On the equivalence of obfuscation and multilinear maps

Omer Paneth* Amit Sahai†

August 5, 2015

Abstract

Garg et al. [FOCS 2013] showed how to construct indistinguishability obfuscation (iO) from a restriction of cryptographic multilinear maps called Multilinear Jigsaw Puzzles. Since then, a number of other works have shown constructions and security analyses for iO from different abstractions of multilinear maps. However, the converse question — whether some form of multilinear maps follows from iO — has remained largely open.

We offer an abstraction of multilinear maps called Polynomial Jigsaw Puzzles, and show that iO for circuits implies Polynomial Jigsaw Puzzles. This implication is unconditional: no additional assumptions, such as one-way functions, are needed. Furthermore, we show that this abstraction of Polynomial Jigsaw Puzzles is sufficient to construct iO for $\mathbf{NC}^1$, thus showing a near-equivalence of these notions.

*Boston University. Email: omer@bu.edu. Supported by the Simons award for graduate students in theoretical computer science and an NSF Algorithmic foundations grant 1218461. This work was done in part while the author was visiting the Simons Institute for the Theory of Computing, supported by the Simons Foundation and by the DIMACS/Simons Collaboration in Cryptography through NSF grant #CNS-1523467.

†UCLA. Email: sahai@cs.ucla.edu. Research supported in part from a DARPA/ONR PROCEED award, a DARPA/ARL SAFEWARE award, NSF Frontier Award 1413955, NSF grants 1228984, 1136174, 1118096, and 1065276, a Xerox Faculty Research Award, a Google Faculty Research Award, an equipment grant from Intel, and an Okawa Foundation Research Grant. This material is based upon work supported by the Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014-11-1-0389. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, or the U.S. Government. This work was done in part while the author was visiting the Simons Institute for the Theory of Computing, supported by the Simons Foundation and by the DIMACS/Simons Collaboration in Cryptography through NSF grant #CNS-1523467.

1 Introduction

Interest in program obfuscation has burgeoned since the candidate construction of indistinguishability obfuscation (iO) [BGI$^+$01] of Garg et al. [GGH$^+$13b]. This construction was built upon Multilinear Jigsaw Puzzles, a variant of multilinear maps [GGH13a]. Since then, a number of works have built indistinguishability obfuscation based on variants of multilinear maps with security argued in a generic model [BR14, BGK$^+$14, AGIS14, SZ14, Zim15, AB15]. In a closely related work, Pass et al. [PST14] construct iO from the semantic security assumption on multi-linear maps, which is also known as a type of relaxed "uber assumption" over multilinear maps, in the spirit of [BBG05]. Finally, Gentry et al. [GLSW14] give a construction of iO based on the multilinear subgroup elimination assumption for composite-order multilinear maps.

Throughout these works on iO from multilinear maps, interestingly the notion of iO has remained quite stable – with a well-accepted definition. However, these works have considered several different plausible "interfaces" for multilinear maps exposed by candidate constructions [GGH13a, CLT13, GGH15, CLT15]. Furthermore, there are a variety of useful security models and assumptions that have been considered over multilinear maps.

Given the tight relationship between current constructions of iO and multilinear maps, a major question today is understanding the relationship between these objects. In particular, the following question, which is the subject of this work, has remained open:

Is there some form of multilinear maps that are necessary for iO, and if so, what is a necessary hardness assumption over these multilinear maps?

1.1 Our contribution

In this work, we put forward a variant of multilinear maps that we call Polynomial Jigsaw Puzzles (PJPs), which is closely related to the notion of Multilinear Jigsaw Puzzles of [GGH$^+$13b]. However, unlike other notions of multilinear maps, our notion of Polynomial Jigsaw Puzzles includes a "built-in" simple and natural hardness assumption over these puzzles. We then establish the following two results:

- We give a construction of iO for $\mathbf{NC}^1$ from Polynomial Jigsaw Puzzles. (From [GGH$^+$13b], we know that iO for $\mathbf{NC}^1$ and LWE imply iO for circuits.)
- We show that iO for circuits implies Polynomial Jigsaw Puzzles. This implication is unconditional, in that it does not require any additional assumptions like one-way functions.

