, and we find these consequences interesting enough to point out.)

As our second application, we consider spooky encryption (with CRS) introduced by Dodis et al. [Dod+16]. Intuitively, a spooky encryption scheme features a particular type of homomorphism in a multi-key, multi-ciphertext setting. More precisely, given ciphertexts $\{c_i = \mathsf{Enc}(\mathsf{pk}_i, x_i)\}_i$, a spooky encryption scheme allows to produce ciphertexts $\{c'_i\}_i$ with $y_i = \mathsf{Dec}(\mathsf{sk}_i, c'_i)$ such that certain so-called "spooky" relations between between the x_i's and the y_i's hold. An important subclass of spooky relations allows to ensure that the y_i's are random subject to $\sum_i y_i = f(x_1, \ldots, x_n)$, for any polynomial-time computable function f. Dodis et al. show that spooky encryption implies (among other things) function secret sharing, and they give a piO-based instantiation of spooky encryption (without the need of a CRS). At the heart of their construction is an obfuscated public "spooky evaluation" algorithm with a hardcoded decryption key. Since this algorithm also takes ciphertexts (and a public key) as input, its input domain can be sparsified much like in the FHE case.

In contrast to the FHE application, however, the spooky encryption application contains more technical subtleties. In particular, some inputs to the "spooky evaluation" algorithm may depend on other inputs, and hence sparsifying inputs needs to proceed in a certain order. The main difficulty here is to find a suitably flexible definition of sparsification; we omit the details in this overview. We note that our results of course also yield all applications of spooky encryption, only from polynomially secure iO (and ELFs). In particular, we obtain a simple protocol for function secret sharing for all functions (with additive reconstruction) from these assumptions [BGI15].

We believe that our new notion of obfuscation will prove useful in other applications; for example, it would likely allow to improve the recent result of [Can+17], which constructed CCA1-secure FHE from subexponentially secure iO.

Follow-up work. In the recent work [DN18], Döttling and Nishimaki define the notion universal proxy re-encryption (UPRE). UPRE schemes allow a proxy to convert any ciphertext under any public key of any existing PKE scheme into a ciphertext under any public key of any possibly different existing PKE scheme. [DN18] instantiate UPRE based on probabilistic IO due to [Can+15]. UPRE for all PKE schemes (including non re-randomizable ones) requires dynamic-input pIO, which implies differing-inputs obfuscation. However, [DN18] observe that our notion of doubly-probabilistic IO suffices which yields an instantiation of UPRE for all PKE schemes based on polynomial IO and exponential DDH.

Organization. In Section 2, we introduce our notations and recall standard preliminaries. Section 3 formally introduces our new variant of piO, called dpiO. Section 4 shows how to instantiate dpiO using polynomially secure iO and ELFs. Eventually, in Section 5 and we revisit the construction of leveled homomorphic encryption from [Can+15], using dpiO instead of piO. In , we revisit the construction of spooky encryption from [Dod+16] using dpiO and analyze our new construction.

Acknowledgments

We would like to thank the anonymous reviewers for many helpful comments.

2 Preliminaries

Notations. Throughout this paper, λ denotes the security parameter. For a natural number $n \in \mathbb{N}$, $[n]$ denotes the set $\{1, \ldots, n\}$. A probabilistic polynomial time algorithm (PPT, also denoted *efficient* algorithm) runs in time polynomial in the (implicit) security parameter λ. A positive function f is *negligible* if for any polynomial p there exists a bound $B > 0$ such that, for any integer $k \geq B$, $f(k) \leq 1/|p(k)|$. An event depending on λ occurs with *overwhelming probability* when its probability is at least $1 - \mathsf{negl}(\lambda)$ for a negligible function negl. Given a finite set S, the notation $x \xleftarrow{\$} S$ means a uniformly random assignment of an element of S to the variable x. The notation $\mathcal{A}^{\mathcal{O}}$ indicates that the algorithm $\mathcal{A}$ is given oracle access to $\mathcal{O}$. Let $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \geq 0}$ be a family of sets of (possibly randomized) circuits, where $\mathcal{C}_\lambda$ contains circuits of size $\mathsf{poly}(\lambda)$. A circuit sampler for $\mathcal{C}$ is a distribution ensemble $D = \{D_\lambda\}_{\lambda \geq 0}$, such that D_λ ranges over triples (C_0, C_1, z) with $(C_0, C_1) \in \mathcal{C}_\lambda^2$ of identical size and taking inputs of the same length, and $z \in \{0,1\}^{\mathsf{poly}(\lambda)}$. A class of samplers $\mathbf{S}$ is a set of circuit samplers for $\mathcal{C}$.

2.1 Indistinguishability Obfuscation for General Samplers

Indistinguishability obfuscation (iO) for general samplers was introduced in [Can+15]. This notion generalizes the classical notion of iO introduced in [Bar+01]. Informally, an iO scheme for a sampler D allows to obfuscate circuits sampled with D so that, given a sample (C_0, C_1) from D, $\mathsf{iO}(C_0) \approx \mathsf{iO}(C_1)$. The standard notion of iO is recovered by considering samplers over functionally equivalent deterministic circuits of the same size. Stronger notions of obfuscation, denoted piO, can be defined for samplers over *probabilistic* circuits, satisfying various indistinguishability notions. We recall below the general definition of [Can+15] of piO for a class of samplers (using a different notion of correctness defined in [Dod+16]). The original correctness definition states that an efficient adversary given oracle access to either the original circuit or the obfuscation (with the restriction that no input can be queried twice), can not tell the difference.

Definition 1 (piO for a Class of Samplers [Can+15; Dod+16]). *A uniform PPT machine piO is an* indistinguishability obfuscator *for a class of samplers $\mathbf{S}$ over a family $\mathcal{C} = \{C_\lambda\}_{\lambda \geq 0}$ of possibly randomized circuits if it satisfies the following conditions:*

Correctness. *For every security parameter λ, every circuit $C \in C_\lambda$, and every input x, the distributions of $C(x)$ over the random coins of C and of $\mathsf{piO}(1^\lambda, C)(x)$ over the random coins of the obfuscator are identical.*

μ-Indistinguishability. *For every sampler $D = \{D_\lambda\}_{\lambda \geq 0} \in \mathbf{S}$, and for every non-uniform PPT machine $\mathcal{A}$, it holds that*

$$\begin{aligned} &|\Pr[(C_0, C_1, z) \xleftarrow{\$} D_\lambda : \mathcal{A}(C_0, C_1, \mathsf{piO}(1^\lambda, C_0), z) = 1] \\ &- \Pr[(C_0, C_1, z) \xleftarrow{\$} D_\lambda : \mathcal{A}(C_0, C_1, \mathsf{piO}(1^\lambda, C_1), z) = 1]| \leq \mu(\lambda). \end{aligned}$$

We remark that the construction of piO from [Can+15] satisfies this notion of correctness if instantiated with a perfect puncturable PRF, see Definition 4. Note that this does not extend to multiple evaluations of the obfuscated circuit. Further, note that this notion of correctness implies that the obfuscated circuit respects the support of the original circuit.

To recover the standard notion of iO, we introduce the class $\mathbf{S}^{\mathsf{eq}}$ of samplers for functionally equivalent (possibly randomized) circuits, *i.e.*, samplers over triplets (C_0, C_1, z) such that $|C_0| = |C_1|$, and for any input x and random coin r, $C_0(x;r) = C_1(x;r)$. The standard iO notion is obtained by considering piO over the subclass $\mathbf{S}^{\mathrm{det}} \subset \mathbf{S}^{\mathsf{eq}}$ of samplers for deterministic functionally equivalent circuits. We denote by $\mathsf{Adv}_{\mathsf{iO}}(\mathcal{A})$ the advantage of a PPT adversary $\mathcal{A}$ in distinguishing between the obfuscation of functionaly equivalent deterministic circuits.

The work of [Can+15] introduced four types of samplers over probabilistic circuits, which define four corresponding variants of piO: dynamic-input piO, worst-case piO, memoryless worst-case piO, and $\mathsf{X\text{-}Ind}$ piO. Informally, a dynamic-input sampler is required to output (possibly randomized) circuits C_0, C_1 such that the output of these circuits on a dynamically chosen input is computationally indistinguishable. The corresponding notion, dynamic-input piO, is the strongest notion defined in [Can+15] and a randomized equivalent of the notion of differing-input obfuscation. Therefore, it inherits the implausibility results of differing-input obfuscation for general circuits [Gar+14b; BSW16]. On the other hand, [Can+15] shows that the weaker notion $\mathsf{X\text{-}Ind}$ piO can be realized from subexponentially secure iO (and subexponentially secure one-way functions). Below, we recall the notion of dynamic-input samplers and dynamic-input piO from [Can+15].

2.2 Dynamic-Input Samplers

Definition 2 (Dynamic-Input Indistinguishable Samplers [Can+15]). *The class $\mathbf{S}^{\mathsf{d\text{-}Ind}}$ of dynamic-input samplers for a circuit family $\mathcal{C}$ contains all circuits samplers $D = \{D_\lambda\}_{\lambda\in\mathbb{N}}$ for $\mathcal{C}$ with the following properties: for every non-uniform PPT $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the advantage $\mathsf{Adv}_{\mathsf{d\text{-}Ind}}(\mathcal{A}) := \Pr[\mathsf{Exp\text{-}d\text{-}Ind}_{\mathcal{A}}(\lambda) = 1] - \frac{1}{2}$ of $\mathcal{A}$ in the experiment $\mathsf{Exp\text{-}d\text{-}Ind}$ represented in Figure 1 is negligible.*

Experiment $\mathsf{Exp\text{-}d\text{-}Ind}_{\mathcal{A}}(\lambda)$

$(C_0, C_1, z) \xleftarrow{\$} D_\lambda$

$(x, \mathsf{st}) \xleftarrow{\$} \mathcal{A}_1(C_0, C_1, z)$ // the challenge input is chosen dynamically

$y \xleftarrow{\$} C_b(x)$ for $b \xleftarrow{\$} \{0,1\}$

$b' \xleftarrow{\$} \mathcal{A}_2(\mathsf{st}, C_0, C_1, z, x, y)$

return $b = b'$

Fig. 1. Experiment $\mathsf{Exp\text{-}d\text{-}Ind}$ for the indistinguishability property of dynamic-input samplers.

Definition 3 (dynamic-input piO). *A uniform PPT machine is a dynamic-input piO scheme if it is a piO for the class of dynamic-input samplers $\mathbf{S}^{\mathsf{d\text{-}Ind}}$ over $\mathcal{C}$ that includes all randomized circuits.*

Note that the class $\mathbf{S}^{\mathsf{eq}}$ of samplers for functionally equivalent circuits that we defined previously, is a subclass of $\mathbf{S}^{\mathsf{d\text{-}Ind}}$: any sampler for triples (C_0, C_1, z) where C_0 and C_1 are functionally equivalent is trivially a dynamic-input sampler.

2.3 Puncturable Pseudorandom Function

A pseudorandom function (PRF) originally introduced in [GGM84] is a tuple of PPT algorithms $\mathsf{F} = (\mathsf{F.KeyGen}, \mathsf{F.Eval})$. Let $\mathcal{K}$ denote the key space, $\mathcal{X}$ denote the domain, and $\mathcal{Y}$ denote the range. The key generation algorithm $\mathsf{F.KeyGen}$ on input of 1^λ, outputs a random key from $\mathcal{K}$ and the evaluation algorithm $\mathsf{F.Eval}$ on input of a key K and $x \in \mathcal{X}$, evaluates the function $F\colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$. The core property of PRFs is that, on a random choice of key K, no probabilistic polynomial-time adversary should be able to distinguish $F(K, \cdot)$ from a truly random function, when given black-box access to it. Puncturable PRFs (pPRFs) have the additional property that some keys can be generated *punctured* at some point, so that they allow to evaluate the PRF at all points except for the punctured point. As observed in [BW13; BGI14; Kia+13], it is possible to construct such punctured keys for the original construction from [GGM84], which can be based on any one-way functions [Hås+99].

Definition 4 (Puncturable Pseudorandom Function [BW13; BGI14; Kia+13]). *A* puncturable pseudorandom function (pPRF) *with punctured key space $\mathcal{K}_p$ is a triple of PPT algorithms* $(\mathsf{F.KeyGen}, \mathsf{F.Punct}, \mathsf{F.Eval})$ *such that*

- $\mathsf{F.KeyGen}(1^\lambda)$ *outputs a random key* $K \in \mathcal{K}$,
- $\mathsf{F.Punct}(K, x)$, *on input* $K \in \mathcal{K}$, $x \in \mathcal{X}$, *outputs a punctured key* $K\{x\} \in \mathcal{K}_p$,
- $\mathsf{F.Eval}(K', x')$, *on input a key* K' *(punctured or not), and a point* x', *outputs an evaluation of the PRF.*

We require F *to meet the following conditions:*

Functionality preserved under puncturing. *For all* $\lambda \in \mathbb{N}$, *for all* $x \in \mathcal{X}$,

$$\Pr[K \xleftarrow{\$} \mathsf{F.KeyGen}(1^\lambda), K\{x\} \xleftarrow{\$} \mathsf{F.Punct}(K, x) : \forall x' \in \mathcal{X} \setminus \{x\} : \mathsf{F.Eval}(K, x') = \mathsf{F.Eval}(K\{x\}, x')] = 1.$$

Pseudorandom at punctured points. *For all PPT adversaries* $\mathcal{A}$,

$$\mathsf{Adv}_{\mathsf{s\text{-}cPRF}}(\mathcal{A}) := \Pr[\mathsf{Exp\text{-}s\text{-}pPRF}_{\mathcal{A}}(\lambda) = 1] - \frac{1}{2}$$

is negligible, where Exp-s-cPRF *is represented Figure 2.*

We call a pPRF F perfect, *if the distribution* $\{\mathsf{F.Eval}(K, x) \mid K \xleftarrow{\$} \mathsf{F.KeyGen}(1^\lambda)\}$ *is identical to the uniform distribution over* $\mathcal{Y}$, *for all inputs* $x \in \mathcal{X}$.[9]

Definition 4 corresponds to a selective security notion for puncturable pseudorandom functions; adaptive security can also be considered, but will not be required in our work. For ease of notation we often write $F(\cdot, \cdot)$ instead of $\mathsf{F.Eval}(\cdot, \cdot)$.

Experiment $\mathsf{Exp\text{-}s\text{-}pPRF}_{\mathcal{A}}(\lambda)$

$(x^*, \mathsf{state}) \xleftarrow{\$} \mathcal{A}(1^\lambda)$

$K \xleftarrow{\$} \mathsf{F.KeyGen}(1^\lambda)$, $K\{x^*\} \xleftarrow{\$} \mathsf{F.Punct}(K, x^*)$

$b \xleftarrow{\$} \{0,1\}$, $y_0 \leftarrow \mathsf{F.Eval}(K, x^*)$, $y_1 \xleftarrow{\$} \mathcal{Y}$

$b' \xleftarrow{\$} \mathcal{A}(\mathsf{state}, K\{x^*\}, y_b)$

return $b = b'$

Fig. 2. Selective security game for puncturable pseudorandom functions.

2.4 Extremely Lossy Function

In this section we present extremely lossy functions (ELFs) introduced in [Zha16]. ELFs are an extremely powerful primitive for complexity absorption allowing to replace subexponential or even exponential security assumptions with polynomial ones. Informally, an ELF is a function that can be generated in two different modes: an injective mode and an extremely lossy mode. In injective mode, the range of the ELF has exponential size whereas the range comprises only polynomially many elements in extremely lossy mode.

Definition 5 (Extremely Lossy Function [Zha16]). *An extremely lossy function* ELF *is an algorithm* ELF.Gen *which, on input* (M, r), *where* M *is an integer and* $r \in [M]$, *outputs the description of a function* $G\colon [M] \to [N]$ *such that*

- G *can be computed in time* $\mathsf{poly}(\log M)$
- *If* $r = M$, G *is injective with overwhelming probability (in* $\log M$*) over the randomness of* $\mathsf{ELF.Gen}(M, M)$*;*
- *For any* $r \in [M]$, $|G([M])| < r$ *with overwhelming probability (in* $\log M$*) over the randomness of* $\mathsf{ELF.Gen}(M, r)$*;*
- ***Indistinguishability:*** *For any large enough* M, *any polynomial* P, *and any inverse polynomial function* δ, *there exists a polynomial* Q *such that for any adversary* $\mathcal{A}$ *running in time at most* $P(\log M)$ *and any* $r \in [Q(\log M), M]$, *the advantage of* $\mathcal{A}$ *in distinguishing* $\mathsf{ELF.Gen}(M, M)$ *from* $\mathsf{ELF.Gen}(M, r)$ *is bounded by* $\delta(\log M)$.

In addition, we will consider extremely lossy functions satisfying *strong regularity*, as defined below.

[9] Given any pPRF F', we can build a perfect pPRF F by sampling two keys $K_1 \xleftarrow{\$} \mathsf{F}'.\mathsf{KeyGen}(1^\lambda)$ and $K_2 \xleftarrow{\$} \mathcal{Y}$ in the key generation algorithm and defining the evaluation algorithm to output $\mathsf{F}'.\mathsf{Eval}(K_1, x) \oplus K_2$ on input of x, see[Dod+16].

Definition 6 (Strong regularity). *An* ELF *is strongly regular if for any (polynomial) r, the distribution $\{x \xleftarrow{\$} [M] : G(x)\}$ is statistically close to uniform over $G([M])$, with overwhelming probability over the choice of $G \xleftarrow{\$} \mathsf{ELF.Gen}(M, r)$.*

We note that, if an ELF is strongly regular, it is possible to efficiently enumerate its image: the set of values obtained by evaluating an ELF on $\lambda r \log r$ random inputs, where r is a bound on the size of its image, contains the entire image of the ELF with overwhelming probability.

Instantiating ELFs. A construction of strongly regular extremely lossy function is given in [Zha16]. It can be based on the exponential hardness of the decision Diffie-Hellman assumption (or any of its variants, such as the decision linear assumption), which we denote eDDH. The eDDH assumption for a group generator GroupGen (which generates a tuple $(\mathbb{G}, p, g)$ where $\mathbb{G}$ is a group, p is its order, and g is a generator of $\mathbb{G}$) states that there exists a polynomial q such that for any time bound t and probability ε, denoting $\kappa \leftarrow \log q(t, 1/\varepsilon)$, any adversary $\mathcal{A}$ running in time at most t has advantage at most ε in distinguishing the following distributions:

$$\{(\mathbb{G}, p, g) \xleftarrow{\$} \mathsf{GroupGen}(1^\kappa), (a, b, c) \xleftarrow{\$} \mathbb{Z}_p^3 : (\mathbb{G}, g, g^a, g^b, g^c)\},$$
$$\{(\mathbb{G}, p, g) \xleftarrow{\$} \mathsf{GroupGen}(1^\kappa), (a, b) \xleftarrow{\$} \mathbb{Z}_p^2 : (\mathbb{G}, g, g^a, g^b, g^{ab})\}.$$

As noted in [Zha16], groups based on elliptic curves are plausible candidates for groups where this assumption holds: in practical instantiations of DDH over elliptic curves, the size of the group is chosen assuming that the best attack takes time $O(\sqrt{p})$, hence disproving eDDH (which amounts to showing that there is an attack which takes time less than p^c for any constant c) would have considerable practical implications. Furthermore, relying on some form of exponential hardness assumption seems necessary, as a construction from polynomial hardness only would have surprising complexity-theoretic implications. More precisely, given access to only some super-logarithmic amount of non-determinism (i.e. $\omega(\log \log M)$ bits, where $[M]$ is the domain of the ELF), it is easy to distinguish between injective and lossy mode of the ELF. This is due to the fact that in lossy mode, the codomain of G has only polynomial size which means that the restriction of G to the set $D = [2^{\omega(\log \log M)}]$ (having super-polynomial cardinality) is guaranteed to have a collision (which is not the case in injective mode), and using only $\omega(\log \log M)$ bits of non-determinism this collision can be guessed.

2.5 Non-interactive Zero-Knowledge proof system

A non-interactive zero-knowledge (NIZK) proof system for a language L with witness relation R enables to prove in a non-interactive manner that some statements are in L without leaking information about corresponding witnesses. NIZK proof systems were originally introduced in [BFM88].

Definition 7 (Non-interactive zero-knowledge proof system [GOS06]). *A* non-interactive zero-knowledge (NIZK) proof system *for a language $L \in \mathsf{NP}$ (with witness relation R) is a tuple of PPT algorithms $\mathsf{NIZK} = (\mathsf{NIZK.Setup}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Verify})$ such that* NIZK.Setup *is a common reference string generation algorithm,* NIZK.Prove *is a proving algorithm* NIZK.Verify *is a (deterministic) verification algorithm.*

- $\mathsf{NIZK.Setup}(1^\lambda)$ *outputs a common reference string* crs.
- $\mathsf{NIZK.Prove}(\mathsf{crs}, x, w)$*, on input* crs*, a statement x and a witness w, outputs a proof π.*
- $\mathsf{NIZK.Verify}(\mathsf{crs}, x, \pi)$*, on input* crs*, a statement x and a proof π, outputs either 1 or 0.*

We require NIZK *to meet the following properties:*

Perfect completeness. *For every $(x, w) \in R$, we have that*

$$\Pr[\mathsf{crs} \xleftarrow{\$} \mathsf{NIZK.Setup}(1^\lambda),\ \pi \xleftarrow{\$} \mathsf{NIZK.Prove}(\mathsf{crs}, x, w) : \mathsf{NIZK.Verify}(\mathsf{crs}, x, \pi) = 1] = 1.$$

Statistical soundness. *For every $x \notin L$ with $|x| = \lambda$ and every (possibly unbounded) adversary $\mathcal{A}$, we have that*

$$\Pr[\mathsf{crs} \xleftarrow{\$} \mathsf{NIZK.Setup}(1^\lambda), \pi \xleftarrow{\$} \mathcal{A}(\mathsf{crs}, x) : \mathsf{NIZK.Verify}(\mathsf{crs}, x, \pi) = 1] < 2^{-\lambda}.$$

Computational zero-knowledge. *There exists a PPT algorithm* $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ *such that for every PPT adversary* $\mathcal{A}$*,*

$$\mathsf{Adv}_{ZK}(\mathcal{A}) := |\Pr\left[\mathsf{crs} \xleftarrow{\$} \mathsf{NIZK.Setup}(1^\lambda) \colon \mathcal{A}^{\mathsf{NIZK.Prove}(\mathsf{crs},\cdot,\cdot)}(\mathsf{crs}) = 1\right] - \Pr\left[(\mathsf{crs},\tau) \xleftarrow{\$} \mathsf{Sim}_0(1^\lambda) \colon \mathcal{A}^{\mathsf{Sim}_1'(\mathsf{crs},\tau,\cdot,\cdot)}(\mathsf{crs}) = 1\right]|$$

is negligible in λ*, where* $\mathsf{Sim}_1'(\mathsf{crs},\tau,x,w)$ *returns* $\mathsf{Sim}_1'(\mathsf{crs},\tau,x)$ *only if* $(x,w) \in R$*.*

For simplicity in the analysis we use a NIZK proof system that satisfies the following property: with overwhelming probability over the coins of $\mathsf{NIZK.Setup}(1^\lambda)$, there does not exist any pair (x,π) such that $x \notin L$ and $\mathsf{NIZK.Verify}(\mathsf{crs},x,\pi) = 1$. We call a NIZK that satisfies this property *almost perfectly sound*. We note that there is a simple folklore method which allows to construct an almost perfectly sound NIZK proof system starting from any statistically sound NIZK proof system. Consider a $2^{-\lambda}$-statistically sound NIZK proof system, for statements $x \in \{0,1\}^n$, for some polynomial $n = n(\lambda)$. Using parallel repetitions, the soundness of the proof system can be amplified to $2^{-\lambda-n}$.[10] Then, it necessarily holds that for all possible crs except a $2^{-\lambda}$ fraction of them, there does not exist any pair (x,π) where $x \notin L$ and π is an accepting proof. To realize this, let E_x^{crs} denote the event that there exists a proof π such that $\mathsf{NIZK.Verify}(\mathsf{crs},x,\pi) = 1$. Then, by a union bound argument, $\Pr_{\mathsf{crs}}[\exists x \in \{0,1\}^n \setminus L \colon E_x^{\mathsf{crs}}] \leq \sum_{x\in\{0,1\}^n\setminus L} \Pr_{\mathsf{crs}}[E_x^{\mathsf{crs}}] \leq 2^n \cdot 2^{-\lambda-n}$. Hence, the NIZK proof system obtained via parallel repetitions is almost perfectly sound.

In [BP15] Bitansky et al. showed that statistically sound NIZK proof systems can be obtained from polynomially secure indistinguishability obfuscation in conjunction with polynomially secure one-way functions.

3 Indistinguishability Obfuscation of Probabilistic Circuits over Distributions of Inputs

We first define the notion of a *sampler with input*. A sampler with input is a family of PPT algorithms which, on input x, sample from some distribution $\mathcal{D}_x$. This notion is convenient to capture the fact that, in many scenarios, the inputs to an obfuscated (probabilistic) circuit are sampled from some distribution $\mathcal{D}_x$, where x is some private input of a player.

Definition 8 (Sampler with Input). *We say that* $\mathcal{SI} = \{\mathcal{SI}_\lambda\}_{\lambda\in\mathbb{N}}$ *is a family of* samplers with input*, with input domain* $\mathcal{I} = \{\mathcal{I}_\lambda\}_{\lambda\in\mathbb{N}}$*, if for any* $\lambda \in \mathbb{N}$*,* $\mathcal{SI}_\lambda$ *is a set of probabilistic algorithms running in polynomial time (in* 1^λ*) with input domain* $\mathcal{I}_\lambda$ *such that for any* $S \in \mathcal{SI}_\lambda$*, and* $x \in \mathcal{I}_\lambda$*,* $S(x)$ *samples from* $\{0,1\}^\lambda$*.*

3.1 Doubly-Probabilistic Indistinguishability Obfuscation

Below, we define a variant of indistinguishability obfuscation, that takes into account the fact that in many applications, obfuscated (probabilistic) circuits might only have to be evaluated on inputs coming from specific distributions. This is formalized by defining an encoding procedure for a sampler with input, which additionally produces auxiliary material that an obfuscated circuit can use to verify that its inputs were produced correctly, and by restricting the correctness of the obfuscated circuit to only hold for such well-formed inputs. We also refer to this auxiliary material as "certificate".

However, this approach faces two issues. First, the inputs to an obfuscated circuit might not be sampled "all at once" from a single distribution; rather, they can come from different and independent sources. We capture this behavior by defining *ℓ-source obfuscation*, to account for the fact that different inputs might have been sampled independently. Second, when inputs are sampled by different parties, there might still be interdependencies which must be accounted for. For example, a party might sample an input (e.g. a public key of an encryption scheme), pass it to a second party, who then samples a second input from a distribution that is parametrized by the first input (e.g. a ciphertext under that public key). We handle this possibility by ordering the ℓ inputs to the obfuscated circuit, and by considering a *stateful* sampler with input S: when S is used to generate the i'th sample y_i, it receives in addition to its input a state $\mathsf{stf}(y_1,\ldots,y_{i-1})$, where stf is some fixed efficiently computable *state function* (which depends on the particular application), and the y_j are outputs sampled by the first $i-1$ sources. The state function captures the fact that a particular

[10] That is, for any statement $x \notin L$, the probability $\Pr_{\mathsf{crs}}[\exists\pi \colon \mathsf{NIZK.Verify}(\mathsf{crs},x,\pi) = 1] \leq 2^{-\lambda-n}$.

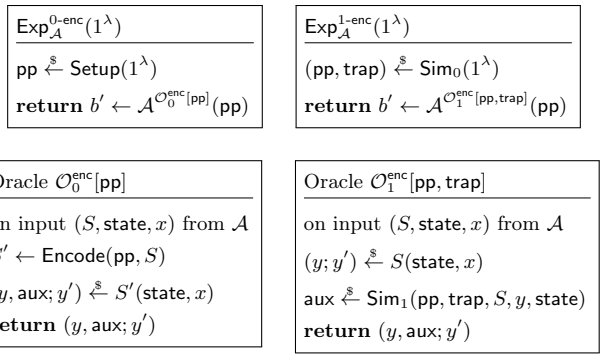

Fig. 3. Experiments $\mathsf{Exp}^{0\text{-enc}}_{\mathcal{A}}(1^\lambda)$ and $\mathsf{Exp}^{1\text{-enc}}_{\mathcal{A}}(1^\lambda)$ for the simulatability of encodings in an ℓ-source dpiO. The PPT algorithm $\mathcal{A}$ can interact polynomially many times with either $\mathcal{O}_0^{\mathsf{enc}}[\mathsf{pp}]$ or $\mathcal{O}_1^{\mathsf{enc}}[\mathsf{pp}, \mathsf{trap}]$. $\mathcal{A}$ wins the experiment when it outputs $b' = b$ in $\mathsf{Exp}^{b\text{-enc}}_{\mathcal{A}}(1^\lambda)$

application might define an arbitrary communication pattern, and specifies which samples a party should have access to when generating his sample.

Additionally, we admit the possibility that a sampler produces some additional correlated output, that will not serve as input to an obfuscated circuit. Hence, there is no need to "certify" this input using the auxiliary information, and we call this output unauthenticated output. Continuing the use case from above, given a sampler producing some public key, the unauthenticated part of that sampler's output could be a corresponding secret key.

Definition 9 (Doubly-Probabilistic Indistinguishability Obfuscation (dpiO)). *Let ℓ be an integer. Let $\{\mathsf{stf}_\lambda : (\{0,1\}^\lambda \cup \{\bot\})^{\ell-1} \to \mathcal{T}_\lambda\}_{\lambda\in\mathbb{N}}$ be a family of efficiently computable functions. Let $\mathcal{SI} = \{\mathcal{SI}_\lambda\}_{\lambda\in\mathbb{N}}$ be a family of samplers with inputs, with input domain $\{\mathcal{T}_\lambda \times \mathcal{I}\}_{\lambda\in\mathbb{N}}$. Let $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ be a family of (probabilistic) circuits, and let $\mathbf{CS}$ be a class of circuit samplers over $\mathcal{C}$. An ℓ-source dpiO scheme for $(\mathsf{stf}, \mathcal{SI}, \mathcal{C}, \mathbf{CS})$ is a triple of PPT algorithms $(\mathsf{Setup}, \mathsf{Encode}, \mathsf{Obfuscate})$ such that*

- $\mathsf{Setup}(1^\lambda)$*, on input the security parameter (in unary), outputs public parameters* pp*;*
- $\mathsf{Encode}(\mathsf{pp}, S)$*, on input the public parameters* pp*, and a sampler with input $S \in \mathcal{SI}_\lambda$, outputs an encoded sampler S';*
- $\mathsf{Obfuscate}(\mathsf{pp}, S, C)$*, on input public parameters* pp*, a sampler with input $S \in \mathcal{SI}_\lambda$, and a circuit $C \in \mathcal{C}_{\ell\lambda}$, outputs a circuit C' of size $\mathsf{poly}(\lambda, |C|)$. We call C' an* obfuscation of C with respect to S*.*

We further assume that the outputs of S on any input (state, x) is of the form $(y; y')$ (looking ahead, we will call y the authenticated output*, and y' the* unauthenticated output*). The scheme should satisfy the three properties given below.*

Informally, the first security requirement ensures that, on any (adversarially chosen) input x, state state, and sampler with input S, the sampler S' obtained by encoding S outputs samples of the form $(y, \mathsf{aux}; y')$ where $(y; y')$ is distributed as an output of $S(\mathsf{state}, x)$, and aux does not leak any non-trivial information about the inputs. This is formalized by requiring the existence of a simulator that can simulate aux given only y.

Definition 10 (Simulatability of Encodings). *An ℓ-source dpiO scheme for $(\mathsf{stf}, \mathcal{SI}, \mathcal{C}, \mathbf{CS})$ satisfies* simulatability of encodings *if for any large enough λ and any (stateful) PPT adversary $\mathcal{A}$, there exists a PPT simulator $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ such that the advantage of $\mathcal{A}$ in distinguishing the experiments $\mathsf{Exp}^{0\text{-enc}}$ and $\mathsf{Exp}^{1\text{-enc}}$ represented on Figure 3 is negligible. We denote by $\mathsf{Adv}_{\mathsf{enc}}(\mathcal{A})$ the advantage of $\mathcal{A}$ in this experiment.*

We now introduce the restricted correctness requirement. Intuitively, it states the following: in an honest scenario, the inputs $(y_1, \ldots, y_\ell)$ should be constructed using the sampler with input S. The restricted correctness property guarantees that if the inputs have indeed been constructed "according to S", then the obfuscated circuit will behave correctly, and its output distribution (taken over the coins of the obfuscator) will be (statistically) indistinguishable from the output distribution of the circuit C (taken over its internal random coins). Note that this statistical indistinguishability does not extend to multiple evaluations. Additionally, when evaluated on such inputs, the obfuscated circuit respects the support of the original circuit.

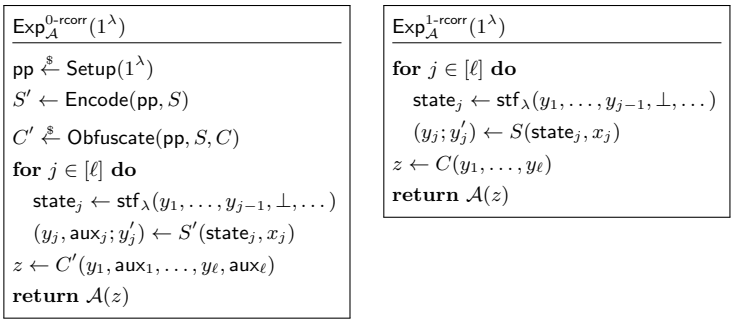

Fig. 4. Experiments $\mathsf{Exp}^{0\text{-rcorr}}_{\mathcal{A}}(1^\lambda)$ and $\mathsf{Exp}^{1\text{-rcorr}}_{\mathcal{A}}(1^\lambda)$ for the restricted correctness property an ℓ-source dpiO. $\mathcal{A}$ wins the experiment when it outputs $b' = b$ in $\mathsf{Exp}^{b\text{-rcorr}}_{\mathcal{A}}(1^\lambda)$ when $b \xleftarrow{\$} \{0,1\}$.

To make this definition meaningful, we need a way to let the obfuscated circuit verify that the inputs are well-formed. Note that we do not want to ensure that they were generated through S with uniformly random coins, but only that they were generated through S with *some* random coins (and some input). To make this verification possible, we let the parties generate their input using the encoded sampler S' instead. This encoded sampler should correctly sample as S, but it will in addition produce auxiliary information which can be used by the obfuscated program to verify that the inputs were honestly constructed (more formally, for a given y, that there exists an input x, coins r, and an unauthenticated part y' such that $(y; y') = S(x; r)$).

A small technicality is that we must allow the sampler with input to depend on state information, to capture the possible interdependencies between the inputs. This means that the auxiliary information will have to certify that an input was generated correctly, with respect to some state that the obfuscated circuit might not have access too (which would prevent it from verifying the certificate). However, this issue disappears by restricting the interdependencies to only involve a state computed from the previous *samples* (as opposed to more complex interdependencies which would involve, for example, the coins used to produce these samples). In this case, the obfuscated circuit can check the certificates in an incremental way: it first checks that y_1 was correctly constructed with respect to the state $\mathsf{st}_\lambda(\bot, \ldots, \bot)$, then it checks that y_2 was correctly constructed with respect to the state $\mathsf{st}_\lambda(y_1, \bot, \ldots, \bot)$, and so on.

Definition 11 (Statistical Restricted Correctness). *An ℓ-source* dpiO *scheme for* $(\mathsf{stf}, \mathcal{SI}, \mathcal{C}, \mathbf{CS})$ *satisfies* restricted correctness *if for any large enough $\lambda \in \mathbb{N}$, any $S \in \mathcal{SI}_\lambda$, $(x_1, \ldots, x_\ell) \in \mathcal{I}^\ell_\lambda$, and $C \in \mathcal{C}_{\ell\lambda}$, the advantage of any (possibly unbounded) adversary $\mathcal{A}$ in distinguishing the experiments* $\mathsf{Exp}^{0\text{-rcorr}}$ *and* $\mathsf{Exp}^{1\text{-rcorr}}$ *represented on Figure 4 is negligible. We denote by* $\mathsf{Adv}_{\mathsf{rcorr}}(\mathcal{A})$ *the advantage of $\mathcal{A}$ in this experiment. Additionally, we require that the encoded sampler and the obfuscated circuit respect the support of the original sampler and the original circuit, respectively. That is for all* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ *and all* $S' \leftarrow \mathsf{Encode}(\mathsf{pp}, S)$ *and all* $C' \leftarrow \mathsf{Obfuscate}(\mathsf{pp}, S, C)$*, we have that for all inputs* (state, x)*,* $S'(\mathsf{state}, x) \in \mathsf{Supp}(S(\mathsf{state}, x))$ *and for all* $(y_1, \mathsf{aux}_1, \ldots, y_\ell, \mathsf{aux}_\ell)$ *produced as in* $\mathsf{Exp}^{0\text{-rcorr}}$*,* $C'(y_1, \mathsf{aux}_1, \ldots, y_\ell, \mathsf{aux}_\ell) \in \mathsf{Supp}(C(y_1, \ldots, y_\ell))$.

We now introduce the indistinguishability notion. It is close in spirit to the standard indistinguishability notion for obfuscation of probabilistic circuits of [Can+15]. However, in our scenario, the security notion must account for the fact that a set of public parameters pp is generated in a setup phase; the indistinguishability property of obfuscated circuits must therefore hold when (polynomially) many circuits are obfuscated with respect to a single string of public parameters. This suggests an oracle-based security notion.

Definition 12 (Indistinguishability with Respect to CS). *An ℓ-source* dpiO *scheme for* $(\mathsf{stf}, \mathcal{SI}, \mathcal{C}, \mathbf{CS})$ *satisfies* indistinguishability with respect to $\mathbf{CS}$ *if for every circuit sampler $D = \{D_\lambda\}_{\lambda \in \mathbb{N}} \in \mathbf{CS}$, for any large enough λ, the advantage of any PPT adversary $\mathcal{A}$ in distinguishing the experiments* $\mathsf{Exp}^{0\text{-ind}}$ *and* $\mathsf{Exp}^{1\text{-ind}}$ *represented on Figure 5 is negligible. We denote by* $\mathsf{Adv}_{\mathsf{ind}}(\mathcal{A})$ *the advantage of $\mathcal{A}$ in this experiment.*

4 Construction

In this section, we will construct an ℓ-source dpiO scheme (for any constant ℓ), for samplers with input over an input domain $\mathcal{I}$ of polynomial size[11], and dynamic-input indistinguishable circuit-samplers. Our construction relies on polynomially-secure indistinguishability obfuscation, a perfect puncturable pseudorandom function, an almost perfectly sound non-interactive zero-knowledge proof system, and an extremely lossy function.

[11] We note that the output domain of such samplers can be of exponential size.

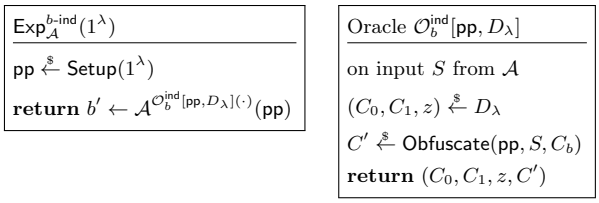

Fig. 5. Experiment $\mathsf{Exp}^{b\text{-}\mathsf{ind}}_{\mathcal{A}}(1^\lambda)$ for the indistinguishability with respect to **CS** in an ℓ-source dpiO. The PPT algorithm $\mathcal{A}$ can interact polynomially many times with $\mathcal{O}^{\mathsf{ind}}_b[\mathsf{pp}, D_\lambda]$. The oracle $\mathcal{O}^{\mathsf{ind}}_b[\mathsf{pp}, D_\lambda]$ is stateful and has (pp, D_λ) hardcoded in its description. $\mathcal{A}$ wins the experiment when it outputs $b' = b$ in $\mathsf{Exp}^{b\text{-}\mathsf{ind}}_{\mathcal{A}}(1^\lambda)$ when $b \xleftarrow{\$} \{0,1\}$.

4.1 Overview

We start by providing a high-level overview of our construction. The Setup procedure generates parameters for the ELF and for the NIZK proof system. To encode a sampler with input S, we define the encoded sampler S' as follows: on input $(\mathsf{state}, x; r)$, S' computes $(y; y') \xleftarrow{\$} S(\mathsf{state}, x; G(r))$ and $\mathsf{aux} \xleftarrow{\$} \mathsf{NIZK.Prove}(y, L^{G,S}_{\mathsf{state}}, (y', x, r))$, and outputs $(y, \mathsf{aux}; y')$. Here, G is the ELF defined by the public parameters, and the language $L^{G,S}_{\mathsf{state}}$ contains all values y for which there exists (y', x, r) such that $(y; y') = S(\mathsf{state}, x, G(r))$. We call *valid input* a value $y \in L^{G,S}_{\mathsf{state}}$. Note that when G is in injective mode, $L^{G,S}_{\mathsf{state}}$ will in general be a trivial language. The simulatability of the encodings directly follows from the injectivity of G, and the zero-knowledge property of the proof system.

We construct the $\mathsf{Obfuscate}$ algorithm for a circuit C as follows (we assume a single source in this overview for simplicity). It first samples a pPRF key K for the pPRF F. Then, it returns an obfuscation of the following circuit: on input (y, aux), run $\mathsf{NIZK.Verify}$ on aux to check that y is a valid input (and output $\perp$ otherwise). Set $r \leftarrow F(K, y)$, and output $C(y; r)$. Restricted correctness follows from the correctness of the NIZK scheme. For indistinguishability between obfuscations of two dynamic-input indistinguishable circuits (C_0, C_1), we follow the standard puncturing strategy of [Can+15]: we proceed through a sequence of hybrids, with successive modifications of the obfuscated circuit. For every possible input y, we construct a sequence of hybrids where the outputs $C_0(y; r)$ are gradually replaced by $C_1(y; r)$. Each replacement relies on the security of the iO scheme, the PRF security, and the dynamic-input indistinguishability of C_0 and C_1.