Thus, we have in PJPs a natural notion of multilinear maps that is (almost) equivalent to iO for circuits. This shows that a natural form of multilinear maps is inherent in iO.

Multilinear Jigsaw Puzzles. As the name suggests, PJPs are closely related to the notion of Multilinear Jigsaw Puzzles of [GGH$^+$13b]. At a high level, the abstract interface of a Multilinear Jigsaw Puzzle is as follows: The puzzle generator specifies a sequence of secret ring elements $(X_1, \ldots, X_m)$ that it wants to encode into a puzzle Z. It also specifies some public predicate V that defines the format of valid solutions. A correct solution to the puzzle is a polynomial P (represented as an arithmetic circuit) that has the correct format and vanishes on the secret ring elements

$$V(P) = 1 \quad \wedge \quad P(X_1, \ldots, X_m) = 0 \ .$$

Given Z, the puzzle solver can publicly test if a given solution is correct or not. In particular, in the case of Multilinear Jigsaw Puzzles, the constraint that $V(P) = 1$ is used to guarantee that the polynomial P is multilinear (and the same is true for all polynomials computed by subcircuits of P). Thus, $V(P) = 1$ is essentially just capturing a "formatting requirement" on the arithmetic circuit computing the putative solution polynomial P.

For Multilinear Jigsaw Puzzles as proffered by [GGH+13b], the underlying ring was imagined to be $\mathbb{Z}_p$. In other forms of multilinear maps, such as those of [CLT13, CLT15], the ring is imagined to be $\mathbb{Z}_N$ for a composite N that is a product of several large random primes.

Polynomial Jigsaw Puzzles. Polynomial Jigsaw Puzzles are exactly the same as Multilinear Jigsaw Puzzles, except that the underlying ring is the ring of multivariate polynomials with integer coefficients, namely $\mathbb{Z}[Y_1, \ldots, Y_\ell]$. In other words, every secret ring element X_i is itself a polynomial over the variables $Y_1, \ldots, Y_\ell$, and a correct solution P must ensure that the composed polynomial $P(X_1, \ldots, X_m)$ is equivalent to the zero polynomial in $\mathbb{Z}[Y_1, \ldots, Y_\ell]$. Furthermore, our formatting requirement $V(P) = 1$ will also include a fixed polynomial size bound on the size of the arithmetic circuit computing P.

One may wonder about this choice of ring: why is it natural to encode polynomials themselves, rather than elements from a more familiar ring like $\mathbb{Z}_p$? Thinking about natural applications of multilinear maps, we see that we rarely care about encoding any specific element in $\mathbb{Z}_p$ – what we really care about is the polynomial that computes that element. Indeed, our construction of iO for $\mathbf{NC}^1$ from PJPs shows that working with the ring of polynomials does not incur a loss of usefulness.

Let us keep as a running example the DDH assumption, in the completely linear case where no multiplications are allowed. Seen from our perspective, there is a variant of DDH that we can call the polynomial-DDH assumption stated as follows: Can an adversary distinguish a puzzle encoding the polynomials $(Y_1, Y_2, Y_1 \cdot Y_2)$ from a puzzle encoding the polynomials (Y_1, Y_2, Y_3)?

Security: the hardness assumption for Polynomial Jigsaw Puzzles. Very succinctly, we can state the security requirement of PJPs as follows: Let Z and Z' be two puzzles sampled by the puzzle generator, corresponding to two secret vectors of ring elements $(X_1, \ldots, X_m)$ and $(X'_1, \ldots, X'_m)$. We require that Z and Z' are computationally indistinguishable iff they have the same set of solutions. In other words, Z and Z' are indistinguishable if the following holds: for any validly formatted polynomial P, we have that $P(X_1, \ldots, X_m) \equiv \mathbf{0}$ iff $P(X'_1, \ldots, X'_m) \equiv \mathbf{0}$.