The main issue of this approach is that the number of possible inputs y (hence the number of hybrids) is exponential – indeed, this is the reason why the piO scheme of [Can+15] requires subexponentially secure primitives (iO and PRF). To get around this issue, we first switch G to an appropriate extremely lossy mode, that the adversary cannot distinguish from the injective mode. Now, the soundness of the NIZK proof system ensures that all valid inputs y are of the form $S(\mathsf{state}, x; G(r))$ for some (x, r) (omitting y' for simplicity). For a given state, the quantity of such values is bounded by the size of the range of G (which is polynomial), times the size of the input domain $\mathcal{I}$. Therefore, in all applications where the inputs to the obfuscated circuit are sampled using private inputs from a small domain, we can base security on polynomially secure iO.

4.2 Construction

For our construction, we employ a perfectly sound NIZK proof system for the following (parametrized) language

$$L^{G,S}_{\mathsf{state}} := \{y \mid \exists (y', x, r) \colon (y; y') = S(\mathsf{state}, x; G(r))\}.$$

Let $\ell \in \mathbb{N}$ be a constant, let $\{\mathsf{stf}_\lambda \colon (\{0,1\}^\lambda \cup \{\perp\})^{\ell-1} \to \mathcal{T}_\lambda\}_\lambda$ be a family of efficiently computable state functions, and let $\mathcal{C} = \{C_\lambda\}_\lambda$ be a family of (randomized) circuits with random space $\{0,1\}^M$ (where $M = M(\lambda)$ is polynomial). Let $\mathcal{SI}$ be a family of samplers with input domain $\mathcal{I}$ of polynomial size. Further, let $\mathbf{S}^{\mathsf{d\text{-}Ind}}$ be the class of dynamic-input indistinguishable samplers (over $\mathcal{C}$).

Theorem 13. *If* ELF *is a strongly regular extremely lossy function,* iO *is a perfectly correct polynomially secure IO scheme,* F *is a polynomially secure perfect puncturable PRF, and* NIZK *is a perfectly sound polynomially zero-knowledge NIZK proof system for the family of languages* $\{L^{G,S}_{\mathsf{state}}\}_{\mathsf{state},G,S}$*, then* $\mathsf{dpiO} = (\mathsf{Setup}, \mathsf{Encode}, \mathsf{Obfuscate})$ *defined in Figure 6 is an ℓ-source dpIO scheme for* $(\mathsf{stf}, \mathcal{SI}, \mathcal{C}, \mathbf{S}^{\mathsf{d\text{-}Ind}})$.

As noted in Section 2.5, almost perfectly correct NIZKs can be constructed from polynomially-secure indistinguishability obfuscation and extremely lossy functions. ELFs also imply the existence of one-way functions, hence of perfect puncturable PRFs [GGM84; Hås+99]. Therefore, we get as corollary:

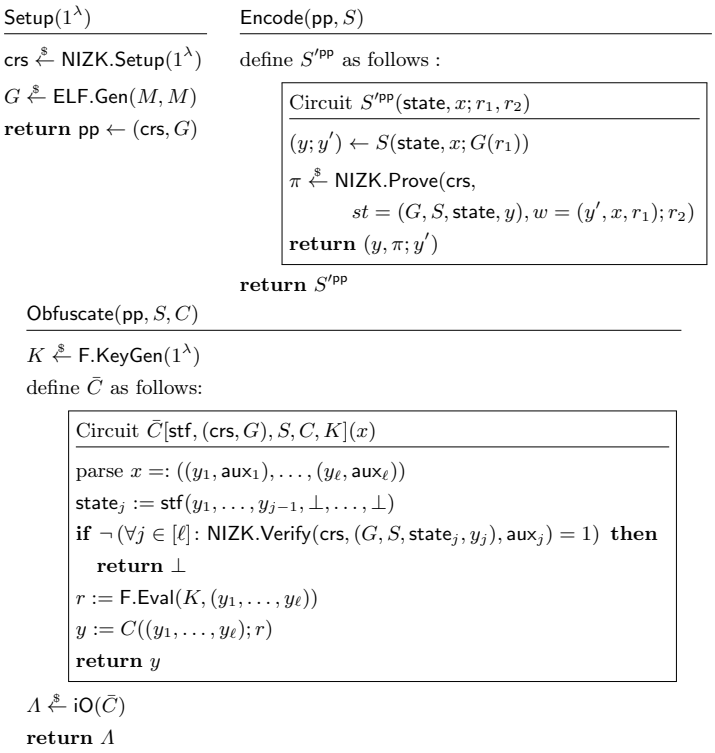

Fig. 6. Construction of ℓ-source dpIO scheme $\mathsf{dpiO} = (\mathsf{Setup}, \mathsf{Encode}, \mathsf{Obfuscate})$.

Corollary 14. *Assuming polynomially-secure indistinguishability obfuscation and extremely lossy functions, there exists (for any constant ℓ) an ℓ-source doubly-probabilistic indistinguishability obfuscation scheme for the class of dynamic-input circuit-samplers, and input-samplers with a polynomial size input domain.*

Proof (of Theorem 13). We prove that dpiO as defined in Figure 6 satisfies simulatability of encodings (cf. Definition 10), statistical restricted correctness (cf. Definition 11), and indistinguishability (cf. Definition 12).

Simulatability of encodings. We prove that there exists a PPT simulator $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ such that for every PPT adversary $\mathcal{A}$, the advantage $\mathsf{Adv}_{\mathsf{enc}}(\mathcal{A})$ is negligible. By the zero-knowledge property of NIZK, there exists a simulator $(\mathsf{NIZK.Sim}_0, \mathsf{NIZK.Sim}_1)$. We construct a simulator $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ as follows:

- Sim_0 produces the CRS using $(\mathsf{crs}, \tau) \xleftarrow{\$} \mathsf{NIZK.Sim}_0(1^\lambda)$, samples the parameters of the ELF G in injective mode, and outputs $\mathsf{pp} := (\mathsf{crs}, G)$ together with $\mathsf{trap} := \tau$.
- Sim_1 on input (pp, trap), a sampler S, a state state, and a value y sampled via $(y; y') \xleftarrow{\$} S(\mathsf{state}, x)$, Sim_1 produces a simulated proof via $\pi \xleftarrow{\$} \mathsf{NIZK.Sim}_1(\mathsf{crs}, \tau, (G, S, \mathsf{state}, y))$ and outputs $\mathsf{aux} := \pi$.

Let $\mathcal{A}$ be a PPT adversary on the simulatability property of dpiO. We prove indistinguishability between the real and the simulated distribution via a series of hybrids starting from the simulated game $\mathsf{Exp}^{\text{1-enc}}_{\mathcal{A}}(1^\lambda)$.

Game $\mathbf{G}_0$: This game is identical to $\mathsf{Exp}^{\text{1-enc}}_{\mathcal{A}}(1^\lambda)$. We remark that in this game, the tuple $(y; y')$ is produced using the adversarially chosen sampler S on input of the adversarially chosen state state and input x supplied with true randomness.

Game $\mathbf{G}_1$: This game is identical to $\mathbf{G}_0$ except for the fact that for each query (S, state, x), the sampler S is supplied with randomness $G(r)$ for uniform r (instead of true randomness). Due to the strong regularity of G and by a standard hybrid argument over all queries, the statistical distance between $\mathbf{G}_0$ and $\mathbf{G}_1$ is negligible.

Game $\mathbf{G}_2$: This game is the same as $\mathbf{G}_1$ with the difference that crs is produced honestly using $\mathsf{NIZK.Setup}(1^\lambda)$. Additionally, for each adversarial query (S, state, x), the proof π is produced honestly by $\mathsf{NIZK.Prove}(\mathsf{crs}, (G, S, \mathsf{state}, y), (y', x, r))$, where $G(r)$ are the random coins supplied to the sampler S. The view of $\mathcal{A}$ in game $\mathbf{G}_2$ is distributed exactly as in the real game $\mathsf{Exp}^{\text{0-enc}}_{\mathcal{A}}(1^\lambda)$.

We construct a PPT adversary $\mathcal{B}$ on the zero-knowledge property of NIZK. Given a CRS crs, $\mathcal{B}$ samples an ELF G in injective mode and invokes $\mathcal{A}$ on input of $\mathsf{pp} := (\mathsf{crs}, G)$. Each time $\mathcal{A}$ queries its oracle on

(S, state, x), $\mathcal{B}$ draws random coins r and invokes the sampler S on input of (state, x) with random coins $G(r)$ to obtain $(y; y')$. In order to produce π, $\mathcal{B}$ calls its prove oracle on input $(G, S, \mathsf{state}, y)$ with witness (y', x, r). Therefore, if $\mathcal{B}$ is supplied with an honest CRS and honestly generated proofs, $\mathcal{B}$ perfectly simulates $\mathbf{G}_2$ for $\mathcal{A}$, else $\mathcal{B}$ perfectly simulates $\mathbf{G}_1$. Hence, $|\Pr[\mathsf{out}_2 = 1] - \Pr[\mathsf{out}_3 = 1]| \leq \mathsf{Adv}_{\mathsf{ZK}}(\mathcal{B})$. This concludes the proof.

Restricted Correctness. Let $S \in \mathbf{SI}_\lambda$ be an arbitrary sampler with input, let $y_1, \ldots, y_\ell$ be arbitrary values from the input domain $\mathcal{I}_\lambda$, and let C be a circuit from the family $\mathcal{C}_{\ell\lambda}$. To prove the correctness of dpiO, we proceed over a series of hybrids.

Game $\mathbf{G}_0$: This game is the ideal game $\mathsf{Exp}^{\text{1-rcorr}}_{\mathcal{A}}(1^\lambda)$. As the sampler S is called using true randomness whereas in $\mathsf{Exp}^{\text{0-rcorr}}_{\mathcal{A}}(1^\lambda)$ samples are generated using $G(r)$, where r is truly random, we need an intermediate hybrid.
Game $\mathbf{G}_1$: This game is identical to $\mathbf{G}_0$ with the difference that each call of the sampler S is supplied with $G(r)$ as randomness (where r is sampled uniformly for each call). Due to the strong regularity of G, and by a hybrid argument over all calls of S, the statistical distance between $\mathbf{G}_0$ and $\mathbf{G}_1$ is negligible.
Game $\mathbf{G}_2$: This game is the real game $\mathsf{Exp}^{\text{0-rcorr}}_{\mathcal{A}}(1^\lambda)$.
We now argue that the view of $\mathcal{A}$ in game $\mathbf{G}_1$ is distributed identically to its view in $\mathbf{G}_2$. $\mathbf{G}_2$ samples public parameters pp via $\mathsf{Setup}(1^\lambda)$ and S' an encoded sampler via $S' \leftarrow \mathsf{Encode}(\mathsf{pp}, S)$. Further, (y_j, aux_j) are sampled as $\mathsf{state}_j \leftarrow \mathsf{stf}(y_1, \ldots, y_{j-1}, \bot, \ldots, \bot)$ and $(y_j, \mathsf{aux}_j, y'_j) \xleftarrow{\$} S'(\mathsf{state}_j, x_j)$, for $j \in [\ell]$. Let Λ be the obfuscation $\Lambda \xleftarrow{\$} \mathsf{Obfuscate}(\mathsf{pp}, S, C)$ of the circuit C with respect to sampler S. Due to the perfect correctness of iO, Λ has the same functionality as $\bar{C}[\mathsf{stf}, (\mathsf{crs}, G), S, C, K]$, where K is a freshly generated key for the PRF F. Hence, by the perfect completeness of NIZK, on input of $((y_1, \mathsf{aux}_1), \ldots, (y_\ell, \mathsf{aux}_\ell))$, Λ evaluates the circuit C on input of $(y_1, \ldots, y_\ell)$ with random coins $F(K, (y_1, \ldots, y_\ell))$. Therefore, the view of $\mathcal{A}$ in the games $\mathbf{G}_1$ and $\mathbf{G}_2$ only differs in the fact that $\mathbf{G}_1$ supplies C with true random coins whereas $\mathbf{G}_2$ supplies C with $F(K, (y_1, \ldots, y_\ell))$ as randomness. As F is a perfect PRF, the distribution $\{F(K, (y_1, \ldots, y_\ell)) \mid K \xleftarrow{\$} \mathsf{F.KeyGen}(1^\lambda)\}$ is identical to the uniform distribution over the image of F. Therefore, the view of $\mathcal{A}$ in $\mathbf{G}_1$ and $\mathbf{G}_2$ is distributed identically.

By construction, all $S' \leftarrow \mathsf{Encode}(\mathsf{pp}, S)$ respect the support of S. Furthermore, by construction, perfect completeness of NIZK and perfect correctness of iO, for all $C' \leftarrow \mathsf{Obfuscate}(\mathsf{pp}, S, C)$ and all $(y_1, \mathsf{aux}_1, \ldots, y_\ell, \mathsf{aux}_\ell)$ produced as in $\mathsf{Exp}^{\text{0-rcorr}}$, $C'(y_1, \mathsf{aux}_1, \ldots, y_\ell, \mathsf{aux}_\ell) \in \mathsf{Supp}(C(y_1, \ldots, y_\ell))$.

Security. Let $D \in \mathbf{S}^{\mathsf{d\text{-}Ind}}$ be an arbitrary dynamic-input indistinguishable circuit sampler over $\mathcal{C}$. To prove that dpiO satisfies indistinguishability (Definition 12), we proceed over a series of hybrids. Toward contradiction, assume that there is a PPT adversary $\mathcal{A}$ distinguishing $\mathsf{Exp}^{\text{0-ind}}_{\mathcal{A}}(1^\lambda)$ from $\mathsf{Exp}^{\text{1-ind}}_{\mathcal{A}}(1^\lambda)$ with non-negligible advantage ε over the random guess after making a polynomial number Q of queries to the oracle.

Game $\mathbf{G}_0$. In this game, the challenger samples $b \xleftarrow{\$} \{0, 1\}$, and sets up the experiment $\mathsf{Exp}^{b\text{-ind}}_{\mathcal{A}}(1^\lambda)$. More precisely, $\mathcal{A}$ has access to the public parameters pp and an oracle $\mathcal{O}^{\mathsf{ind}}_b[\mathsf{pp}, D_\lambda]$, that on input of a sampler with input S, draws a sample (C_0, C_1, z) from D and outputs (C_0, C_1, z) together with an obfuscation $\mathsf{Obfuscate}(\mathsf{pp}, S, C_b)$. $\mathcal{A}$ outputs a guess b' and the challenger returns 1 if $b' = b$. By assumption, $\Pr[\mathsf{out}_0 = 1] = \varepsilon$.
Game $\mathbf{G}_1$. In this game, the challenger samples G as $G \xleftarrow{\$} \mathsf{ELF.Gen}(M, t)$, where t is a polynomial such that any PPT algorithm of circuit size s has advantage at most $\varepsilon/2$ in distinguishing $\mathsf{ELF.Gen}(M, M)$ from $\mathsf{ELF.Gen}(M, t)$. The advantage of $\mathcal{A}$ in this game is therefore lower bounded by $\varepsilon/2$: $\Pr[\mathsf{out}_1 = 1] \geq \varepsilon/2$.
Game $\mathbf{G}'_1$. This game proceeds exactly as $\mathbf{G}_1$, except that after sampling $b \xleftarrow{\$} \{0, 1\}$, the challenger always sets up the experiment $\mathsf{Exp}^{\text{1-ind}}_{\mathcal{A}}(1^\lambda)$. The challenger still returns 1 iff $b' = b$.
By using a standard hybrid argument over the oracle queries, we prove that $|\Pr[\mathsf{out}_1 = 1] - \Pr[\mathsf{out}'_1 = 1]| \leq Q \cdot \mathsf{negl}(\lambda)$, where Q is a polynomial in λ.
Game $\mathbf{G}_{1.q}$ This game is identical to $\mathbf{G}_1$ except for the fact that the first q oracle queries are answered using an obfuscation Λ_q of C_1 instead of C_b. Hence, $\Pr[\mathsf{out}_{1.0} = 1] = \Pr[\mathsf{out}_1 = 1]$ and $\Pr[\mathsf{out}_{1.Q} = 1] = \Pr[\mathsf{out}'_1 = 1]$, where Q is the number of adversarial oracle queries.
As $|\Pr[\mathsf{out}_1 = 1] - \Pr[\mathsf{out}'_1 = 1]| \leq \sum_{q=1}^{Q} |\Pr[\mathsf{out}_{1.q} = 1] - \Pr[\mathsf{out}_{1.q+1} = 1]|$, it suffices to upper bound the distinguishing gap between $\mathbf{G}_{1.q}$ and $\mathbf{G}_{1.q+1}$.
We observe that due to the (almost) perfect soundness of NIZK, the obfuscated circuit in the q-th oracle answer simulates the randomized computation of the circuit $C_{q,0}$ only on well-formed inputs, i.e. on outputs

of S_q using random coins from the range of G. As ELF is in extremely lossy mode, this set of well-formed inputs is *extremely sparsified*. Therefore, by the strong regularity of ELF, we can enumerate over all possible outputs at all input positions $j \in [\ell]$. Let $B_{q,j}$ be the set of all well-formed inputs for input position j:

$$B_{q,j} := \{S_q(\mathsf{stf}(y_1, \ldots, y_{j-1}), x; G(r)) \mid x \in \mathcal{I}_\lambda, r \in \{0,1\}^M, y_k \in B_k \text{ for } k \in [j-1]\}.$$

The set $B_{q,j}$ contains at most $|\mathcal{I}| \cdot t^{j-1}$ elements. Further, let $\gamma_{q,1} < \cdots < \gamma_{q,\bar{t}}$ be the ordered enumeration of all ℓ-tuples in $B_q := \prod_{j=1}^{\ell} B_{q,j}$.[12] Hence, the total number of well-formed inputs $\bar{t} = \prod_{j=1}^{\ell} |B_{q,j}| \leq (|\mathcal{I}| \cdot t^{\ell-1})^\ell \leq |\mathcal{I}|^\ell \cdot t^{(\ell^2)}$ is polynomial in λ (given that ℓ is a constant, and $|\mathcal{I}|$ and t are polynomial).

Towards proving indistinguishability between $\mathbf{G}_{1.q}$ and $\mathbf{G}_{1.q+1}$, we conduct a hybrid argument over all well-formed inputs for the obfuscation Λ_q and gradually replace the evaluation of circuit $C_{q,b}$ with $C_{q,1}$. From here on, our proof strategy is similar to the one employed in [Can+15]. However, we only need to consider polynomially many hybrids (as we assume $|\mathcal{I}|$ to be polynomial), hence we only lose a polynomial factor to the underlying assumptions.

Game $\mathbf{G}_{1.q.i}$. In game $\mathbf{G}_{1.q.i}$ the oracle answers the q-th query using an obfuscation of the circuit

$$\bar{C}'[\mathsf{stf}, (\mathsf{crs}, G), S_q, C_{q,b}, C_{q,1}, K_q, \gamma_{q,i}]$$

that is defined in Figure 7 using iO.

Circuit $\bar{C}'[\mathsf{stf}, (\mathsf{crs}, G), S, C_0, C_1, K, \gamma_i](x)$

parse $x =: ((y_1, \mathsf{aux}_1), \ldots, (y_\ell, \mathsf{aux}_\ell))$
$\mathsf{state}_j := \mathsf{stf}(y_1, \ldots, y_{j-1}, \bot, \ldots, \bot)$
if $\neg(\forall j \in [\ell] \colon \mathsf{NIZK.Verify}(\mathsf{crs}, (G, S, \mathsf{state}_j, y_j), \mathsf{aux}_j) = 1)$ **then**
  return $\bot$
$\gamma := (y_1, \ldots, y_\ell)$
if $\gamma < \gamma_i$ **then** $r := F(K, \gamma)$; **return** $C_1(\gamma; r)$
if $\gamma = \gamma_i$ **then** $r := F(K, \gamma)$; **return** $C_b(\gamma; r)$
if $\gamma > \gamma_i$ **then** $r := F(K, \gamma)$; **return** $C_b(\gamma; r)$

Fig. 7. Definition of the circuit $\bar{C}'$.

The circuits $\bar{C}[\mathsf{stf}, (\mathsf{crs}, G), S_q, C_{q,b}, K_q]$ and $\bar{C}'[\mathsf{stf}, (\mathsf{crs}, G), S_q, C_{q,0}, C_{q,1}, K_q, \gamma_{q,1}]$ are functionally equivalent (on input $x = ((y_1, \mathsf{aux}_1), \ldots, (y_\ell, \mathsf{aux}_\ell))$, both return $C_{q,b}(y_1, \ldots, y_\ell)$ with randomness $F(K_q, (y_1, \ldots, y_\ell))$). Hence, this game hop is justified by the indistinguishability property of iO, more formally there exists a PPT adversary $\mathcal{B}$ such that $|\Pr[\mathsf{out}_{1.q} = 1] - \Pr[\mathsf{out}_{1.q.1}] = 1| \leq \mathsf{Adv}_{\mathsf{iO}}(\mathcal{B})$.

We aim to reduce the game hop from $\mathbf{G}^b_{1.q.i}$ to $\mathbf{G}^b_{1.q.i+1}$ to the dynamic-input indistinguishability of the circuit sampler D_λ. For this purpose, we first need to supply $C_{q,b}$ with true randomness. Hence, we define an other series of hybrids between $\mathbf{G}_{1.q.i}$ and $\mathbf{G}_{1.q.i+1}$.