We can now see, for example, why our security property for PJPs would imply polynomial-DDH in the completely linear case. Recall that in the completely linear case, the formatting requirement $V(P) = 1$ would ensure that only completely linear polynomials P can be submitted by the adversary. Now, for DDH, we would have $(X_1, X_2, X_3) = (Y_1, Y_2, Y_1Y_2)$ and $(X'_1, X'_2, X'_3) = (Y_1, Y_2, Y_3)$. It is trivial to see, however, that the only linear polynomial P that would yield the zero polynomial when given the input (X_1, X_2, X_3) is $P \equiv \mathbf{0}$, and the same is true for (X'_1, X'_2, X'_3). Thus, the puzzle encoding (X_1, X_2, X_3) and the puzzle encoding (X'_1, X'_2, X'_3) would have identical solutions, and therefore, our security requirement would imply that polynomial-DDH would hold.

On distributional vs. static assumptions. An interesting feature of our formulation is the following: Many assumptions, like DDH, normally require a *distributional* formulation – where the challenger chooses a, b, and c at random, and releases (g^a, g^b) together with either g^{ab} or g^c. In such a distributional formulation, it is crucial that the challenger's choices of random a, b, c are kept secret from the adversary. However, in polynomial-DDH as sketched above, the challenger is *static* – it is not bound to any distribution. Instead, it releases encodings of explicit fixed polynomials (Y_1, Y_2), and either $Y_1 \cdot Y_2$ or Y_3, where Y_1, Y_2, and Y_3 are formal variables, not randomly generated values. These polynomials are completely public, and known to the adversary.

On "uber assumptions." The fact that we do not require a distributional formulation of our security property may seem to be a curiosity. In fact, in the context of "uber assumptions," this fact offers surprising benefits.

As introduced by [BBG05], an uber assumption is an assumption that quantifies over a large class of natural and unnatural assumptions. For example, an uber assumption could imply a natural assumption $\mathcal{X}$ not because of a careful reduction, but just because the challenge posed by Assumption $\mathcal{X}$ falls within the quantification of the uber assumption. It is clear that our security property for PJPs is such an uber assumption; as we have seen, it already implies our polynomial-DDH as a special case.

However, our security property is a *static* uber assumption: The quantification in our case is over the polynomials $(X_1, \ldots X_m)$ and $(X'_1, \ldots, X'_m)$. Our security property does not quantify over distributions.

In the context of multilinear maps, this marks a significant departure over previous uber assumptions, as considered by [PST14, BCKP14]. In these works, several uber assumptions over multilinear maps, seen as Multilinear Jigsaw Puzzles, were given, where the quantification of the uber assumption was also over *distributions* of values to be encoded in these puzzles.

Uber assumptions that quantify over distributions of values to be encoded can be very tricky to reason about. As observed by [PST14], the most natural distributional uber assumption for multilinear maps would actually imply that VBB obfuscation exists for circuits, something that we know to be false if one-way functions exist [BGI$^+$01]. Other natural variants of distributional uber assumptions were shown to imply VGB obfuscation, and even to be necessary for VGB obfuscation [BCKP14].

Thus, our use of a static security definition, one that does not quantify over distributions, distinguishes us from previous uber assumptions over multilinear maps. Indeed, prior to our work, it was an open question to formulate a natural uber assumption over multilinear maps that would only imply iO but not VGB obfuscation. Since we show that our security notion is implied by iO for circuits, we know that it cannot yield VGB obfuscation unless iO itself yields VGB obfuscation.

Discussion on the implications of our results. It may be tempting to conclude from our results that constructions of iO *must* be based on candidates for multilinear maps. But of course, this is fallacious: We know that one-way functions and digital signatures are equivalent, but clearly one need not construct a digital signature scheme to construct a one-way function. However, equally obviously, our result does mean that if a new method for constructing iO without multilinear maps is found, this method would also necessarily yield multilinear maps in the form of PJPs.