Game $\mathbf{G}_{1.q.i.1}$. This game is identical to $\mathbf{G}_{1.q.i}$ except for the fact that we use a punctured PRF key $K_q\{\gamma_{q,i}\} \xleftarrow{\$} \mathsf{F.Punct}(K_q, \gamma_{q,i})$ and obfuscate the circuit

$$\bar{C}''[\mathsf{stf}, (\mathsf{crs}, G), C_{q,0}, C_{q,1}, K_q\{\gamma_{q,i}\}, Y := C_{q,b}(\gamma_{q,i}; F(K_q, \gamma_{q,i})), \gamma_{q,i}]$$

defined in Figure 8 using iO.

As F preserves the functionality under punctured keys, the circuits $\bar{C}'[\mathsf{stf}, (\mathsf{crs}, G), S_q, C_{q,0}, C_{q,1}, K_q, \gamma_{q,i}]$ and $\bar{C}''[\mathsf{stf}, (\mathsf{crs}, G), S_q, C_{q,0}, C_{q,1}, K_q\{\gamma_{q,i}\}, Y := C_{q,b}(\gamma_{q,i}; F(K_q, \gamma_{q,i})), \gamma_{q,i}]$ are functionally equivalent. Hence, there exists a PPT adversary $\mathcal{B}$ such that $|\Pr[\mathsf{out}_{1.q.i} = 1] - \Pr[\mathsf{out}_{1.q.i.1} = 1]| \leq \mathsf{Adv}_{\mathsf{iO}}(\mathcal{B})$.

We note that the view of $\mathcal{A}$ in game $\mathbf{G}_{1.q.i.1}$ does not depend on the PRF key K. This enables to exploit the selective security of F.

Game $\mathbf{G}_{1.q.i.2}$. In this game we replace the randomness $F(K_q, (\gamma_{q,i}))$ by true randomness, i.e. we produce Y as follows: $Y := C_{q,b}(\gamma_{q,i}; R)$. This game hop is justified by the selective PRF property, more formally $|\Pr[\mathsf{out}_{1.q.i.1} = 1] - \Pr[\mathsf{out}_{1.q.i.2} = 1]| \leq \mathsf{Adv}_{\mathsf{s\text{-}cPRF}}(\mathcal{B})$ for some PPT adversary $\mathcal{B}$.

[12] We remark that the values of each set B_j can be computed efficiently by evaluating S on all possible inputs from $\mathcal{I} \times (\prod_{k=1}^{j-1} B_j)$ and all possible images in the range of G. Furthermore, it is possible to enumerate the image of G in polynomial time because G is strongly regular.

Circuit $\bar{C}''[\mathsf{stf}, (\mathsf{crs}, G), S, C_0, C_1, K\{\gamma_i\}, Y, \gamma_i](x)$

parse $x =: ((y_1, \mathsf{aux}_1), \ldots, (y_\ell, \mathsf{aux}_\ell))$
$\mathsf{state}_j := \mathsf{stf}(y_1, \ldots, y_{j-1}, \bot, \ldots, \bot)$
if $\neg\,(\forall j \in [\ell]\colon \mathsf{NIZK.Verify}(\mathsf{crs}, (G, S, \mathsf{state}_j, y_j), \mathsf{aux}_j) = 1)$ **then**
  return $\bot$
$\gamma := (y_1, \ldots, y_\ell)$
if $\gamma < \gamma_i$ **then** $r := F(K\{\gamma_i\}, \gamma)$; **return** $C_1(\gamma; r)$
if $\gamma = \gamma_i$ **then** **return** Y
if $\gamma > \gamma_i$ **then** $r := F(K\{\gamma_i\}, \gamma)$; **return** $C_b(\gamma; r)$

Fig. 8. Definition of the circuit $\bar{C}''$.

Game $\mathbf{G}_{1.q.i.3}$. Game $\mathbf{G}_{1.q.i.3}$ is the same as $\mathbf{G}_{1.q.i.2}$ except for the fact that Y is produced using the circuit $C_{q,1}$, i.e. $Y := C_{q,1}(\gamma_{q,i}; R)$. This game hop is justified by the fact that the circuit sampler D_λ is a dynamic-input indistinguishable sampler.

Game $\mathbf{G}_{1.q.i.4}$. This game is the same as $\mathbf{G}_{1.q.i.3}$ with the difference that we again use pseudorandom coins to compute Y, i.e. $Y := C_{q,1}(\gamma_{q,i}; F(K_q, \gamma_{q,i}))$. For every PPT adversary $\mathcal{A}$ there exists a PPT adversary $\mathcal{B}$ such that $|\Pr[\mathsf{out}_{1.q.i.3} = 1] - \Pr[\mathsf{out}_{1.q.i.4} = 1]| \leq \mathsf{Adv}_{\mathsf{s\text{-}cPRF}}(\mathcal{B})$.

As the pPRF F preserves functionality under punctured keys, the two circuits $\bar{C}''[\mathsf{stf}, (\mathsf{crs}, G), S_q, C_{q,0}, C_{q,1}, K_q\{\gamma_{q,i}\}, Y := C_{q,1}(\gamma_{q,i}; F(K_q, \gamma_{q,i})), \gamma_{q,i}]$ and $\bar{C}'[\mathsf{stf}, (\mathsf{crs}, G), S_q, C_{q,0}, C_{q,1}, K_q, \gamma_{q,i+1}]$ are functionally equivalent. Therefore, we have that $|\Pr[\mathsf{out}_{1.q.i.4} = 1] - \Pr[\mathsf{out}_{1.q.i+1} = 1]| \leq \mathsf{Adv}_{\mathsf{iO}}(\mathcal{B})$.

Summing up, the advantage to distinguish $\mathbf{G}_1$ and $\mathbf{G}_{1.Q}$ is bounded by $|\mathcal{I}|^\ell \cdot t^{\ell^2} \cdot \mathsf{negl}(\lambda)$. As ℓ is constant and $|\mathcal{I}|, t$ are polynomial, this quantity is negligible. As the circuit obfuscated in $\mathbf{G}_{1.Q}$ is now functionally equivalent to the circuit obfuscated in $\mathbf{G}_1^1$, the game hop to $\mathbf{G}_1'$ is justified by the indistinguishability property of iO. More formally there exists a PPT adversary $\mathcal{B}$ such that $|\Pr[\mathsf{out}_{1.Q} = 1] - \Pr[\mathsf{out}_1'] = 1| \leq \mathsf{Adv}^{\mathsf{iO}}_{\mathcal{B}}(\lambda)$. This implies that the advantage of $\mathcal{A}$ in game $\mathbf{G}_1'$ is lower bounded by $\varepsilon/2 - \mathsf{negl}(\lambda)$, which is non-negligible. However, the view of $\mathcal{A}$ in $\mathbf{G}_1'$ is perfectly independent of b, hence its advantage in this game cannot be non-zero; therefore, we reach a contradiction, which concludes the proof. □

4.3 Extension

We sketch a straightforward extension of our above construction. It follows easily by inspection that the same proof strategy would work even if the ℓ sources, which sample inputs accorded to an encoding of a sampler S with respect to public parameters pp, are not required anymore to use the *same* public parameters. The ℓ sources could even each use different public parameters $(\mathsf{pp}_1, \ldots, \mathsf{pp}_\ell)$. The modified proof for this scenario would proceed by first switching the ELFs in $(\mathsf{pp}_1, \ldots, \mathsf{pp}_\ell)$ to an extremely-lossy mode, through a sequence of ℓ hybrids. Each extremely-lossy mode is chosen so that $\mathcal{A}$ as advantage at most $\varepsilon/2\ell$ in distinguishing it from the injective mode. By a union bound, $\mathcal{A}$ has therefore advantage at most $\varepsilon/2$ in distinguishing the all-injective modes from the all-lossy modes. Then, enumerating over all possible valid inputs to an obfuscated circuit takes polynomial time as before, as each input of a source comes from a set of polynomial size. Therefore, the exact same sequence of hybrids proves security, with a polynomial loss in the underlying primitives. To adapt the security properties of our definition of dpiO to this multi-parameter setting, it suffices to let all experiments initially sample and send to the adversary ℓ public parameters $(\mathsf{pp}_1, \ldots, \mathsf{pp}_\ell)$ instead of one. In the simulatability of encodings definition (resp. in the indistinguishability definition), the adversary is allowed to specify under which public parameters it wants to receive a (real or simulated) sample $(y, \mathsf{aux}; y')$ (resp. under which public parameters it wants C_b to be obfuscated in the indistinguishability experiment).

It can prove convenient to simplify the construction in some applications to allow different sources to use different public parameters. Let us illustrate the syntax we adopt on an example: if $(\mathsf{Setup}, \mathsf{Encode}, \mathsf{Obfuscate})$ is a 5-source dpiO scheme, we denote by $\mathsf{Obfuscate}(\mathsf{pp}_1[1-3], \mathsf{pp}_2[4,5], , S, C)$ an obfuscation of a circuit C, whose first three inputs should be sampled with respect to pp_1, and whose last two inputs should be sampled with respect to pp_2. We will also sometimes slightly abuse our notation, noting that an ℓ-source dpiO scheme directly implies an i-source dpiO scheme for $i \leq \ell$, and allow an ℓ-source scheme to obfuscate a circuit C that takes $i < \ell$ inputs.

5 Leveled Homomorphic Encryption

In this section we show that our notion of dpIO from Section 3 can be applied to construct leveled homomorphic encryption in a similar way as in [Can+15]. This construction leads to a transformation which operates on an encryption scheme E, satisfying IND-CPA security (and possibly other security properties, e.g., KDM security), and produces a leveled homomorphic encryption scheme that retains the security properties of E. We recall the definition of IND-CPA secure encryption schemes in .

Definition 15 (Leveled Homomorphic Encryption [Gen09]). *A leveled homomorphic encryption scheme* LHE *is a family of encryption schemes* $\{\mathsf{Enc}_L : L \geq 0\}$ *such that each* Enc_L *is homomorphic for all polysize depth-*L *circuits (*i.e.*, it allows to homomorphically and compactly evaluate any polysize depth-*L *circuit) with an algorithm* Eval *of size polynomial in* (λ, L)*, and all the* Enc_L *use the same decryption circuit. We require* LHE *to meet the following properties:*

Correctness. *We say that* LHE *is* correct *if for every* $\lambda \in \mathbb{N}$*, for every polysize depth-*L *circuits* C *and respective inputs* $m_1, \ldots, m_k \in \{0,1\}$*, we have that*

$$\Pr\left[\mathsf{LHE.Dec}(\mathsf{sk}, (\mathsf{LHE.Eval}(\mathsf{ek}, C, c_1, \ldots, c_k))) = C(m_1, \ldots, m_k)\right]$$

is negligible in λ*, where* $(\mathsf{pk}, \mathsf{ek}, \mathsf{sk}) \xleftarrow{\$} \mathsf{LHE.KeyGen}(1^\lambda)$ *and* $m_j \xleftarrow{\$} \mathsf{LHE.Enc}(\mathsf{pk}, m_j)$ *for* $j \in [k]$.

Security. *We say that* LHE *is* IND-CPA secure *if for every PPT adversary* $\mathcal{A}$ *the advantage*

$$\begin{aligned}\mathsf{Adv}_{\mathsf{IND\text{-}CPA}}(\mathcal{A}) := |&\Pr\left[\mathcal{A}(\mathsf{pk}, \mathsf{ek}, \mathsf{LHE.Enc}(\mathsf{pk}, 0))\right] \\ &- \Pr\left[\mathcal{A}(\mathsf{pk}, \mathsf{ek}, \mathsf{LHE.Enc}(\mathsf{pk}, 1))\right]|\end{aligned}$$

is negligible in λ*, where* $(\mathsf{pk}, \mathsf{ek}, \mathsf{sk}) \xleftarrow{\$} \mathsf{LHE.KeyGen}(1^\lambda)$.

Let stf_λ be the trivial state function, i.e. $\mathsf{stf}\colon (y_1, y_2) \mapsto \bot$ for each $(y_1, y_2) \in (\{0,1\}^\lambda \cup \{\bot\})^2$. Let $E = (E.\mathsf{KeyGen}, E.\mathsf{Enc}, E.\mathsf{Dec})$ be an $\mathsf{IND\text{-}CPA}$-secure public-key encryption scheme. Let the class $\mathcal{SI}$ contain all samplers S^{pk} that on input of a state state and an input $x \in \mathcal{I} := \{0,1\}$, produce an encryption $y := E.\mathsf{Enc}(\mathsf{pk}, x)$ and $y' := \bot$ ignoring state, where pk is a public key in the range of $E.\mathsf{KeyGen}(1^\lambda)$. Let $\mathcal{C}$ be the class of polynomially sized randomized circuits and let $\mathbf{S}^{\mathsf{d\text{-}Ind}}$ be the class of dynamic-input indistinguishable samplers over $\mathcal{C}$.

Theorem 16. *Let* $(\mathsf{Setup}, \mathsf{Encode}, \mathsf{Obfuscate})$ *be a* 2*-source* dpiO *scheme for* $(\mathsf{stf}, \mathcal{SI}, \mathcal{C}, \mathbf{S}^{\mathsf{d\text{-}Ind}})$ *and let* E *be an IND-CPA secure public-key encryption scheme. Then,* LHE *as defined in Figure 9 is an IND-CPA secure LHE scheme.*

$\mathsf{LHE.KeyGen}(1^\lambda, 1^L)$

for $i \in \{0, \ldots, L\}$ **do**
  $(\mathsf{pk}_i, \mathsf{sk}_i) \xleftarrow{\$} E.\mathsf{KeyGen}(1^\lambda)$
  $\mathsf{pp}_i \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$
  $S'^{\mathsf{pk}_i} \leftarrow \mathsf{Encode}(\mathsf{pp}_i, S^{\mathsf{pk}_i})$
for $i \in \{1, \ldots, L\}$ **do**
  $\Lambda_i \xleftarrow{\$} \mathsf{Obfuscate}\left(\mathsf{pp}_{i-1}, S^{\mathsf{pk}_{i-1}}, C[S'^{\mathsf{pk}_i}, \mathsf{sk}_{i-1}]\right)$
$\mathsf{pk} := S'^{\mathsf{pk}_0}$, $\mathsf{sk} := \mathsf{sk}_L$, $\mathsf{ek} := (\Lambda_1, \ldots, \Lambda_L)$
return $(\mathsf{pk}, \mathsf{ek}, \mathsf{sk})$

$\mathsf{LHE.Enc}(\mathsf{pk}, m \in \{0,1\})$

parse $\mathsf{pk} =: S'^{\mathsf{pk}_0}$
$(y, \mathsf{aux}, y') \xleftarrow{\$} S'^{\mathsf{pk}_0}(\bot, m)$
return $c \leftarrow (y, \mathsf{aux})$

$\mathsf{LHE.Dec}(\mathsf{sk}, c)$

parse $\mathsf{sk} =: \mathsf{sk}_L$
parse $c =: (y, aux)$
return $E.\mathsf{Dec}(\mathsf{sk}_L, y)$

$\mathsf{LHE.Eval}(\mathsf{ek}, C, (c_1, \ldots, c_l))$

for $i \in \{1, \ldots, L\}$ **do**
  foreach gate g on level i **do**
    // let α_g, β_g denote the respective inputs
    $\gamma_g := \Lambda_i(\alpha_g, \beta_g)$

Fig. 9. Description of the LHE scheme LHE. The circuit C is defined in Figure 10.

The proof strategy is similar as in [Can+15]. On a high level, we want to reduce the security of LHE to the security of the underlying encryption scheme E. However, the evaluation key ek contains information (even though obfuscated) on the secret keys of each level. For the purpose of invoking the security of E on the challenge ciphertext, we need to remove this dependency on sk_0. Therefore, we gradually (starting from level L) replace the obfuscations of the circuits C with an obfuscation of *trapdoor* circuits tC that simply output samples produced by the encoded sampler S' on input of 0 (hence, not needing any information on decryption keys). These two circuits only differ in the fact that they sample from the same encoded sampler S' using (possibly) different inputs. Due to the simulatability of encodings and the IND-CPA security of E, the two circuits are dynamic-input indistinguishable. Hence, by the indistinguishability property of dpiO for $\mathbf{S}^{\mathsf{d\text{-}Ind}}$, an honest evaluation key and an evaluation key consisting only of trapdoor circuits are indistinguishable.

Given these modifications, the challenge ciphertext c^* consists of an encryption of a bit b under pk_0 accompanied by some auxiliary information produced by the corresponding encoded sampler. This auxiliary information might leak information on the bit b and thereby prevents to directly employ the IND-CPA security of E. However, as dpiO satisfies simulatability of encodings, this auxiliary information can be simulated without knowledge of b and, hence, contains no information about b. Therefore, by the IND-CPA security of E, LHE is IND-CPA secure. Given our construction of dpiO from Section 4, we obtain the following corollary:

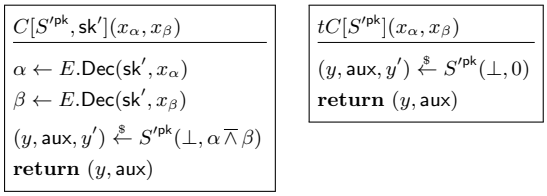

Fig. 10. Definition of the circuits C and tC.

Before we start the analysis of our construction, we briefly recall the definition of an IND-CPA secure public-key encryption scheme.

Definition 17 (Public-key bit-encryption scheme). *A* public-key bit-encryption scheme *is a triple of PPT algorithms* $E = (E.\mathsf{KeyGen}, E.\mathsf{Enc}, E.\mathsf{Dec})$ *that satisfies the following properties:*

Perfect correctness. *E is perfectly correct, if for every $\lambda \in \mathbb{N}$ and every message $m \in \{0,1\}$*

$$\Pr[E.\mathsf{Dec}(\mathsf{sk}, E.\mathsf{Enc}(\mathsf{pk}, m)) = m | (\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} E.\mathsf{KeyGen}(1^\lambda)] = 1.$$

IND-CPA security. *We say that E is* IND-CPA secure *if for every PPT adversary $\mathcal{A}$, the advantage*

$$\begin{aligned}\mathsf{Adv}_{\mathsf{IND\text{-}CPA}}(\mathcal{A}) := |\Pr[\mathcal{A}(\mathsf{pk}, \mathsf{LHE.Enc}(\mathsf{pk}, 0))] \\ - \Pr[\mathcal{A}(\mathsf{pk}, \mathsf{LHE.Enc}(\mathsf{pk}, 1))]|\end{aligned}$$

is negligible in λ, where $(\mathsf{pk}, \mathsf{sk}) \leftarrow E.\mathsf{KeyGen}(1^\lambda)$.

Proof (of Theorem 16). We prove that LHE satisfies correctness and IND-CPA security.

Correctness. Let C be a polysized circuit of depth L that consists only of $\barwedge$-gates. We prove that on every level i every gate g in C is evaluated as the original gate given the decrypted inputs (except for an negligible error). By a union bound argument over the number of gates, the probability that an error occurs when evaluating circuit C is negligible.
Let $i \in [L]$ be a level and let g be a gate on level i. Further, let $(x_\alpha, \mathsf{aux}_\alpha, \cdot)$ be in the scope of $S'^{\mathsf{pk}_{i-1}}(\bot, \alpha)$ and $(x_\beta, \mathsf{aux}_\beta, \cdot)$ be in the scope of $S'^{\mathsf{pk}_{i-1}}(\bot, \beta)$. By the perfect correctness of E, the circuit $C[S'^{\mathsf{pk}_i}, \mathsf{sk}_{i-1}]$ on input of $(x_\alpha, \mathsf{aux}_\alpha)$ and $(x_\beta, \mathsf{aux}_\beta)$ always outputs an element from the scope of $S'^{\mathsf{pk}_i}(\bot, \alpha \barwedge \beta)$.[13] Furthermore, due to the statistical correctness of dpiO, the distribution $C[S'^{\mathsf{pk}_i}, \mathsf{sk}_{i-1}]((x_\alpha, \mathsf{aux}_\alpha), (x_\beta, \mathsf{aux}_\beta))$ is statistically close to the output of Λ_i (over the random coins of Setup, $\mathsf{Obfuscate}$, and S'). Hence, the probability that gate g is not evaluated correctly is negligible. Therefore, the probability (over the random coins of $\mathsf{LHE.KeyGen}$ and $\mathsf{LHE.Enc}$) that the circuit C is not evaluated correctly is negligible.[14]

[13] We implicitly use that the scopes of $S'(\bot, 0)$ and $S'(\bot, 1)$ are disjoint.
[14] Our construction from Section 4 even guarantees perfect correctness.

Security. To prove that LHE is IND-CPA secure, we proceed over a series of hybrids. Let $\mathcal{A}$ be a PPT adversary.

Game $\mathbf{G}_0^b$. This game is exactly the IND-CPA game for LHE, where the challenge ciphertext c^* contains the bit b.

Game $\mathbf{G}_1^b$. This game is the same as $\mathbf{G}_0^b$ with the difference that the evaluation key consists of obfuscations $t\Lambda_i$ of the circuit $tC[S'^{\mathsf{pk}_i}]$.