Variations. There are many natural variants of the notion of PJPs. For example, we can modify the set of possible formatting predicates for PJPs. Specifically, we also consider the notion of a *generalized* Polynomial Jigsaw Puzzle, where the predicate V can be given by an arbitrary circuit, and does not necessarily check that P is multilinear. This notion is also implied by iO for circuits.

1.2 Our Techniques

We give an overview of our construction and their proof of security.

1.2.1 Polynomial Jigsaw Puzzles from iO for circuits

We show how to construct a puzzle Z encoding a sequence of polynomials $X_1, \ldots, X_m$ and a formatting predicate V. In our construction, the puzzle Z is just an obfuscation of circuit C that has the polynomials $X_1, \ldots, X_m$ and the predicate V hard coded into it. This circuit C takes as input a solution polynomial, described as an arithmetic circuit of bounded sizes, and verifies it. To verify the puzzle Z, we simply evaluate it on the proposed solution.

The circuit C is implemented as follows. Recall that a solution P is a valid iff $V(P) = 1$ and $P(X_1, \ldots, X_m) \equiv 0$. Since V is public, verifying that $V(P) = 1$ is straightforward. To verify that $P(X_1, \ldots, X_m) \equiv 0$, C first chooses a field F that is much larger then the total degree of the composed polynomial $P(X_1, \ldots, X_m)$. C then evaluates evaluates the composed polynomial $P(X_1, \ldots, X_m)$ on many sets of random inputs from the field F. In case $P(X_1, \ldots, X_m) \equiv 0$, the outcome of all these evaluation must be zero. However, in case $P(X_1, \ldots, X_m) \not\equiv 0$, by the Schwartz-Zippel lemma, most evaluations will result in a non-zero outcome. Since the solution circuit is bounded in size, we can bound the number of possible solutions. Based on this bound, we set the number of random evaluation points that are hard-coded into C such that, with overwhelming probability, no invalid solution is accepted.

In the security proof we consider a pair of puzzles (Z_1, Z_2) that have the same set of solutions and we want to show that they are indistinguishable. By the correctness property of our construction, these puzzles must be obfuscating a pair of functionally equivalent verification circuit and therefore, by the security of indistinguishability obfuscation, the puzzles are indistinguishable.

Note that our construction crucially relies on the fact that all solution must be of bounded size. This fact is used as follows:

1. Since our puzzle is a *circuit* that takes as input a solution and verifies it, we must have a bound on the size of possible solutions at the time the puzzle is generated.

2. The bound on the solution size allows to bound the degree of solution polynomial, and therefore, also the degree of the composed polynomial $P(X_1, \ldots, X_m)$. This bound is required for selecting a large enough field F.

3. The bound on the solution size is also used to bound the number of possible solutions, and thus, also the number of random evaluation points required to guarantee correctness.

1.2.2 iO from Polynomial Jigsaw Puzzles

Our construction of iO from Polynomial Jigsaw Puzzles is based on a sequence of works constructing obfuscation based on Multilinear Jigsaw Puzzles in generic models [GGH$^+$13b, BR14, BGK$^+$14]. Specifically our construction closely follows that of Barak et al. [BGK$^+$14]. We first describe the main components in construction of [BGK$^+$14], and then explain how these are used in our setting.

The [BGK$^+$14] construction. Roughly, the construction in [BGK$^+$14] follows the steps below.

1. The input circuit is first translated into a branching program that consists of n pairs of matrices. To evaluate the program on a given input, we select one matrix from every pair based on the input bits. The product of these selected matrices encodes the program's output.

2. The branching program matrices are randomized following Kilian's randomization technique [Kil88]. After this step we have that in every program evaluation, the selected matrices look like completely random matrices (over some large field) conditioned on the product of these matrices encoding the correct output.

3. The program is farther randomize by multiplying every matrix in every level by an independent random field element. The purpose of this randomization is to break any algebraic dependencies between the matrices participating in different program evaluations.

4. The matrices are encoded, entry by entry, in a Multilinear Jigsaw Puzzle. The formatting predicate V is define using a straddling set system, in a way that forces any evaluator to select matrices that are consistent with a single program evaluation.

In the above description we skipped the "bookends" and the notion of dual-input branching programs used in [BGK$^+$14]. These tools are needed in proof of the generic VBB security and are not used in our settings.

Our construction. The main observation that allows translating the techniques of [BGK$^+$14] to the setting of Polynomial Jigsaw Puzzles is that in the [BGK$^+$14] construction, every matrix entry in the randomize branching program is computed as a polynomial evaluated over random field elements (sampled during the randomization steps). The coefficients of this polynomial depend on the entries of the original branching program. Additionally, the evaluation of the program on a given input also takes the form of a polynomial evaluated over the entries of the branching program matrices. The coefficients of this polynomial depend on the input to the program.

In our construction of iO, the obfuscated program is just a Polynomial Jigsaw Puzzle encoding the set of polynomials that compute the entries of the randomize branching program. To evaluate the obfuscated program we try to solve the puzzle with the polynomial describing the program evaluation on the desired input. We show that the program accepts the input iff the solution is valid.

In the proof of security we show that any pair of puzzles encoding functionally equivalent branching programs have exactly the same set of solutions. The proof follows closely the argument in [BGK$^+$14], except that is argues about formal polynomial instead of the evaluations of these polynomials over random inputs.

2 Preliminaries

2.1 Arithmetic Circuits

In this work we consider arithmetic circuits that may contain addition, subtraction and multiplication gates, and the constants $\{0, 1\}$. Such arithmetic circuits can be evaluated over arbitrary rings. We often identify an arithmetic circuit with a formal polynomial that it computes over $\mathbb{Z}$. For example, we define the degree of the circuit as the degree of the polynomial it computes, and we say that two arithmetic circuits are equivalent if the polynomials they compute are equivalent over $\mathbb{Z}$. We also use the following fact stating that an arithmetic circuit of size s computes a polynomial with coefficients and total degree at most 2^s. This follows from a trivial induction.

Fact 2.1. Let C be an arithmetic circuit of size s and let P be the polynomial computed by C. Let F be a finite field field of characteristic larger than 2^s. We have that:

- The total degree of P is at most 2^s.
- $P \equiv 0$ over $\mathbb{Z}$ iff $P \equiv 0$ over F.

2.2 Indistinguishability Obfuscation

We define the notion of indistinguishability obfuscation for an arbitrary class of circuit. We also explicitly define indistinguishability obfuscation for low-depth circuits ($\mathbf{NC}^1$) and for polynomial-size circuits ($\mathbf{P}/poly$). Finally we state a bootstrapping theorem from indistinguishability obfuscation for low-depth circuit to indistinguishability obfuscation for all polynomial-size circuits.

Definition 2.1 (Indistinguishability obfuscation [BGI$^+$01])**.** *An PPT algorithm $i\mathcal{O}$ is an indistinguishability obfuscator for an ensemble of circuit families $\{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$, if it satisfies the following properties:*

- *Correctness: There exists a negligible function μ such that such that for all $\lambda \in \mathbb{N}$, and every circuits $C \in \mathcal{C}_\lambda$*
$$\Pr\left[\forall x : i\mathcal{O}(C)(x) = C(x)\right] \geq 1 - \mu(\lambda) \ .$$
- *Security: for any poly-size distinguisher D, there exists a negligible function μ such that for all $\lambda \in \mathbb{N}$, and any two circuits $C_1, C_2 \in \mathcal{C}_\lambda$ of the same size and functionality,*
$$|\Pr[D(i\mathcal{O}(C_1)) = 1] - \Pr[D(i\mathcal{O}(C_2)) = 1]| \leq \mu(n) \ .$$

Definition 2.2 (Indistinguishability obfuscation for $\mathbf{NC}^1$)**.** *For every constant $d \in \mathbb{N}$, let $\left\{\mathcal{C}^d_\lambda\right\}_{\lambda\in\mathbb{N}}$ be the class of circuits of depth at most $d \cdot \log(\lambda)$ and size at most λ. An indistinguishability obfuscator for $\mathbf{NC}^1$ is defined by a family of obfuscators $\left\{i\mathcal{O}^d\right\}_{d\in\mathbb{N}}$ such that for every $d \in \mathbb{N}$, $i\mathcal{O}^d$ is an indistinguishability obfuscator for the class $\mathcal{C}^d_\lambda$.*