Claim. For every PPT adversary $\mathcal{A}$, we have that $|\Pr[\mathsf{out}_0^b = 1] - \Pr[\mathsf{out}_1^b = 1]| \leq \mathsf{negl}(\lambda)$ for some negligible function negl.

Proof. We define a series of $L+1$ hybrid games $\mathbf{G}_{0.i}^b$ for $i \in \{0, \ldots, L\}$ as follows:

Game $\mathbf{G}_{0.i}^b$. This game is identical to $\mathbf{G}_{0.i-1}^b$ with the difference that we replace the obfuscated circuit Λ_{L-i+1} with the obfuscation $t\Lambda_{L-i+1} := \mathsf{Obfuscate}(\mathsf{pp}_{L-i}, S^{\mathsf{pk}_{L-i}}, tC[\mathsf{pk}_{L-i+1}])$. Hence, in $\mathbf{G}_{0.i}^b$ the evaluation key ek has the form

$$\mathsf{ek} := (\Lambda_1, \ldots, \Lambda_{L-i}, t\Lambda_{L-i+1}, \ldots, t\Lambda_L)\,.$$

We define $\mathbf{G}_{0.0}^b := \mathbf{G}_0^b$. Hence, we have $\mathbf{G}_{0.L}^b = \mathbf{G}_1^b$.

Claim. For every PPT adversary $\mathcal{A}$, there exists a PPT adversary $\mathcal{B}$ on the indistinguishability property of dpiO, such that $|\Pr[\mathsf{out}_{0.i}^b = 1] - \Pr[\mathsf{out}_{0.i+1}^b = 1]| \leq \mathsf{Adv}_{\mathsf{ind}}(\mathcal{B})$.

By a standard hybrid argument, this implies that $|\Pr[\mathsf{out}_0^b = 1] - \Pr[\mathsf{out}_1^b = 1]| \leq L \cdot \mathsf{Adv}_{\mathsf{ind}}(\mathcal{B})$.

Proof. Let D be a circuit sampler, that samples public parameters $\mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$ and two public key pairs $(\mathsf{pk}, \mathsf{sk}), (\mathsf{pk}', \mathsf{sk}') \xleftarrow{\$} E.\mathsf{KeyGen}(1^\lambda)$, produces the encoded sampler $S'^{\mathsf{pk}} \leftarrow \mathsf{Encode}(\mathsf{pp}, S^{\mathsf{pk}})$ and outputs the circuits $C_0 := C[S'^{\mathsf{pk}}, \mathsf{sk}']$ and $C_1 := tC[S'^{\mathsf{pk}}]$ together with its state $z := (\mathsf{pk}, \mathsf{pk}', \mathsf{sk}', \mathsf{pp})$.

Claim. The circuit sampler D as defined above is a dynamic-input indistinguishable sampler.

Proof. We prove this using two hybrids. Let $\mathcal{A}$ be an adversary for the dynamic-input indistinguishability game for the circuit sampler D.

Game $\mathbf{H}_0$. This is the dynamic-input indistinguishability game as defined in Figure 1. In detail, given the descriptions of the sampled circuits C_0 and C_1 together with the state z, $\mathcal{A}$ chooses an input x (dynamically). Game $\mathbf{H}_0$ evaluates the circuit C_b (for a random bit b) at x using fresh random coins and sends the output to $\mathcal{A}$.

Game $\mathbf{H}_1$. By the simulatability of encodings property of dpiO, there exists a simulator $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$. Game $\mathbf{H}_1$ is the same as $\mathbf{H}_0$ with the difference that the public parameters are sampled as $(\mathsf{pp}, \mathsf{trap}) \xleftarrow{\$} \mathsf{Sim}_0(1^\lambda)$. Given the input $x = (y_1, y_2)$ for the circuits, compute $(m_0, m_1) := (\alpha \barwedge \beta, 0)$ using the secret key sk'. Furthermore, instead of evaluating C_b at x honestly, game $\mathbf{H}_1$ produces $(y, \cdot) \xleftarrow{\$} S^{\mathsf{pk}}(\bot, m_b)$ and $\mathsf{aux} \xleftarrow{\$} \mathsf{Sim}_1(\mathsf{trap}, S^{\mathsf{pk}}, y, \bot)$. Due to the simulatability of encodings, the difference $|\Pr[\mathsf{out}_{H_0} = 1] - \Pr[\mathsf{out}_{H_1} = 1]|$ between these two games is negligible.

The advantage of $\mathcal{A}$ in game $\mathbf{H}_1$ is negligible. To realize that, we construct an adversary $\mathcal{B}$ on the IND-CPA security of E with respect to the public key pk. On input of pk, $\mathcal{B}$ samples (C_0, C_1, z) in the same way as in $\mathbf{H}_1$ (embedding pk) and calls $\mathcal{A}$ on input of (C_0, C_1, z) to obtain (x, state). $\mathcal{B}$ decrypts $x = (y_1, y_2)$ to obtain α, β using sk', and outputs $(m_0, m_1) := (\alpha \barwedge \beta, 0)$ to the IND-CPA game. In return $\mathcal{B}$ receives a ciphertext c^*, simulates the corresponding auxiliary information $\mathsf{aux} \xleftarrow{\$} \mathsf{Sim}_1(\mathsf{pp}, \mathsf{trap}, S^{\mathsf{pk}}, c^*, \bot)$, and invokes $\mathcal{A}$ on input of its previous state state and (c^*, aux). If c^* encrypts $m_{\tilde{b}}$, y is distributed exactly as in $\mathbf{H}_1$ conditioned on $b = \tilde{b}$. Finally, $\mathcal{B}$ outputs $\mathcal{A}$'s output. Hence, by the IND-CPA security of E, $|\Pr[\mathsf{out}_{H.1} = 1] - \frac{1}{2}|$ is negligible. □

In order to upper bound the distinguishing gap between $\mathbf{G}_{0.i}$ and $\mathbf{G}_{0.i+1}$, we construct a PPT adversary $\mathcal{B}$ on the indistinguishability property of dpiO. $\mathcal{B}$ gets as input public parameters pp', the circuits C_0, C_1 together with auxiliary information z produced by D, and an obfuscation $\bar{\Lambda} \xleftarrow{\$} \mathsf{Obfuscate}(\mathsf{pp}', S^{\mathsf{pk}'}, C_{\tilde{b}})$ for some $\tilde{b} \in \{0, 1\}$. Initially, $\mathcal{B}$ samples parameters as $\mathsf{LHE.KeyGen}$ embedding its input as $\mathsf{pp}_{L-i-1} := \mathsf{pp}'$, $\mathsf{pk}_{L-i-1} := \mathsf{pk}'$, and $\mathsf{pk}_{L-i} := \mathsf{pk}$, and defines the public key to be $\mathsf{pk} := S'^{\mathsf{pk}_0}$ for $S'^{\mathsf{pk}_0} \leftarrow \mathsf{Encode}(\mathsf{pp}_0, S^{\mathsf{pk}})$. $\mathcal{B}$ produces the obfuscations

$$\Lambda_j \xleftarrow{\$} \mathsf{Obfuscate}(\mathsf{pp}_{j-1}, S^{\mathsf{pk}_{j-1}}, C[S^{\mathsf{pk}_j}, \mathsf{sk}_{j-1}]) \text{ for } j \in \{1, \ldots, L-i-1\},$$
$$t\Lambda_j \xleftarrow{\$} \mathsf{Obfuscate}(\mathsf{pp}_{j-1}, S^{\mathsf{pk}_{j-1}}, tC[S^{\mathsf{pk}_j}]) \text{ for } j \in \{L-i+1, \ldots, L\}$$

and defines the evaluation key $\mathsf{ek} := (\Lambda_1, \ldots, \Lambda_{L-i-1}, \bar{\Lambda}, t\Lambda_{L-i+1}, \ldots, t\Lambda_L)$.

The challenge ciphertext c^* is produced for the bit b as $c^* \xleftarrow{\$} S'^{\mathsf{pk}_0}(b)$. Finally, $\mathcal{B}$ calls $\mathcal{A}$ on input $(\mathsf{pk}, \mathsf{ek}, c^*)$ and outputs the resulting output b'. We observe that if $\tilde{b} = 0$, $\mathcal{B}$ simulates $\mathbf{G}_{0.i}^b$ for $\mathcal{A}$, else $\mathcal{B}$ simulates $\mathbf{G}_{0.i+1}^b$ for $\mathcal{A}$. Hence, $|\Pr[\mathsf{out}_{0.i}^b = 1] - \Pr[\mathsf{out}_{0.i+1}^b = 1]| \leq \mathsf{Adv}_{\mathsf{ind}}(\mathcal{B})$ which is negligible by assumption. □

□

With the objective of exploiting the IND-CPA security of E to the challenge ciphertext c^*, we need to ensure that the auxiliary information that accompanies the ciphertext from $S^{\mathsf{pk}_0}(b)$ does not leak any information about b. This is formalized via another game transition.

Game $\mathbf{G}_2^b$. This game is identical to $\mathbf{G}_1^b$ except for the fact that the public parameters pp_0 are simulated by Sim_0 (additionally yielding a trapdoor trap_0). Furthermore, the challenge ciphertext c^* is produced by drawing a sample $(y; y')$ from $S^{\mathsf{pk}_0}(b)$ and simulating the auxiliary information aux using $\mathsf{Sim}_1(\mathsf{pp}_0, \mathsf{trap}_0, S^{\mathsf{pk}_0}, y, \bot)$. We recall that for any PPT adversary $\mathcal{A}$ such a simulator $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ exists as dpiO satisfies simulatability of encodings. Hence, by simulatability of encodings, the distinguishing gap $|\Pr[\mathsf{out}_1^b = 1] - \Pr[\mathsf{out}_2^b = 1]|$ is negligible in λ.

It remains to argue, that the games $\mathbf{G}_2^0$ and $\mathbf{G}_2^1$ are computationally indistinguishable. This can be reduced to the IND-CPA security of E with respect to pk_0 as in $\mathbf{G}_2^b$ neither the corresponding secret key sk_0 nor the plaintext b are necessary for simulation.

Claim. For every PPT adversary $\mathcal{A}$, the distinguishing gap $|\Pr[\mathsf{out}_1^0 = 1] - \Pr[\mathsf{out}_1^1 = 1]|$ is negligible.

Proof. Let $\mathcal{A}$ be a PPT adversary on distinguishing $\mathbf{G}_2^0$ and $\mathbf{G}_2^1$. We construct a PPT adversary $\mathcal{B}$ on the IND-CPA security of E. On input of pk, $\mathcal{B}$ defines $\mathsf{pk}_0 := \mathsf{pk}$, samples key pairs $(\mathsf{pk}_j, \mathsf{sk}_j) \xleftarrow{\$} E.\mathsf{KeyGen}(1^\lambda)$ for $j \in \{1, \ldots, L\}$ and produces ek as in $\mathbf{G}_2^1$. $\mathcal{B}$ outputs the messages 0 and 1 to the IND-CPA game and in turn receives a ciphertext c'^*. Furthermore, $\mathcal{B}$ simulates the auxiliary information by $\mathsf{aux}* \xleftarrow{\$} \mathsf{Sim}_1(\mathsf{pp}_0, \mathsf{trap}_0, S^{\mathsf{pk}_0}, c'^*, \bot)$. Finally, $\mathcal{B}$ invokes $\mathcal{A}$ on input of $(\mathsf{pk}_0, \mathsf{ek}, (c'^*, \mathsf{aux}*)$ and forwards its output to the IND-CPA experiment. Hence, $|\Pr[\mathsf{out}_1^0 = 1] - \Pr[\mathsf{out}_1^1 = 1]|$ is negligible by the IND-CPA security of E. □

□

Given our construction of dpiO from Section 4, we obtain the following corollary:

Corollary 18. *Assuming polynomially secure indistinguishability obfuscation and extremely lossy functions, there exists a leveled homomorphic encryption scheme.*

Note that IND-CPA secure cryptosystems, as required in our construction, can be constructed from (polynomially secure) IO and one-way function (the latter being implied by ELFs). Previously, constructions of LHE were only known from the learning with error assumption, or from *subexponentially secure* indistinguishability obfuscation (together with lossy encryption, which can be based e.g. on DDH). Using the generic transformation from leveled homomorphic encryption to fully homomorphic encryption from [Can+15], we also get:

Corollary 19. *Assuming slightly-superpolynomially secure indistinguishability obfuscation and extremely lossy functions, there exists a fully homomorphic encryption scheme.*

Previously, constructions of FHE were only known from circular-security assumptions over lattice-based cryptosystems, or subexponentially secure indistinguishability obfuscation and lossy encryption. Below, we sketch an improvement to Corollary 19, which removes the need for superpolynomially-secure iO.

Fully Homomorphic Encryption from Polynomial iO and ELFs. Applying the [Can+15] transform of LHE into FHE requires to use superpolynomially secure iO. However, using the same techniques as we used to build dpiO, we can actually base this transformation on polynomially secure iO (and ELFs). We briefly sketch the strategy. The LHE-to-FHE transform of [Can+15] stems from the following observation: if an LHE scheme can handle a (slightly) superpolynomial number of levels L, then it is fully homomorphic for all polynomial size circuits. However, this would require generating and storing a superpolynomial number of evaluation keys $(\mathsf{ek}_1, \ldots, \mathsf{ek}_L)$.

To overcome this issue, the strategy of [Can+15] is to obfuscate a master key generation program MProg, which generates the evaluation keys *on the fly*: on input i, MProg returns ek_i. It was observed in [Can+15] that probabilistic iO cannot be directly used to obfuscate this program, because the coins used by MProg must be correlated between two consecutive levels (for example, in our LHE construction, ek_i is an obfuscated program with a secret key sk_i and a public key pk_{i+1} hardcoded; hence, ek_{i+1} must contain the matching

secret key for pk_{i+1}). Nevertheless, "derandomizing" this program, using the standard technique of generating the coins using a pseudorandom function F (the coins for sk_i are generated as $\mathsf{F}(K, i)$, and the coins for pk_{i+1} are generated as $\mathsf{F}(K, i+1)$, which guarantees the appropriate correlation between the coins used in ek_i and ek_{i+1}), and obfuscating with standard iO, allows to prove security of this approach.

To apply the techniques developped in this work, we must make the input to MProg "sparsifiable", so as to puncture at every possible input with a polynomial loss in the security reduction. We outline a natural approach to achieve this. Instead of directly taking the index i as input, the program MProg takes as input a pair (c_i, π), where c_i is a counter defined as follows: $c_0 = 0$, and $c_{i+1} = G(c_i)$, where G is an extremely lossy function in injective mode, and π is a proof (with a statistically sound NIZK proof system) that c_i is in the image of G. On input (c_i, π), the program MProg checks the proof, and outputs $\perp$ if the check fails. If the check passes, it computes $c_{i+1} \leftarrow G(c_i)$, "draws" random coins $r_i \leftarrow \mathsf{F}(K, c_i)$ and $r_{i+1} \xleftarrow{\$} \mathsf{F}(K, c_{i+1})$, and uses these random coins to generate the evaluation key ek_i. The exact same proof strategy as [Can+15] applies, except that in the security analysis, we first switch G to an extremely lossy mode, and enumerate over all pairs of the form $(x, G(x))$ where x is in the image of G; this enumeration is done in polynomial time, hence the reduction only looses a polynomial factor. Therefore, we get:

Corollary 20. *Assuming polynomially-secure indistinguishability obfuscation and extremely lossy functions, there exists a fully homomorphic encryption scheme.*

By replacing the IND-CPA secure encryption scheme E in the construction of Figure 9 by a KDM-secure encryption scheme for the identity function, which remains secure even when the adversary is allowed to obtain ciphertexts of the form $E(\mathsf{pk}, \mathsf{sk})$ (*i.e.*, encryptions of the secret key), we obtain a fully-homomorphic encryption scheme which satisfies KDM-security for the identity function. As already noted in the introduction, this directly implies a fully-KDM secure encryption scheme (which remains secure even when the adversary can receive encryptions of arbitrary functions of its secret key). Plugging in the DDH-based KDM-secure encryption scheme of [Bon+08], we obtain:

Corollary 21. *Assuming polynomially-secure indistinguishability obfuscation and* eDDH*, there exists a fully KDM-secure encryption scheme.*

6 Spooky Encryption

In this section, we recall the definition of spooky encryption from [Dod+16], describe a new construction of spooky encryption from dpiO, and discuss some applications.

6.1 Tools and Definitions

Definition 22 (Spooky Encryption [Dod+16]). *Let $\mathcal{C}$ be a class of (possibly randomized) circuits with n inputs and n outputs. An n-key $\mathcal{C}$-spooky encryption scheme* SE *is a five-tuple of PPT algorithms* $(\mathsf{SE.Setup}, \mathsf{SE.KeyGen}, \mathsf{SE.Enc}, \mathsf{SE.Eval}, \mathsf{SE.Dec})$ *such that*

- $(\mathsf{SE.Setup}, \mathsf{SE.KeyGen}, \mathsf{SE.Enc}, \mathsf{SE.Dec})$ *is an* $\mathsf{IND\text{-}CPA}$*-secure bit encryption scheme;*
- $\mathsf{SE.Eval}(C, (\mathsf{pk}_1, c_1), \ldots, (\mathsf{pk}_n, c_n))$*, on input a circuit $C \in \mathcal{C}$, and n pairs of public key and ciphertext, outputs n ciphertexts;*

which additionally satisfies the following condition: for every λ, every $C \in \mathcal{C}$, every input $x = (x_1, \ldots, x_n)$ to C, the distribution

$$\{\forall i, (\mathsf{pk}_i, \mathsf{sk}_i) \xleftarrow{\$} \mathsf{SE.KeyGen}(1^\lambda), c_i \xleftarrow{\$} \mathsf{SE.Enc}(\mathsf{pk}_i, x_i),$$
$$(c'_1, \ldots, c'_n) \leftarrow \mathsf{SE.Eval}(C, (\mathsf{pk}_i, c_i)_{i \le n}) : (\mathsf{SE.Dec}(\mathsf{sk}_i, c'_i))_{i \le n}\}$$

is statistically close (in λ) to the distribution $C(x_1, \ldots, x_n)$.

Note that we allow the encryption scheme to have a universal setup algorithm, that generates common parameters used (implicitly) in all other algorithms (in particular in the key generation). This corresponds to

a "common reference string" flavor of spooky encryption, as for the LWE-based construction in [Dod+16]. The piO-based construction given in [Dod+16], however, does not require any such setup, but the definition of dpiO requires a universal setup, which we therefore assume for our construction of spooky encryption below.

An important class of spooky relations are the circuits that, on input $(x_1, \ldots, x_n)$, output random additive (bitwise) shares of a function $f(x_1, \ldots, x_n)$. A spooky encryption scheme handling this type of relation is called AFS-spooky.

Two-Key Spooky Encryption of Re-Sampleable Circuits. The work of [Dod+16] constructs two-key spooky encryption schemes, for probabilistic circuits satisfying *efficient re-sampleability*, assuming piO (as well as other primitives, which can be instanciated from DDH). Below, we revisit their construction, and show that we can replace the underlying piO scheme by a dpiO scheme for samplers with input over a small domain. We first recall the necessary notions and primitives.

Definition 23 (Efficient Re-Sampleability). *A probabilistic circuit* $C\colon \{0,1\}^{\ell_1} \times \{0,1\}^{\ell_2} \to \{0,1\}^{\ell_1'} \times \{0,1\}^{\ell_2'}$ *of polynomial size is* efficiently re-sampleable *if there exists a polynomial-size resampling circuit* RS_C *such that for any* $(x_1, x_2) \in \{0,1\}^{\ell_1} \times \{0,1\}^{\ell_2}$*, the distribution* $C(x_1, x_2)$ *is identical to the distribution* $\{(y_1, y_2) \xleftarrow{\$} C(x_1, x_2), y_2' \xleftarrow{\$} \mathsf{RS}_C(x_1, x_2, y_1) : (y_1, y_2')\}$.

Let $M : \{0,1\}^2 \mapsto \{0,1\}^2$ be a probabilistic circuit which performs a *spooky multiplication*: on input (b_1, b_2), it outputs random bit-shares of their product $b_1 b_2$. It is clear that M is efficiently re-sampleable. We write $M(b_1, b_2; r)_1$ to denote the first output of M on input (b_1, b_2) with randomness r.

Maliciously Circuit-Private (Non-Compact) Homomorphic Encryption Scheme. A public key encryption scheme $E = (E.\mathsf{KeyGen}, E.\mathsf{Enc}, E.\mathsf{Dec})$ is *homomorphic* for a class of circuits $\mathcal{C}$ if there exists an algorithm $E.\mathsf{Eval}$ such that for every key pair $(\mathsf{pk}, \mathsf{sk})$, circuit $C \in \mathcal{C}$, and ciphertext $c \leftarrow E.\mathsf{Enc}(\mathsf{pk}, x)$, it holds that $E.\mathsf{Dec}(\mathsf{sk}, E.\mathsf{Eval}(\mathsf{pk}, C, c)) = C(x)$.