Definition 2.3 (Indistinguishability obfuscation for $\mathbf{P}/poly$)**.** *Let $\{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ be the class of circuits of size at most λ. An obfuscator $i\mathcal{O}$ is an indistinguishability obfuscator for $\mathbf{P}/poly$ if it is an indistinguishability obfuscator the class $\mathcal{C}_\lambda$.*

Theorem 2.1 (Bootstrapping for indistinguishability obfuscation [GGH+13b])**.** *Assuming fully-homomorphic encryption and indistinguishability obfuscation for $\mathbf{NC}^1$ there exists indistinguishability obfuscator for $\mathbf{P}/poly$.*

2.3 Branching Programs

Our main obfuscation construction is for polynomial size *branching programs*, which are powerful enough to simulate $\mathbf{NC}^1$ circuits.

A branching program consists of a sequence of steps, where each step is defined by a pair of permutations. In each step the the program examines one input bit, and depending on its value the program chooses one of the permutations. The program outputs 1 if and only if the multiplications of the permutations chosen in all steps is the identity permutation.

Definition 2.4 (Oblivious Matrix Branching Program)**.** *A* branching program *of width w and length n for ℓ-bit inputs is given by a permutation matrix $P_{\mathsf{rej}} \in \{0,1\}^{w\times w}$ such that $P_{\mathsf{rej}}[1,1] = 0$ and by a sequence:*

$$\mathsf{BP} = \left(\mathsf{inp}(i), B_{i,0}, B_{i,1}\right)_{i=1}^{n} \ ,$$

where each $B_{i,b}$ is a permutation matrix in $\{0,1\}^{w\times w}$, and $\mathsf{inp}(i) \in [\ell]$ is the input bit position examined in step i. The output of the branching program on input $x \in \{0,1\}^\ell$ is as follows:

$$\mathsf{BP}(x) = \begin{cases} 1 & \text{if } \prod_{i=1}^{n} B_{i,x_{\mathsf{inp}(i)}} = I_{w\times w} \\ 0 & \text{if } \prod_{i=1}^{n} B_{i,x_{\mathsf{inp}(i)}} = P_{\mathsf{rej}} \\ \perp & \text{otherwise} \end{cases}$$

The branching program is said to be oblivious *if $\mathsf{inp} : [n] \to [\ell]$ is a fixed function, independent of the function being evaluated.*

Theorem 2.2 ([Bar86])**.** *For any depth-d fan-in-2 boolean circuit C, there exists an oblivious branching program of width 5 and length at most 4^d that computes the same function as the circuit C.*

2.4 Straddling Set System

One ingredient in obfuscation construction is a straddling set system satisfying certain combinatorial properties.

Definition 2.5 ([BGK+14])**.** *A* straddling set system *with n entries is a collection of sets:*

$$\mathbb{S}_n = \{S_{i,b}, : i \in [n], b \in \{0,1\}\} \ ,$$

over a universe U, such that:

$$\cup_{i\in[n]} S_{i,0} = \cup_{i\in[n]} S_{i,1} = U \ .$$

and for every distinct non-empty sets $A, B \subseteq \mathbb{S}_n$ we have that if:

1. *(Disjoint Sets:) A contains only disjoint sets. B contains only disjoint sets.*
2. *(Collision:) $\cup_{S\in A} S = \cup_{S\in B} S$*

Then, it must be that $\exists\, b \in \{0,1\}$*:*

$$A = \{S_{j,b}\}_{j\in[n]} \ , \ B = \{S_{j,(1-b)}\}_{j\in[n]} \ .$$

Therefore, in a straddling set system, the only exact covers of the universe U *are* $\{S_{j,0}\}_{j\in[n]}$ *and* $\{S_{j,1}\}_{j\in[n]}$*.*

Fact 2.2 ([BGK+14]). There exists a straddling set system with n entries over a universe of size $2n-1$

3 Polynomial Jigsaw Puzzles

In this section we define the notion of Polynomial Jigsaw Puzzles. A central component in the definition of Polynomial Jigsaw Puzzles are structured arithmetic circuit that we call set respecting.

Definition 3.1 (Set-Respecting Arithmetic Circuits)**.** *Given a universe set* U*, and a vector of sets* $\vec{S} \in (2^U)^m$*, we say that an arithmetic circuit* C *taking* m *input bits is* $\vec{S}$*-respecting if there exists a function* Tag *from the wires of* C *to* 2^U *such that the following holds:*

- *For every* $i \in [m]$*, the* i*-th input wire* w^i_{in} *satisfies* $\mathsf{Tag}(w^i_{in}) = \vec{S}[i]$*.*
- *Every* $+$ *or* $-$ *gate in* C *connecting input wires* u *and* v *to an output wire* w*, satisfies:*

$$\mathsf{Tag}(u) = \mathsf{Tag}(v) = \mathsf{Tag}(w) \ .$$

- *Every* $\times$ *gate in* C *connecting input wires* u *and* v *to an output wire* w*, satisfies*

$$\mathsf{Tag}(u) \cap \mathsf{Tag}(v) = \emptyset \quad \wedge \quad \mathsf{Tag}(u) \cup \mathsf{Tag}(v) = \mathsf{Tag}(w) \ .$$

- *The output wire* w_{out} *satisfies* $\mathsf{Tag}(w_{out}) = U$*.*

Definition 3.2 (Polynomial jigsaw puzzle)**.** *A polynomial jigsaw puzzle scheme consists of a pair of PPT algorithms* $(\mathsf{Gen}, \mathsf{Ver})$ *satisfying the following properties:*

Syntax *The puzzle generation algorithm* Gen *is given as input:*

- *A security parameter* 1^λ*.*
- *A universe set* U*.*
- *A vector* $\vec{S} \in (2^U)^m$ *of* m *sets such that no entry of* $\vec{S}$ *is empty.*
- *A vector* $\vec{F}$ *of* m *arithmetic circuits of over the same set of input variables* X*.*

Gen *outputs a puzzle* Z*.*

The solution verification algorithm Ver *takes a puzzle* Z *and a description of an arithmetic circuit* G *taking* m *input variables.* Ver *outputs a bit.*

Correctness *For every security parameter* 1^λ*, every universe set* U*, every vector* $\vec{S}$ *of sets, and every vector* $\vec{F}$ *of arithmetic circuits*

$$\vec{F} = (F_1, \ldots F_m) \ ,$$

following the syntax above, let

$$Z = \mathsf{Gen}(1^\lambda, U, \vec{S}, \vec{F}) \ .$$

For every arithmetic circuit G *taking* m *input variables let* $G[\vec{F}]$ *be the composed circuit*

$$G[\vec{F}](X) = G(F_1(X), \ldots, F_m(X)) \ .$$

We require that with overwhelming probability over the choice of Z the following holds: for every arithmetic circuit G of size at most $s(\lambda)$

$$\mathsf{Ver}(Z, G) = 1 \quad \Leftrightarrow \quad \Big(G \text{ is } \vec{S}\text{-respecting} \quad \wedge \quad G[\vec{F}] \equiv 0\Big) \ .$$

Security. *Let:*

$$\left\{U_\lambda, \vec{S}_\lambda, \left(\vec{F}^1_\lambda, \vec{F}^2_\lambda\right)\right\}_{\lambda\in\mathbb{N}} \ ,$$

be a polynomial size ensemble such that:

1. *For every $i \in [m]$, $\vec{F}^1_\lambda[i]$ and $\vec{F}^2_\lambda[i]$ are arithmetic circuits of the same size and total degree.*
2. *For every $\lambda \in \mathbb{N}$ and every $\vec{S}_\lambda$-respecting arithmetic circuit G.*

$$G[\vec{F}^1_\lambda] \equiv 0 \quad \Leftrightarrow \quad G[\vec{F}^2_\lambda] \equiv 0 \ .$$