Definition 24 (Perfect Malicious Circuit Privacy [Dod+16]). *A homomorphic encryption scheme* $E = (E.\mathsf{KeyGen}, E.\mathsf{Enc}, E.\mathsf{Dec}, E.\mathsf{Eval})$ *for a class of circuits* $\mathcal{C}$ *has* perfect malicious circuit privacy *if for every alleged public key* pk *(possibly outside of the support of* KeyGen*) and ciphertext* c^* *(possibly maliciously computed), there exists an "effective plaintext"* x *such that for every two circuits* $(C_1, C_2) \in \mathcal{C}$ *with* $C_1(x)$ *and* $C_2(x)$*, it holds that* $E.\mathsf{Eval}(\mathsf{pk}, C_1, c^*)$ *and* $E.\mathsf{Eval}(\mathsf{pk}, C_2, c^*)$ *are identically distributed.*

Our construction will employ an homomorphic encryption scheme with perfect malicious circuit privacy. As noted in [Dod+16], a perfect malicious circuit private encryption scheme homomorphic for NC_1 (which suffices for the construction) can be constructed from the DDH assumption.

6.2 Overview of the dpIO-Based Construction

At an intuitive level, our construction of spooky encryption from dpiO mimics the piO-based construction of [Dod+16], in the same way that our construction of LHE from dpiO in the previous section follows the same path as the piO-based construction of LHE from [Can+15]. However, this strategy faces a number of technical challenges. Below, we provide a high level overview of our construction, the challenges which must be overcome, and our solutions.

Overview. The construction of [Dod+16] proceeds in two steps: first, it constructs a two-key spooky encryption scheme for bit inputs that supports evaluation of spooky multiplication. This evaluation procedure is performed by an obfuscated circuit. Second, it enhances this two-key spooky encryption scheme into a scheme that supports d hops of (interleaved) two-key spooky multiplications, and single-key homomorphic addition (for any polynomial $d(\lambda)$). This second step is done by applying Canetti et al's piO-based transformation of an encryption scheme into a d-leveled encryption scheme [Can+15]. Eventually, the authors observed that a scheme that supports both single key homomorphic addition and two-key spooky multiplication leads to a leveled AFS-spooky encryption scheme for all circuits, drawing upon an elegant connection to the GMW protocol for multiparty computation.

We therefore start by building a dpiO-based two-key spooky encryption scheme for bit inputs and evaluation of spooky multiplication. The inputs to the obfuscated evaluation circuit P are tuples of the form $(\mathsf{pk}_1, \mathsf{pk}_2, c_1, \mathsf{pk}, c)$, where $\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{pk}$ are public keys, c_1 is a ciphertext under pk_1, and c is a ciphertext under pk. P will decrypt c_1 with sk_1 to some x, encrypt with pk_2 a plaintext computed from x, and homomorphically evaluate (using pk) the corresponding second part of the M-spooky evaluation on c. Toward obfuscating P with a dpiO scheme, we define a sampler with input S which has five possible states. With the first four states, it generates the appropriate input $(\mathsf{pk}_1, \mathsf{pk}_2, c_1, \mathsf{pk}, c)$ for P. With the last state, it generates the output of P. Note that unlike in the LHE construction of the previous section, S must be stateful to allow for sampling, e.g., c_1 as a fresh ciphertext under the public key pk_1, which is itself sampled with S.

While this would suffice to construct a dpiO-based two-key M-spooky encryption scheme, we cannot apply the second step of [Dod+16] to transform it into an n-key spooky encryption scheme for all circuits. Indeed, the transformation requires to enhance the M-spooky scheme into a leveled-homomorphic M-spooky scheme. It was done in [Dod+16] using the piO-based approach of [Can+15]; here, we would like to follow the same approach, but using our dpiO-based construction of LHE from the previous section. However, for this to work, we need to ensure that all inputs to the (dpiO-obfuscated) leveled homomorphic evaluation circuits are appropriately authenticated samples from the encryption algorithm (with respect to some hardcoded public key). This is not the case, as ciphertexts obtained by evaluating P are not distributed as fresh ciphertexts. To overcome this issue, we therefore add to the public key of our two-key scheme a dpiO-obfuscated circuit that performs a re-encryption procedure. More precisely, it takes all the inputs of P, plus the output of P, and checks that they were correctly constructed as samples with S. Then, it decrypts the outputs of P, and re-encrypt them freshly.

With this last modification, the outputs of a spooky evaluation procedure are generated as fresh ciphertexts and authenticated as such. By replacing the encryption scheme E in the LHE construction of the previous section with our two-key M-spooky encryption scheme, the correctness argument follows as before and we get a 2-key M-spooky encryption scheme that supports leveled (single-key) homomorphic evaluation. We can therefore invoke Theorem 9 from [Dod+16], which states that such a scheme directly implies an n-spooky encryption scheme for all circuits.

6.3 Two-Key Spooky Encryption for Bit-Inputs

We proceed with a construction of a two-key M-spooky encryption scheme for bit-inputs.

Sampler with Input for Spooky Encryption. Let $E = (E.\mathsf{KeyGen}, E.\mathsf{Enc}, E.\mathsf{Dec}, E.\mathsf{Eval})$ be a maliciously circuit-private homomorphic encryption scheme for NC_1. Fix any security parameter $\lambda \in \mathbb{N}$. Let $\mathcal{T}_\lambda \leftarrow [3] \times (\{0,1\}^\lambda \cup \{\bot\})^3$. We denote by $\mathsf{st}_\lambda : (\{0,1\}^\lambda \cup \{\bot\})^5 \mapsto \mathcal{T}_\lambda$ the following state function:

- on input $(y_1, \bot, \bot, \bot, \bot)$ with any $y_1 \in \{0,1\}^\lambda \cup \{\bot\}$, st_λ output $(1, \bot, \bot, \bot)$;
- on input $(y_1, y_2, \bot, \bot, \bot)$ with $y_2 \neq \bot$, st_λ output $(2, y_1, \bot, \bot)$;
- on input $(y_1, y_2, y_3, \bot, \bot)$ with $y_3 \neq \bot$, st_λ output $(1, \bot, \bot, \bot)$;
- on input $(y_1, y_2, y_3, y_4, \bot)$ with $y_4 \neq \bot$, st_λ output $(2, y_4, \bot, \bot)$;
- on input $(y_1, y_2, y_3, y_4, y_5)$ with $y_4 \neq \bot$, st_λ output $(3, y_2, y_4, y_5)$.

We now describe the sampler with input S, with input domain $\mathcal{T}_\lambda \times \{0,1\}$. On input (state, x), S parses state as (i, y, y', y''), and does the following:

- if $i = 1$, run $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} E.\mathsf{KeyGen}(1^\lambda)$, and output $(\mathsf{pk}; \mathsf{sk})$.
- if $i = 2$, run $c \xleftarrow{\$} E.\mathsf{Enc}(y, x)$, and output $(c; \bot)$;
- if $i = 3$, pick random coins (r, r', r_1), define

$$M'[x, r, r'](\cdot) \equiv \mathsf{RS}_M(x, \cdot, M(x, 0; r)_1; r'),$$

  set $c'_1 \leftarrow E.\mathsf{Enc}(y, M(x, 0; r)_1; r_1)$ and $c'_2 \leftarrow E.\mathsf{Eval}(y', M'[x, r, r'], y'')$, and output $((c'_1, c'_2); \bot)$.

Description of the Scheme. Let $\mathcal{SI} = \{\mathcal{SI}_\lambda\}_{\lambda\in\mathbb{N}}$ be the family of samplers with input as defined in the previous paragraph, with input domain $\mathcal{T}_\lambda \times \{0,1\}$, where each $\mathcal{SI}_\lambda$ contains a single sampler with input S, with state function st_λ. Let $(\mathsf{Setup}, \mathsf{Encode}, \mathsf{Obfuscate})$ be a 5-source dpiO scheme for $(\mathsf{st}_\lambda, \mathcal{SI}, \mathcal{C}, \mathbf{CS})$, where $\mathcal{C}$ contains all circuits of the following form:

- the programs $P[\mathsf{sk}_1, \mathsf{pk}_1, \mathsf{pk}_2, \mathsf{pp}_2, S]$ represented on Figure 11 for all strings $(\mathsf{sk}_1, \mathsf{pk}_1, \mathsf{pk}_2, \mathsf{pp}_2)$;
- the programs $Q_j[\mathsf{sk}_{3-j}, \mathsf{pk}_3, \mathsf{pp}_3, S]$ represented on Figure 11 for all strings $(\mathsf{sk}_{3-j}, \mathsf{pk}_3, \mathsf{pp}_3)$ for $j = 1, 2$;
- the programs $\hat{P}[\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{pp}_2, \mathsf{trap}_2, S]$ represented on Figure 13 for all strings $(\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{pp}_2, \mathsf{trap}_2)$;
- the programs $\hat{Q}_j[\mathsf{pk}_3, \mathsf{pp}_3, S, \mathsf{trap}_3]$ represented on Figure 12 for all strings for $j = 1, 2$,

and the circuit sampler $\mathbf{CS}$ randomly samples keys $(\mathsf{sk}_i, \mathsf{pk}_i)$ sampled with $S_1((1,\bot,\bot,\bot), 0)$, public parameters $\mathsf{pp}_2, \mathsf{pp}_3$ with Setup (or with the simulator algorithm for the Setup when it must generate $(\mathsf{pp}_2, \mathsf{trap}_2)$ or $(\mathsf{pp}_3, \mathsf{trap}_3)$), and outputs either $(C_0, C_1, z) = (P[\mathsf{sk}_1, \mathsf{pk}_1, \mathsf{pk}_2, \mathsf{pp}_2, S], \hat{P}[\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{pp}_2, \mathsf{trap}_2, S], (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{pp}_2))$ or $(C_0, C_1, z) = (Q_j[\mathsf{sk}_{3-j}, \mathsf{pk}_3, \mathsf{pp}_3, S], \hat{Q}_j[\mathsf{pk}_3, \mathsf{pp}_3, S, \mathsf{trap}_3], (\mathsf{pk}_3, \mathsf{pp}_3))$ for $j = 1$ or $j = 2$. We will prove that this circuit sampler outputs pairs of circuits satisfying dynamic-input indistinguishability in the security analysis of our protocol. We represent on Figure 11 our construction of a two-key M-spooky encryption scheme for bit inputs.

Theorem 25. *Let* dpiO *be a* 5*-source dpIO scheme for* $(\mathsf{st}, \mathcal{SI}, \mathcal{C}, \mathbf{CS})$*, and let* E *be an* $\mathsf{IND\text{-}CPA}$*-secure circuit-private (non-compact) homomorphic encryption scheme for one-bit functions. Then* SE *defined in Figure 11 is a two-key* M*-spooky encryption scheme for bit inputs.*

Plugging in our construction of dpiO of the previous section, and the DDH-based construction of a circuit-private encryption scheme [Dod+16], we get the following corollary:

Corollary 26. *Assuming polynomially-secure indistinguishability obfuscation and* eDDH*, there exists a two-key* M*-spooky encryption scheme for bit inputs.*

Proof. We first show that the scheme SE is complete and M-spooky. Consider the following simplification of the scheme of the Figure 11: the algorithm $\mathsf{SE.Setup}$ is removed, all invocations of (S_1, S_2, S_3) are replaced by invocations of S (hence all auxiliary values are removed), the obfuscated programs $\tilde{Q}_1, \tilde{Q}_2$ are removed (and $\mathsf{SE.Eval}$ outputs $((2, c'_1), (1, c'_2))$ directly), and the program P is obfuscated using a standard piO scheme. The new scheme SE' obtained this way is exactly the piO-based spooky encryption scheme of [Dod+16]; we refer the reader to [Dod+16] for a thorough analysis of its properties.

Here, we focus on showing that our modifications do not alter the completeness and M-spooky properties. Let us first show completeness of the encryption and decryption algorithms. Let $x \in \{0,1\}$. Let $\mathsf{crs} \xleftarrow{\$} \mathsf{SE.Setup}(1^\lambda)$. For any $j \in \{1,3\}$, let $(\mathsf{pk}_j, \mathsf{aux}_j; \mathsf{sk}_j) \xleftarrow{\$} S_j((1,\bot,\bot,\bot), 0)$, let $(c, \mathsf{aux}) \xleftarrow{\$} S_j((2, \mathsf{pk}_j, \bot, \bot), x)$ and let $x' \leftarrow E.\mathsf{Dec}(\mathsf{sk}_j, c)$. It necessarily holds that $x' = x$: otherwise, this would distinguish the output distribution of S_j from the output distribution of S (as $S((1,\bot,\bot,\bot), 0)$ outputs a random key pair $(\mathsf{pk}, \mathsf{sk})$ for E, and $S((2, \mathsf{pk}, \bot, \bot), x)$ outputs a random encryption of x under E).

It remains to show the correctness of the spooky evaluation procedure. First, we show that the outputs (c'_1, c'_2) of $\tilde{P}^0$ are encryptions (under pk_2^0 and pk_1^1 respectively) of the spooky evaluation of M on the plaintexts of (c^0, c^1). Observe that the inputs to $\tilde{P}^0$ satisfy the requirements of the statistical restricted correctness of the dpiO scheme. It follows immediatly from the definition of $\mathsf{SE.KeyGen}$ and $\mathsf{SE.Enc}$ that the inputs $(\mathsf{pk}_1^0, \mathsf{pk}_2^0, c^0, \mathsf{pk}_1^1, c^1)$ are appropriately authenticated inputs sampled using $S_1 = \mathsf{Encode}(\mathsf{pp}_1, S)$, with the appropriate auxiliary inputs, with respective states

$$(1,\bot,\bot,\bot), (1,\bot,\bot,\bot), (2, \mathsf{pk}_1^0, \bot, \bot), (1,\bot,\bot,\bot), (2, \mathsf{pk}_1^1, \bot, \bot).$$

Therefore, by the restricted correctness of dpiO, the correctness analysis of [Dod+16] applies and allows to conclude that (c'_1, c'_2) of $\tilde{P}^0$ are encryptions (under pk_2^0 and pk_1^1 respectively) of the spooky evaluation of M on the plaintexts of (c^0, c^1). We now argue that the outputs c''_1 and c''_2 are encryptions (under pk_3^0 and pk_3^1 respectively) of the spooky evaluation of M on the plaintexts of (c^0, c^1). As Q_1 and Q_2 do only perform decryption (of c''_1 and c''_2 respectively) and re-encryption with pk_3, it suffices to show that the inpus to $(\tilde{Q}_1, \tilde{Q}_2)$ satisfy the requirements of the statistical restricted correctness of the dpiO scheme. This follows immediatly by our previous observation that the inputs $(\mathsf{pk}_1^0, \mathsf{pk}_2^0, c^0, \mathsf{pk}_1^1, c^1)$ are appropriately

$P[\mathsf{sk}_1, \mathsf{pk}_1, \mathsf{pk}_2, \mathsf{pp}_2, S](y_1, y_2, c_1, y, c)$

if $\mathsf{pk}_i \neq y_i$ for $i \in \{1,2\}$ **return** $\perp$
$x_1 \leftarrow E.\mathsf{Dec}(\mathsf{sk}_1, c_1)$
$S_2 \leftarrow \mathsf{Encode}(\mathsf{pp}_2, S)$
$(c_1', c_2', \mathsf{aux}) \xleftarrow{\$} S_2((3, y_2, y, c), x_1)$
return $((2, c_1'), (1, c_2'), \mathsf{aux})$

$Q_j[\mathsf{sk}_{3-j}, \mathsf{pk}_3, \mathsf{pp}_3, S](y_1, y_2, c_1, y, c, (c_1', c_2'))$

if $\mathsf{pk}_i \neq y_i$ for $i \in \{1,2\}$ **return** $\perp$
$x_j' \leftarrow E.\mathsf{Dec}(\mathsf{sk}_{3-j}, c_j')$
$S_3 \leftarrow \mathsf{Encode}(\mathsf{pp}_3, S)$
$(c_j'', \mathsf{aux}) \xleftarrow{\$} S_3((2, \mathsf{pk}_3, \perp, \perp), x_j')$
return $(3, c_j'', \mathsf{aux})$

$\mathsf{SE.Setup}(1^\lambda)$

$\mathsf{pp}_1 \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$
$\mathsf{pp}_2 \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$
$\mathsf{pp}_3 \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$
$\mathsf{crs} \leftarrow (\mathsf{pp}_1, \mathsf{pp}_2, \mathsf{pp}_3)$
return crs

$\mathsf{SE.KeyGen}(\mathsf{crs})$

parse crs as $(\mathsf{pp}_1, \mathsf{pp}_2, \mathsf{pp}_3)$
for $j \in \{1,3\}$, $S_j \leftarrow \mathsf{Encode}(\mathsf{pp}_j, S)$
$(\mathsf{pk}_1, \mathsf{aux}_1; \mathsf{sk}_1) \xleftarrow{\$} S_1((1, \perp, \perp, \perp), 0)$
$(\mathsf{pk}_2, \mathsf{aux}_2; \mathsf{sk}_2) \xleftarrow{\$} S_2((1, \perp, \perp, \perp), 0)$
$(\mathsf{pk}_3, \mathsf{aux}_3; \mathsf{sk}_3) \xleftarrow{\$} S_3((1, \perp, \perp, \perp), 0)$
$\tilde{P} \xleftarrow{\$} \mathsf{Obfuscate}(\mathsf{pp}_1, S, P[\mathsf{sk}_1, \mathsf{pk}_1, \mathsf{pk}_2, \mathsf{pp}_2, S])$
for $j \in \{1,2\}$,
$\tilde{Q}_j \xleftarrow{\$} \mathsf{Obfuscate}(\mathsf{pp}_1[1-5], \mathsf{pp}_2[6], S, Q_j[\mathsf{sk}_{3-j}, \mathsf{pk}_3, \mathsf{pp}_3, S])$
$\mathsf{sk} \leftarrow (\mathsf{sk}_1, \mathsf{sk}_2, \mathsf{sk}_3)$
$\mathsf{pk} \leftarrow (\mathsf{pk}_1, \mathsf{aux}_1, \mathsf{pk}_2, \mathsf{aux}_2, \mathsf{pk}_3, \mathsf{aux}_3, \tilde{P}, \tilde{Q}_1, \tilde{Q}_2)$
return $(\mathsf{pk}, \mathsf{sk})$

$\mathsf{SE.Enc}(\mathsf{crs}, \mathsf{pk}, x)$

parse crs as $(\mathsf{pp}_1, \mathsf{pp}_2, \mathsf{pp}_3)$, and parse pk as $(\mathsf{pk}_1, \mathsf{aux}_1, \mathsf{pk}_2, \mathsf{aux}_2, \mathsf{pk}_3, \mathsf{aux}_3, \tilde{P}, \tilde{Q}_1, \tilde{Q}_2)$
$S_1 \leftarrow \mathsf{Encode}(\mathsf{pp}_1, S)$
$(c, \mathsf{aux}) \xleftarrow{\$} S_1((2, \mathsf{pk}_1, \perp, \perp), x)$
return $(1, c, \mathsf{aux})$

$\mathsf{SE.Dec}((\mathsf{sk}_1, \mathsf{sk}_2, \mathsf{sk}_3), i, c)$

return $E.\mathsf{Dec}(\mathsf{sk}_i, c)$

$\mathsf{SE.Eval}(\mathsf{pk}^0, c^0, \mathsf{aux}^0, \mathsf{pk}^1, c^1, \mathsf{aux}^1)$

parse pk^0 as $(\mathsf{pk}_1^0, \mathsf{aux}_1^0, \mathsf{pk}_2^0, \mathsf{aux}_2^0, \mathsf{pk}_3^0, \mathsf{aux}_3^0, \tilde{P}^0, {\tilde{Q}_1}^0, {\tilde{Q}_2}^0)$
parse pk^1 as $(\mathsf{pk}_1^1, \mathsf{aux}_1^1, \mathsf{pk}_2^1, \mathsf{aux}_2^1, \mathsf{pk}_3^1, \mathsf{aux}_3^1, \tilde{P}^1, {\tilde{Q}_1}^1, {\tilde{Q}_2}^1)$
$((2, c_1'), (1, c_2'), \mathsf{aux}) \leftarrow \tilde{P}^0(\mathsf{pk}_1^0, \mathsf{aux}_1^0, \mathsf{pk}_2^0, \mathsf{aux}_2^0, c^0, \mathsf{aux}^0, \mathsf{pk}_1^1, \mathsf{aux}_1^1, c^1, \mathsf{aux}^1)$
$(3, c_1'', \mathsf{aux}_1'') \leftarrow \tilde{Q}_1(pk_1^0, \mathsf{aux}_1^0, \mathsf{pk}_2^0, \mathsf{aux}_2^0, c^0, \mathsf{aux}^0, \mathsf{pk}_1^1, \mathsf{aux}_1^1, c^1, \mathsf{aux}^1, (c_1', c_2', \mathsf{aux}))$
$(3, c_2'', \mathsf{aux}_2'') \leftarrow \tilde{Q}_2(pk_1^0, \mathsf{aux}_1^0, \mathsf{pk}_2^0, \mathsf{aux}_2^0, c^0, \mathsf{aux}^0, \mathsf{pk}_1^1, \mathsf{aux}_1^1, c^1, \mathsf{aux}^1, (c_1', c_2', \mathsf{aux}))$
return $((3, c_1'', \mathsf{aux}_1''), (3, c_2'', \mathsf{aux}_2''))$

Fig. 11. dpiO-based construction of a two-key M-spooky encryption scheme for bit inputs.

authenticated inputs sampled using $S_1 = \mathsf{Encode}(\mathsf{pp}_1, S)$, with the appropriate auxiliary inputs, and by observing that (c_1', c_2') are appropriately authenticated inputs sampled using $S_2 = \mathsf{Encode}(\mathsf{pp}, 2)$, with state $(3, \mathsf{pk}_2^0, \mathsf{pk}_1^1, c^1) = \mathsf{st}_\lambda(\mathsf{pk}_1^0, \mathsf{pk}_2^0, c^0, \mathsf{pk}_1^1, c^1)$.

Let us now show that the scheme $(\mathsf{SE.Setup}, \mathsf{SE.KeyGen}, \mathsf{SE.Enc}, \mathsf{SE.Dec})$ is an IND-CPA-secure bit encryption scheme. Let $\mathcal{A}$ be an adversary against the IND-CPA-security of the scheme. Recall that the IND-CPA security notion for bit encryption schemes is equivalent to the following experiment: the challenger picks $b \xleftarrow{\$} \{0,1\}$, samples $\mathsf{crs} \xleftarrow{\$} \mathsf{SE.Setup}(1^\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{SE.KeyGen}(\mathsf{crs})$, and $c \xleftarrow{\$} \mathsf{SE.Enc}(\mathsf{pk}, b)$. It sends $(\mathsf{crs}, \mathsf{pk}, c)$ to $\mathcal{A}$; the adversary wins the game if it correctly guesses b. We prove security through a sequence of games.

Game $\mathbf{G}_0$: This is the normal game, where the challenger picks $b \xleftarrow{\$} \{0,1\}$, samples $\mathsf{crs} \xleftarrow{\$} \mathsf{SE.Setup}(1^\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{SE.KeyGen}(\mathsf{crs})$, and $c \xleftarrow{\$} \mathsf{SE.Enc}(\mathsf{pk}, b)$.

Game $\mathbf{G}_1$: In this game, we use the simulatability of encodings property of the dpiO scheme. Let $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ be the simulator whose existence is guaranteed by Definition 10. The challenger computes $(\mathsf{pp}_1, \mathsf{trap}_1) \xleftarrow{\$} \mathsf{Sim}_0(1^\lambda)$, $(\mathsf{pp}_2, \mathsf{trap}_2) \xleftarrow{\$} \mathsf{Sim}_0(1^\lambda)$, $(\mathsf{pp}_3, \mathsf{trap}_3) \xleftarrow{\$} \mathsf{Sim}_0(1^\lambda)$, and sends $\mathsf{crs} \leftarrow (\mathsf{pp}_1, \mathsf{pp}_2, \mathsf{pp}_3)$. It

computes (pk, c) as before. This game is indistinguishable from the previous one, by the simulatability of encodings property of the dpiO scheme.

Game $\mathbf{G_2}$: In this game, we modify the generation of pk by the challenger. We divide this game into two successive hybrids, for $j = 1$ and $j = 2$, where the challenger generates pk as follows: instead of building $\tilde{Q}_j$ as $\mathsf{Obfuscate}(\mathsf{pp}_1[1-5], \mathsf{pp}_2[6], S, Q_j[\mathsf{sk}_{3-j}, \mathsf{pk}_3, \mathsf{pp}_3, S])$ in $\mathsf{SE.KeyGen}(\mathsf{crs})$, the challenger sets $\tilde{Q}_j \leftarrow \mathsf{Obfuscate}(\mathsf{pp}_1[1-5], \mathsf{pp}_2[6], S, \hat{Q}_j[\mathsf{pk}_3, \mathsf{pp}_3, \mathsf{trap}_3, S])$ instead, with the program $\hat{Q}_j$ represented on Figure 12. The indistinguishability argument proceeds by showing that Q_j and $\hat{Q}_j$ are dynamic-input indistinguishable. The argument proceeds in 4 steps.

1. Consider the program Q_j^0 represented on Figure 12. Its only difference with Q_j is that it generates c_j'' through S instead of S_3, and generate the auxiliary input using Sim_1. By the simulatability of encodings property of the dpiO scheme, the output distribution of Q_j and Q_j^0 are computationally indistinguishable (even on adversarially chosen inputs), hence Q_j and Q_j^0 are dynamic-input indistinguishable.
2. Consider the program Q_j^1 represented on Figure 12. By definition of S, c_j'' is now a uniformly random encryption (with E) of 0 under the key pk_3. Note that the corresponding secret key sk_3 is not used in any program. Therefore, under the $\mathsf{IND\text{-}CPA}$ security property of E, Q_j^1 is dynamic-input indistinguishable from Q_j^0.
3. Consider the program Q_j^2 represented on Figure 12. It is identical to Q_j^1, except that it does not have sk_{3-j} hardcoded in its description anymore (note that Q_j^1 does not use this hardcoded key). Therefore, Q_j^1 and Q_j^2 are functionally equivalent.
4. Observe now that the only difference between Q_j^2 and $\hat{Q}_j$, represented on Figure 12, is that $\hat{Q}_j$ generates (c_j'', aux) using S_3, while Q_j^2 uses S and Sim_1. Therefore, by the simulatability of encodings property of the dpiO scheme, Q_j^2 and $\hat{Q}_j$ are dynamic-input indistinguishable.

From there, the indistinguishability property of the dpiO scheme allows to conclude that the obfuscation of Q_j and the obfuscation of $\hat{Q}_j$ are computationally indistinguishable, hence this game is indistinguishable from the previous one.

$Q_j^0[\mathsf{sk}_{3-j}, \mathsf{pk}_3, \mathsf{pp}_3, \mathsf{trap}_3, S](y_1, y_2, c_1, y, c, (c_1', c_2'))$

if $\mathsf{pk}_i \neq y_i$ for $i \in \{1,2\}$ **return** $\perp$
$x_j' \leftarrow E.\mathsf{Dec}(\mathsf{sk}_{3-j}, c_j')$
$c_j'' \xleftarrow{\$} S((2, \mathsf{pk}_3, \perp, \perp), x_j')$
$\mathsf{aux} \xleftarrow{\$} \mathsf{Sim}_1(\mathsf{pp}_3, \mathsf{trap}_3, S, c_j'', (2, \mathsf{pk}_3, \perp, \perp))$
return $(3, c_j'', \mathsf{aux})$

$Q_j^1[\mathsf{sk}_{3-j}, \mathsf{pk}_3, \mathsf{pp}_3, \mathsf{trap}_3, S](y_1, y_2, c_1, y, c, (c_1', c_2'))$

if $\mathsf{pk}_i \neq y_i$ for $i \in \{1,2\}$ **return** $\perp$
$x_j' \leftarrow E.\mathsf{Dec}(\mathsf{sk}_{3-j}, c_j')$
$c_j'' \xleftarrow{\$} S((2, \mathsf{pk}_3, \perp, \perp), 0)$
$\mathsf{aux} \xleftarrow{\$} \mathsf{Sim}_1(\mathsf{pp}_3, \mathsf{trap}_3, S, c_j'', (2, \mathsf{pk}_3, \perp, \perp))$
return $(3, c_j'', \mathsf{aux})$

$Q_j^2[\mathsf{pk}_3, \mathsf{pp}_3, \mathsf{trap}_3, S](y_1, y_2, c_1, y, c, (c_1', c_2'))$

if $\mathsf{pk}_i \neq y_i$ for $i \in \{1,2\}$ **return** $\perp$
$c_j'' \xleftarrow{\$} S((2, \mathsf{pk}_3, \perp, \perp), 0)$
$\mathsf{aux} \xleftarrow{\$} \mathsf{Sim}_1(\mathsf{pp}_3, \mathsf{trap}_3, S, c_j'', (2, \mathsf{pk}_3, \perp, \perp))$
return $(3, c_j'', \mathsf{aux})$

$\hat{Q}_j[\mathsf{pk}_3, \mathsf{pp}_3, \mathsf{trap}_3, S](y_1, y_2, c_1, y, c, (c_1', c_2'))$

if $\mathsf{pk}_i \neq y_i$ for $i \in \{1,2\}$ **return** $\perp$
$S_3 \leftarrow \mathsf{Encode}(\mathsf{pp}_3, S)$
$(c_j'', \mathsf{aux}) \xleftarrow{\$} S_3((2, \mathsf{pk}_3, \perp, \perp), 0)$
return $(3, c_j'', \mathsf{aux})$

Fig. 12. Successive transformations of the program Q_j.

Game $\mathbf{G_3}$: In this game, we further modify the generation of pk by the challenger. Instead of generating $\tilde{P}$ as $\mathsf{Obfuscate}(\mathsf{pp}_1, S, P[\mathsf{sk}_1, \mathsf{pk}_1, \mathsf{pk}_2, \mathsf{pp}_2, S])$, the challenger sets $\tilde{P} \leftarrow \mathsf{Obfuscate}(\mathsf{pp}_1, S, \hat{P}[\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{pp}_2, \mathsf{trap}_2, S])$ instead, with the program $\hat{P}$ represented on Figure 13. The proof proceeds by showing that P and $\hat{P}$ are dynamic-input indistinguishable. This is done in the same way as in the previous game, by exhibiting a sequence of programs $P, P^0, P^1, P^2 2, \hat{P}$ which are all dynamic-input indistinguishable. The programs are represented on Figure 13. Indistinguishability between P and P^0 follows by the same argument as in step 1 of the previous game. P^1 is constructed by replacing all occurences of x_1 by 0 in P^0.

Lemma 27. *The circuits P^0 and P^1 are dynamic-input indistinguishable.*

Proof. The indistinguishability argument is essentially identical to the proof of the claim 4.10.3 of [Dod+16]. We recall it briefly for the sake of completeness. Observe that in P^0, c_1' is distributed as a random encryption of x_1, and c_2' is the output of an homomorphic evaluation on c. We consider an intermediate programs $P^{0.1}$. Tt proceeds as P^0, except that it uses instead of S a sampler $S^{0.1}$ which, on input the state $(3, y_2, y, c)$ and x_1, samples the output c_1' as a random encryption of 0 instead (it proceeds as S otherwise). Note that the

encryption of c'_1 is under pk_2, whose corresponding secrey key is not known to the program. Therefore, under the IND-CPA-security property of E, $P^{0.1}$ is dynamic-input indistinguishable from P^0.
Now, the only difference between $P^{0.1}$ and P^1 is that c'_2 is obtained by homomorphically evaluating $M'[x_1, r, r']$ on c in $P^{0.1}$, and $M'[0, r, r']$ on c in P^1. By the malicious circuit privacy of E, c'_2 is therefore distributed as an encryption with fresh random coin (of some plaintext under pk) in both $P^{0.1}$ and P^1. Therefore, by the IND-CPA-security property of E, $P^{0.1}$ and P^1 are dynamic-input indistinguishable. □

Then, the circuit P^2 represented on Figure 13 is obtained by not harcoding sk_1 in P^1 anymore; as in step 3 of the previous game, P^2 is functionally equivalent to P^1. Eventually, P^2 and $\hat{P}$ are dynamic-input indistinguishable by the simulatability of encodings property of the dpiO scheme, using the same argument as step 4 of the previous game. From there, the indistinguishability property of the dpiO scheme allows to conclude that the obfuscation of P and the obfuscation of $\hat{P}$ are computationally indistinguishable, hence this game is indistinguishable from the previous one.

$P^0[\mathsf{sk}_1, \mathsf{pk}_1, \mathsf{pk}_2, \mathsf{pp}_2, \mathsf{trap}_2, S](y_1, y_2, c_1, y, c)$

if $\mathsf{pk}_i \neq y_i$ for $i \in \{1, 2\}$ **return** $\perp$
$x_1 \leftarrow E.\mathsf{Dec}(\mathsf{sk}_1, c_1)$
$(c'_1, c'_2) \xleftarrow{\$} S((3, y_2, y, c), x_1)$
$\mathsf{aux} \xleftarrow{\$} \mathsf{Sim}_1(\mathsf{pp}_2, \mathsf{trap}_2, S, (c'_1, c'_2), (3, y_2, y, c))$
return $((2, c'_1), (1, c'_2), \mathsf{aux})$

$P^1[\mathsf{sk}_1, \mathsf{pk}_1, \mathsf{pk}_2, \mathsf{pp}_2, \mathsf{trap}_2, S](y_1, y_2, c_1, y, c)$

if $\mathsf{pk}_i \neq y_i$ for $i \in \{1, 2\}$ **return** $\perp$
$x_1 \leftarrow E.\mathsf{Dec}(\mathsf{sk}_1, c_1)$
$(c'_1, c'_2) \xleftarrow{\$} S((3, y_2, y, c), 0)$
$\mathsf{aux} \xleftarrow{\$} \mathsf{Sim}_1(\mathsf{pp}_2, \mathsf{trap}_2, S, (c'_1, c'_2), (3, y_2, y, c))$
return $((2, c'_1), (1, c'_2), \mathsf{aux})$

$P^2[\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{pp}_2, \mathsf{trap}_2, S](y_1, y_2, c_1, y, c)$

if $\mathsf{pk}_i \neq y_i$ for $i \in \{1, 2\}$ **return** $\perp$
$(c'_1, c'_2) \xleftarrow{\$} S((3, y_2, y, c), 0)$
$\mathsf{aux} \xleftarrow{\$} \mathsf{Sim}_1(\mathsf{pp}_2, \mathsf{trap}_2, S, (c'_1, c'_2), (3, y_2, y, c))$
return $((2, c'_1), (1, c'_2), \mathsf{aux})$

$\hat{P}[\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{pp}_2, \mathsf{trap}_2, S](y_1, y_2, c_1, y, c)$

if $\mathsf{pk}_i \neq y_i$ for $i \in \{1, 2\}$ **return** $\perp$
$S_2 \leftarrow \mathsf{Encode}(\mathsf{pp}_2, S)$
$(c'_1, c'_2, \mathsf{aux}) \xleftarrow{\$} S_2((3, y_2, y, c), 0)$
return $((2, c'_1), (1, c'_2), \mathsf{aux})$

Fig. 13. Successive transformations of the program P.

Game $\mathbf{G}_4$: In this game, the challenger sets $c \xleftarrow{\$} \mathsf{SE.Enc}(\mathsf{pk}, 0)$ instead of $c \xleftarrow{\$} \mathsf{SE.Enc}(\mathsf{pk}, b)$. By definition of SE.Enc, c is now computed as $(c, \mathsf{aux}) \leftarrow S_1((2, \mathsf{pk}_1), x)$, with $x = b$ in the previous game, and $x = 0$ in this game. By definition of S, it follows that c is a random encryption under E of x, with key pk_1. Observe that the key sk_1 is not hardcoded anymore in any of the obfuscated circuits of the public key pk. Therefore, this game is indistinguishable from the previous game under the IND-CPA security property of E. Observe now that in the current game, the output of the challenger does not depend on b anymore, hence the advantage of $\mathcal{A}$ in this game is 0. This concludes the proof that $(\mathsf{SE.Setup}, \mathsf{SE.KeyGen}, \mathsf{SE.Enc}, \mathsf{SE.Dec})$ is an IND-CPA-secure bit encryption scheme.

□

6.4 From Two-Key M-Spooky Encryption to n-Key Spooky Encryption for all Circuits

By plugging our construction of two-key M-spooky encryption scheme in the transformation of the previous section, we obtain a two-key M-spooky and (single-key) leveled homomorphic encryption scheme. More precisely, let us sketch how to integrate our scheme to the LHE construction of the previous section. Our two-key M-spooky scheme SE requires three public keys of E, $(\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{pk}_3)$. Fresh ciphertexts are samples with S corresponding to encryptions with E under pk_1, and outputs of a spooky-evaluation are samples with S corresponding to encryptions with E under pk_3. To simplify the integration, we add to the public key of SE an (obfuscated) re-encryption circuit, which takes fresh ciphertexts under pk_1 as input (sampled with S), performs decryption with sk_1, and outputs fresh re-encryption with E sampled with S under pk_3.

The obfuscated circuit for homomorphic operations takes as input two ciphertexts under pk_3 (sampled with S), decrypts them with sk_3, computes a XOR gate,[15] and re-encrypts the result under a new key pk_4. We can generalize this to d levels of interleaved two-key spooky evaluations and single-key homomorphic evaluations by using $3d$ public keys for E. The computation at each level i will output the result of either a spooky evaluation or an homomorphic evaluation, under the public key pk_{3i+1}. Note that the output of a

[15] homomorphic evaluation of XOR gate is sufficient for the transformation of [Dod+16].

spooky evaluation procedure at level i are encrypted under pk_{3i}, but given an encryption of x under pk_{3i}, one can always encrypt 0 under pk_{3i} and homomorphically XOR x with 0, obtaining an encryption of x under the key pk_{3i+1}.

From there, we can invoke Theorem 9 of [Dod+16]:

Theorem 28 (2-Spooky to n-Spooky [Dod+16]). *Let $d = d(\lambda)$ and assume that there exists a public-key bit encryption cheme that supports $2d$ (interleaving) hops of (1) single-key compact additive homomorphism and (2) two-key spooky multiplication. Then, that same scheme is a d-level AFS-spooky encryption for all circuits.*

The proof of Theorem 9 can be found in [Dod+16]. It follows from an elegant connection with the GMW protocol: it evaluates an arbitrary circuit gate by gate, given n encrypted shares under n different public keys $(\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$ of the value on a gate. If the gate is a XOR, it performs an homomorphic XOR operations on each pair of shares encrypted with the same public key. If its an AND gate, it performs a two-key spooky multiplication for each pair of public keys, between the shares encrypted under these two keys, obtaining shares of the products. Then, it homomorphically XOR all outputs which are encrypted under the same public keys (if there are n keys, this requires $\log n$ levels of 2-input homomorphic XOR). It follows by inspection that the resulting n ciphertexts encrypt additive shares of the output of the gate, under the n different public keys $(\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$.

Using our construction of a dpiO scheme (Corollary 14) together with Theorem 28, we obtain:

Theorem 29. *Assuming polynomially-secure indistinguishability obfuscation and eDDH, there exists an n-key spooky encryption scheme for all circuits.*

As observed in [Dod+16], this implies a de-centralized function secret sharing scheme for all circuits, and a counter-example to the [Aie+00] heuristics (which transforms multi-prover protocols to single-prover protocols), under the same assumptions.

References

[Aie+00] William Aiello, Sandeep N. Bhatt, Rafail Ostrovsky, and Sivaramakrishnan Rajagopalan. "Fast Verification of Any Remote Procedure Call: Short Witness-Indistinguishable One-Round Proofs for NP". In: *ICALP 2000*. Ed. by Ugo Montanari, José D. P. Rolim, and Emo Welzl. Vol. 1853. LNCS. Springer, Heidelberg, July 2000, pp. 463–474. DOI: 10.1007/3-540-45022-X_39 (cit. on p. 30).

[AJ15] Prabhanjan Ananth and Abhishek Jain. "Indistinguishability Obfuscation from Compact Functional Encryption". In: *CRYPTO 2015, Part I*. Ed. by Rosario Gennaro and Matthew J. B. Robshaw. Vol. 9215. LNCS. Springer, Heidelberg, Aug. 2015, pp. 308–326. DOI: 10.1007/978-3-662-47989-6_15 (cit. on p. 3).

[Ana+13] Prabhanjan Ananth, Dan Boneh, Sanjam Garg, Amit Sahai, and Mark Zhandry. *Differing-Inputs Obfuscation and Applications*. Cryptology ePrint Archive, Report 2013/689. http://eprint.iacr.org/2013/689. 2013 (cit. on pp. 3, 6).

[AS17] Prabhanjan Ananth and Amit Sahai. "Projective Arithmetic Functional Encryption and Indistinguishability Obfuscation from Degree-5 Multilinear Maps". In: *EUROCRYPT 2017, Part I*. Ed. by Jean-Sébastien Coron and Jesper Buus Nielsen. Vol. 10210. LNCS. Springer, Heidelberg, 2017, pp. 152–181. DOI: 10.1007/978-3-319-56620-7_6 (cit. on p. 3).

[Bar+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. "On the (Im)possibility of Obfuscating Programs". In: *CRYPTO 2001*. Ed. by Joe Kilian. Vol. 2139. LNCS. Springer, Heidelberg, Aug. 2001, pp. 1–18. DOI: 10.1007/3-540-44647-8_1 (cit. on pp. 3, 6, 7).

[Bar+10] Boaz Barak, Iftach Haitner, Dennis Hofheinz, and Yuval Ishai. "Bounded Key-Dependent Message Security". In: *EUROCRYPT 2010*. Ed. by Henri Gilbert. Vol. 6110. LNCS. Springer, Heidelberg, 2010, pp. 423–444. DOI: 10.1007/978-3-642-13190-5_22 (cit. on p. 6).

[BB04a] Dan Boneh and Xavier Boyen. "Efficient Selective-ID Secure Identity Based Encryption Without Random Oracles". In: *EUROCRYPT 2004*. Ed. by Christian Cachin and Jan Camenisch. Vol. 3027. LNCS. Springer, Heidelberg, May 2004, pp. 223–238. DOI: 10.1007/978-3-540-24676-3_14 (cit. on p. 5).

[BB04b] Dan Boneh and Xavier Boyen. "Secure Identity Based Encryption Without Random Oracles". In: *CRYPTO 2004*. Ed. by Matthew Franklin. Vol. 3152. LNCS. Springer, Heidelberg, Aug. 2004, pp. 443–459. DOI: 10.1007/978-3-540-28628-8_27 (cit. on p. 5).

[BCP14] Elette Boyle, Kai-Min Chung, and Rafael Pass. "On Extractability Obfuscation". In: *TCC 2014*. Ed. by Yehuda Lindell. Vol. 8349. LNCS. Springer, Heidelberg, Feb. 2014, pp. 52–73. DOI: 10.1007/978-3-642-54242-8_3 (cit. on p. 3).

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. "Non-Interactive Zero-Knowledge and Its Applications (Extended Abstract)". In: *20th ACM STOC*. ACM Press, May 1988, pp. 103–112. DOI: 10.1145/62212.62222 (cit. on p. 10).

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. "Functional Signatures and Pseudorandom Functions". In: *PKC 2014*. Ed. by Hugo Krawczyk. Vol. 8383. LNCS. Springer, Heidelberg, Mar. 2014, pp. 501–519. DOI: 10.1007/978-3-642-54631-0_29 (cit. on pp. 8, 9).

[BGI15] Elette Boyle, Niv Gilboa, and Yuval Ishai. "Function Secret Sharing". In: *EUROCRYPT 2015, Part II*. Ed. by Elisabeth Oswald and Marc Fischlin. Vol. 9057. LNCS. Springer, Heidelberg, Apr. 2015, pp. 337–367. DOI: 10.1007/978-3-662-46803-6_12 (cit. on p. 6).

[Bit+14] Nir Bitansky, Ran Canetti, Yael Tauman Kalai, and Omer Paneth. "On Virtual Grey Box Obfuscation for General Circuits". In: *CRYPTO 2014, Part II*. Ed. by Juan A. Garay and Rosario Gennaro. Vol. 8617. LNCS. Springer, Heidelberg, Aug. 2014, pp. 108–125. DOI: 10.1007/978-3-662-44381-1_7 (cit. on p. 3).

[Bit17] Nir Bitansky. "Verifiable Random Functions from Non-interactive Witness-Indistinguishable Proofs". In: *TCC 2017, Part II*. Ed. by Yael Kalai and Leonid Reyzin. Vol. 10678. LNCS. Springer, Heidelberg, Nov. 2017, pp. 567–594. DOI: 10.1007/978-3-319-70503-3_19 (cit. on p. 3).

[BL18] Fabrice Benhamouda and Huijia Lin. "k-Round Multiparty Computation from k-Round Oblivious Transfer via Garbled Interactive Circuits". In: *EUROCRYPT 2018, Part II*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Vol. 10821. LNCS. Springer, Heidelberg, 2018, pp. 500–532. DOI: 10.1007/978-3-319-78375-8_17 (cit. on p. 3).

[Bon+08] Dan Boneh, Shai Halevi, Michael Hamburg, and Rafail Ostrovsky. "Circular-Secure Encryption from Decision Diffie-Hellman". In: *CRYPTO 2008*. Ed. by David Wagner. Vol. 5157. LNCS. Springer, Heidelberg, Aug. 2008, pp. 108–125. DOI: 10.1007/978-3-540-85174-5_7 (cit. on pp. 6, 23).

[BP13] Nir Bitansky and Omer Paneth. "On the impossibility of approximate obfuscation and applications to resettable cryptography". In: *45th ACM STOC*. Ed. by Dan Boneh, Tim Roughgarden, and Joan Feigenbaum. ACM Press, June 2013, pp. 241–250. DOI: 10.1145/2488608.2488639 (cit. on p. 3).

[BP15] Nir Bitansky and Omer Paneth. "ZAPs and Non-Interactive Witness Indistinguishability from Indistinguishability Obfuscation". In: *TCC 2015, Part II*. Ed. by Yevgeniy Dodis and Jesper Buus Nielsen. Vol. 9015. LNCS. Springer, Heidelberg, Mar. 2015, pp. 401–427. DOI: 10.1007/978-3-662-46497-7_16 (cit. on p. 11).

[BRS03] John Black, Phillip Rogaway, and Thomas Shrimpton. "Encryption-Scheme Security in the Presence of Key-Dependent Messages". In: *SAC 2002*. Ed. by Kaisa Nyberg and Howard M. Heys. Vol. 2595. LNCS. Springer, Heidelberg, Aug. 2003, pp. 62–75. DOI: 10.1007/3-540-36492-7_6 (cit. on p. 6).

[BSW16] Mihir Bellare, Igors Stepanovs, and Brent Waters. "New Negative Results on Differing-Inputs Obfuscation". In: *EUROCRYPT 2016, Part II*. Ed. by Marc Fischlin and Jean-Sébastien Coron. Vol. 9666. LNCS. Springer, Heidelberg, May 2016, pp. 792–821. DOI: 10.1007/978-3-662-49896-5_28 (cit. on pp. 6, 8).

[BV15] Nir Bitansky and Vinod Vaikuntanathan. "Indistinguishability Obfuscation from Functional Encryption". In: *56th FOCS*. Ed. by Venkatesan Guruswami. IEEE Computer Society Press, Oct. 2015, pp. 171–190. DOI: 10.1109/FOCS.2015.20 (cit. on p. 3).

[BW13] Dan Boneh and Brent Waters. "Constrained Pseudorandom Functions and Their Applications". In: *ASIACRYPT 2013, Part II*. Ed. by Kazue Sako and Palash Sarkar. Vol. 8270. LNCS. Springer, Heidelberg, Dec. 2013, pp. 280–300. DOI: 10.1007/978-3-642-42045-0_15 (cit. on pp. 8, 9).

[Can+15] Ran Canetti, Huijia Lin, Stefano Tessaro, and Vinod Vaikuntanathan. "Obfuscation of Probabilistic Circuits and Applications". In: *TCC 2015, Part II*. Ed. by Yevgeniy Dodis and Jesper Buus

Nielsen. Vol. 9015. LNCS. Springer, Heidelberg, Mar. 2015, pp. 468–497. DOI: 10.1007/978-3-662-46497-7_19 (cit. on pp. 3–8, 13, 14, 17, 19, 20, 22–25).

[Can+17] Ran Canetti, Srinivasan Raghuraman, Silas Richelson, and Vinod Vaikuntanathan. "Chosen-Ciphertext Secure Fully Homomorphic Encryption". In: *PKC 2017, Part II*. Ed. by Serge Fehr. Vol. 10175. LNCS. Springer, Heidelberg, Mar. 2017, pp. 213–240. DOI: 10.1007/978-3-662-54388-7_8 (cit. on p. 6).

[Can+18] Ran Canetti, Yilei Chen, Leonid Reyzin, and Ron D. Rothblum. "Fiat-Shamir and Correlation Intractability from Strong KDM-Secure Encryption". In: *EUROCRYPT 2018, Part I*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Vol. 10820. LNCS. Springer, Heidelberg, 2018, pp. 91–122. DOI: 10.1007/978-3-319-78381-9_4 (cit. on p. 3).

[Can97] Ran Canetti. "Towards Realizing Random Oracles: Hash Functions That Hide All Partial Information". In: *CRYPTO'97*. Ed. by Burton S. Kaliski Jr. Vol. 1294. LNCS. Springer, Heidelberg, Aug. 1997, pp. 455–469. DOI: 10.1007/BFb0052255 (cit. on p. 3).

[DN18] Nico Döttling and Ryo Nishimaki. *Universal Proxy Re-Encryption*. Cryptology ePrint Archive, Report 2018/840. https://eprint.iacr.org/2018/840. 2018 (cit. on p. 7).

[Dod+16] Yevgeniy Dodis, Shai Halevi, Ron D. Rothblum, and Daniel Wichs. "Spooky Encryption and Its Applications". In: *CRYPTO 2016, Part III*. Ed. by Matthew Robshaw and Jonathan Katz. Vol. 9816. LNCS. Springer, Heidelberg, Aug. 2016, pp. 93–122. DOI: 10.1007/978-3-662-53015-3_4 (cit. on pp. 3, 4, 6, 7, 9, 23–26, 28–30).

[Far+18] Pooya Farshim, Julia Hesse, Dennis Hofheinz, and Enrique Larraia. "Graded Encoding Schemes from Obfuscation". In: *PKC 2018, Part II*. Ed. by Michel Abdalla and Ricardo Dahab. Vol. 10770. LNCS. Springer, Heidelberg, Mar. 2018, pp. 371–400. DOI: 10.1007/978-3-319-76581-5_13 (cit. on p. 3).

[Gar+13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. "Candidate Indistinguishability Obfuscation and Functional Encryption for all Circuits". In: *54th FOCS*. IEEE Computer Society Press, Oct. 2013, pp. 40–49. DOI: 10.1109/FOCS.2013.13 (cit. on p. 3).

[Gar+14a] Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. "Two-Round Secure MPC from Indistinguishability Obfuscation". In: *TCC 2014*. Ed. by Yehuda Lindell. Vol. 8349. LNCS. Springer, Heidelberg, Feb. 2014, pp. 74–94. DOI: 10.1007/978-3-642-54242-8_4 (cit. on p. 3).

[Gar+14b] Sanjam Garg, Craig Gentry, Shai Halevi, and Daniel Wichs. "On the Implausibility of Differing-Inputs Obfuscation and Extractable Witness Encryption with Auxiliary Input". In: *CRYPTO 2014, Part I*. Ed. by Juan A. Garay and Rosario Gennaro. Vol. 8616. LNCS. Springer, Heidelberg, Aug. 2014, pp. 518–535. DOI: 10.1007/978-3-662-44371-2_29 (cit. on pp. 6, 8).

[Gar+17] Sanjam Garg, Omkant Pandey, Akshayaram Srinivasan, and Mark Zhandry. "Breaking the Sub-Exponential Barrier in Obfustopia". In: *EUROCRYPT 2017, Part III*. Ed. by Jean-Sébastien Coron and Jesper Buus Nielsen. Vol. 10212. LNCS. Springer, Heidelberg, 2017, pp. 156–181. DOI: 10.1007/978-3-319-56617-7_6 (cit. on pp. 3, 5).

[Gen09] Craig Gentry. "A fully homomorphic encryption scheme". PhD thesis. Stanford University, 2009 (cit. on p. 19).

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. "How to Construct Random Functions (Extended Abstract)". In: *25th FOCS*. IEEE Computer Society Press, Oct. 1984, pp. 464–479. DOI: 10.1109/SFCS.1984.715949 (cit. on pp. 8, 14).

[GK05] Shafi Goldwasser and Yael Tauman Kalai. "On the Impossibility of Obfuscation with Auxiliary Input". In: *46th FOCS*. IEEE Computer Society Press, Oct. 2005, pp. 553–562. DOI: 10.1109/SFCS.2005.60 (cit. on p. 3).

[GOS06] Jens Groth, Rafail Ostrovsky, and Amit Sahai. "Perfect Non-interactive Zero Knowledge for NP". In: *EUROCRYPT 2006*. Ed. by Serge Vaudenay. Vol. 4004. LNCS. Springer, Heidelberg, 2006, pp. 339–358. DOI: 10.1007/11761679_21 (cit. on p. 10).

[Goy+17] Rishab Goyal, Susan Hohenberger, Venkata Koppula, and Brent Waters. "A Generic Approach to Constructing and Proving Verifiable Random Functions". In: *TCC 2017, Part II*. Ed. by Yael Kalai and Leonid Reyzin. Vol. 10678. LNCS. Springer, Heidelberg, Nov. 2017, pp. 537–566. DOI: 10.1007/978-3-319-70503-3_18 (cit. on p. 3).

[GPS16] Sanjam Garg, Omkant Pandey, and Akshayaram Srinivasan. "Revisiting the Cryptographic Hardness of Finding a Nash Equilibrium". In: *CRYPTO 2016, Part II*. Ed. by Matthew Robshaw and Jonathan Katz. Vol. 9815. LNCS. Springer, Heidelberg, Aug. 2016, pp. 579–604. DOI: 10.1007/978-3-662-53008-5_20 (cit. on p. 3).

[GR07] Shafi Goldwasser and Guy N. Rothblum. "On Best-Possible Obfuscation". In: *TCC 2007*. Ed. by Salil P. Vadhan. Vol. 4392. LNCS. Springer, Heidelberg, Feb. 2007, pp. 194–213. DOI: 10.1007/978-3-540-70936-7_11 (cit. on p. 3).

[GS16] Sanjam Garg and Akshayaram Srinivasan. "Single-Key to Multi-Key Functional Encryption with Polynomial Loss". In: *TCC 2016-B, Part II*. Ed. by Martin Hirt and Adam D. Smith. Vol. 9986. LNCS. Springer, Heidelberg, 2016, pp. 419–442. DOI: 10.1007/978-3-662-53644-5_16 (cit. on p. 3).

[GS18] Sanjam Garg and Akshayaram Srinivasan. "Two-Round Multiparty Secure Computation from Minimal Assumptions". In: *EUROCRYPT 2018, Part II*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Vol. 10821. LNCS. Springer, Heidelberg, 2018, pp. 468–499. DOI: 10.1007/978-3-319-78375-8_16 (cit. on p. 3).

[Had00] Satoshi Hada. "Zero-Knowledge and Code Obfuscation". In: *ASIACRYPT 2000*. Ed. by Tatsuaki Okamoto. Vol. 1976. LNCS. Springer, Heidelberg, Dec. 2000, pp. 443–457. DOI: 10.1007/3-540-44448-3_34 (cit. on p. 3).

[Hås+99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. "A Pseudorandom Generator from any One-way Function". In: *SIAM Journal on Computing* 28.4 (1999), pp. 1364–1396 (cit. on pp. 8, 14).

[HK08] Dennis Hofheinz and Eike Kiltz. "Programmable Hash Functions and Their Applications". In: *CRYPTO 2008*. Ed. by David Wagner. Vol. 5157. LNCS. Springer, Heidelberg, Aug. 2008, pp. 21–38. DOI: 10.1007/978-3-540-85174-5_2 (cit. on p. 5).

[HMS07] Dennis Hofheinz, John Malone-Lee, and Martijn Stam. "Obfuscation for Cryptographic Purposes". In: *TCC 2007*. Ed. by Salil P. Vadhan. Vol. 4392. LNCS. Springer, Heidelberg, Feb. 2007, pp. 214–232. DOI: 10.1007/978-3-540-70936-7_12 (cit. on p. 3).

[Hoh+07] Susan Hohenberger, Guy N. Rothblum, abhi shelat, and Vinod Vaikuntanathan. "Securely Obfuscating Re-encryption". In: *TCC 2007*. Ed. by Salil P. Vadhan. Vol. 4392. LNCS. Springer, Heidelberg, Feb. 2007, pp. 233–252. DOI: 10.1007/978-3-540-70936-7_13 (cit. on p. 3).

[HRW16] Dennis Hofheinz, Vanishree Rao, and Daniel Wichs. "Standard Security Does Not Imply Indistinguishability Under Selective Opening". In: *TCC 2016-B, Part II*. Ed. by Martin Hirt and Adam D. Smith. Vol. 9986. LNCS. Springer, Heidelberg, 2016, pp. 121–145. DOI: 10.1007/978-3-662-53644-5_5 (cit. on p. 3).

[HSW14] Susan Hohenberger, Amit Sahai, and Brent Waters. "Replacing a Random Oracle: Full Domain Hash from Indistinguishability Obfuscation". In: *EUROCRYPT 2014*. Ed. by Phong Q. Nguyen and Elisabeth Oswald. Vol. 8441. LNCS. Springer, Heidelberg, May 2014, pp. 201–220. DOI: 10.1007/978-3-642-55220-5_12 (cit. on p. 5).

[HW09] Susan Hohenberger and Brent Waters. "Short and Stateless Signatures from the RSA Assumption". In: *CRYPTO 2009*. Ed. by Shai Halevi. Vol. 5677. LNCS. Springer, Heidelberg, Aug. 2009, pp. 654–670. DOI: 10.1007/978-3-642-03356-8_38 (cit. on p. 5).

[IPS15] Yuval Ishai, Omkant Pandey, and Amit Sahai. "Public-Coin Differing-Inputs Obfuscation and Its Applications". In: *TCC 2015, Part II*. Ed. by Yevgeniy Dodis and Jesper Buus Nielsen. Vol. 9015. LNCS. Springer, Heidelberg, Mar. 2015, pp. 668–697. DOI: 10.1007/978-3-662-46497-7_26 (cit. on p. 3).

[Kia+13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. "Delegatable pseudorandom functions and applications". In: *ACM CCS 2013*. Ed. by Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung. ACM Press, Nov. 2013, pp. 669–684. DOI: 10.1145/2508859.2516668 (cit. on pp. 8, 9).

[Lin17] Huijia Lin. "Indistinguishability Obfuscation from SXDH on 5-Linear Maps and Locality-5 PRGs". In: *CRYPTO 2017, Part I*. Ed. by Jonathan Katz and Hovav Shacham. Vol. 10401. LNCS. Springer, Heidelberg, Aug. 2017, pp. 599–629. DOI: 10.1007/978-3-319-63688-7_20 (cit. on p. 3).

[LM16] Baiyu Li and Daniele Micciancio. "Compactness vs Collusion Resistance in Functional Encryption". In: *TCC 2016-B, Part II*. Ed. by Martin Hirt and Adam D. Smith. Vol. 9986. LNCS. Springer, Heidelberg, 2016, pp. 443–468. DOI: 10.1007/978-3-662-53644-5_17 (cit. on p. 3).

[LPS04] Ben Lynn, Manoj Prabhakaran, and Amit Sahai. "Positive Results and Techniques for Obfuscation". In: *EUROCRYPT 2004*. Ed. by Christian Cachin and Jan Camenisch. Vol. 3027. LNCS. Springer, Heidelberg, May 2004, pp. 20–39. DOI: 10.1007/978-3-540-24676-3_2 (cit. on p. 3).

[LT17] Huijia Lin and Stefano Tessaro. "Indistinguishability Obfuscation from Trilinear Maps and Block-Wise Local PRGs". In: *CRYPTO 2017, Part I*. Ed. by Jonathan Katz and Hovav Shacham. Vol. 10401. LNCS. Springer, Heidelberg, Aug. 2017, pp. 630–660. DOI: 10.1007/978-3-319-63688-7_21 (cit. on p. 3).

[LZ17] Qipeng Liu and Mark Zhandry. "Decomposable Obfuscation: A Framework for Building Applications of Obfuscation from Polynomial Hardness". In: *TCC 2017, Part I*. Ed. by Yael Kalai and Leonid Reyzin. Vol. 10677. LNCS. Springer, Heidelberg, Nov. 2017, pp. 138–169. DOI: 10.1007/978-3-319-70500-2_6 (cit. on pp. 3, 5).

[Ps16] Rafael Pass and abhi shelat. "Impossibility of VBB Obfuscation with Ideal Constant-Degree Graded Encodings". In: *TCC 2016-A, Part I*. Ed. by Eyal Kushilevitz and Tal Malkin. Vol. 9562. LNCS. Springer, Heidelberg, Jan. 2016, pp. 3–17. DOI: 10.1007/978-3-662-49096-9_1 (cit. on p. 3).

[PST14] Rafael Pass, Karn Seth, and Sidharth Telang. "Indistinguishability Obfuscation from Semantically-Secure Multilinear Encodings". In: *CRYPTO 2014, Part I*. Ed. by Juan A. Garay and Rosario Gennaro. Vol. 8616. LNCS. Springer, Heidelberg, Aug. 2014, pp. 500–517. DOI: 10.1007/978-3-662-44371-2_28 (cit. on p. 3).

[SW14] Amit Sahai and Brent Waters. "How to use indistinguishability obfuscation: deniable encryption, and more". In: *46th ACM STOC*. Ed. by David B. Shmoys. ACM Press, 2014, pp. 475–484. DOI: 10.1145/2591796.2591825 (cit. on pp. 3, 4).

[Wat05] Brent R. Waters. "Efficient Identity-Based Encryption Without Random Oracles". In: *EUROCRYPT 2005*. Ed. by Ronald Cramer. Vol. 3494. LNCS. Springer, Heidelberg, May 2005, pp. 114–127. DOI: 10.1007/11426639_7 (cit. on p. 5).

[Wee05] Hoeteck Wee. "On obfuscating point functions". In: *37th ACM STOC*. Ed. by Harold N. Gabow and Ronald Fagin. ACM Press, May 2005, pp. 523–532. DOI: 10.1145/1060590.1060669 (cit. on p. 3).

[Zha16] Mark Zhandry. "The Magic of ELFs". In: *CRYPTO 2016, Part I*. Ed. by Matthew Robshaw and Jonathan Katz. Vol. 9814. LNCS. Springer, Heidelberg, Aug. 2016, pp. 479–508. DOI: 10.1007/978-3-662-53018-4_18 (cit. on pp. 3–5, 9, 10).

[Zim15] Joe Zimmerman. "How to Obfuscate Programs Directly". In: *EUROCRYPT 2015, Part II*. Ed. by Elisabeth Oswald and Marc Fischlin. Vol. 9057. LNCS. Springer, Heidelberg, Apr. 2015, pp. 439–467. DOI: 10.1007/978-3-662-46803-6_15 (cit. on p. 3).