We require that:

$$\left\{\mathsf{Gen}(1^\lambda, U_\lambda, \vec{S}_\lambda, \vec{F}^1_\lambda)\right\}_{\lambda\in\mathbb{N}} \approx_c \left\{\mathsf{Gen}(1^\lambda, U_\lambda, \vec{S}_\lambda, \vec{F}^2_\lambda)\right\}_{\lambda\in\mathbb{N}} \ .$$

We say the the polynomial jigsaw puzzle scheme is s-bounded for a polynomial $s = s(\lambda)$ if the description of the solution circuit G is required to be of size at most s (both in the correctness and in the security properties).

4 $i\mathcal{O}$ from Polynomial Jigsaw Puzzles

In this section we construct an indistinguishability obfuscator $i\mathcal{O}$ for $\mathbf{NC}^1$ circuits based on Polynomial Jigsaw Puzzles. See Section 1.2 for a high-level overview of the construction.

Given a circuit taking ℓ input bits, the obfuscator $i\mathcal{O}$ first transforms it into an oblivious matrix branching program BP of width $w = 5$ and length n:

$$\mathsf{BP} = \big(\mathsf{inp}(i), B_{i,0}, B_{i,1}\big)_{i=1}^{n} \ ,$$

Recall that each $B_{i,b}$ is a permutation matrix in $\{0,1\}^{w\times w}$, and $\mathsf{inp}(i) \in [\ell]$ is the position of the input bit inspected in step i. Without loss of generality, we assume that every bit of the input is inspected by BP exactly n' times. More precisely, for input bit $j \in [\ell]$, we denote by $\mathsf{ind}(j)$ the set of steps that inspect the j'th bit:

$$\mathsf{ind}(j) = \{i \in [n] : \mathsf{inp}(i) = j\} \ .$$

We assume that for every input bit $j \in [\ell]$, $|\mathsf{ind}(j)| = n'$.

$i\mathcal{O}$ will initialize a jigsaw puzzle Z based on the program BP. We use a polynomial jigsaw puzzle scheme $(\mathsf{Gen}, \mathsf{Ver})$ of size $s = O(n)$. Next we define the a sequence of arithmetic circuits and a corresponding sequence of sets encoded in the puzzle.

The arithmetic circuits. The arithmetic circuits are defined over the set X of input variables:

$$X = \{r_{i,u,v}\}_{i\in[n-1],u,v\in[n]} \cup \{\alpha_{i,b}\}_{i\in[n],b\in\{0,1\}} \ .$$

For $i \in [n-1]$ consider the $w \times w$ matrix of variables $R_i = (r_{i,u,v})_{u,v\in[w]}$. For simplicity of notation we also consider the matrices $R_0 = R_n = I_{w\times w}$. For every $i \in [n]$ and every $b \in \{0,1\}$ let $\tilde{B}_{i,b}$ be the matrix:

$$\tilde{B}_{i,b} = R_{i-1} \cdot B_{i,b} \cdot \mathrm{adj}(R_i) \ . \tag{1}$$

We consider the the vector $\vec{F}$ of m arithmetic circuits

$$\vec{F} = (F_{i,u,v,b})_{i\in[n],u,v\in[w],b\in\{0,1\}} \ ,$$

where $F_{i,u,v,b}$ is the circuit over the input variables X that evaluates the expression

$$\alpha_{i,b} \cdot \tilde{B}_{i,b}[u,v] \ .$$

Note that since w is a constant, the size of the circuit $F_{i,u,v,b}$ is also constant. Also note that since the elements of $B_{i,b}$ are in $\{0,1\}$, the circuit $F_{i,u,v,b}$ includes only constants in $\{0,1\}$.

The sets. Let $\mathbb{S}^1,\ldots,\mathbb{S}^\ell$ be ℓ disjoint copies of a straddling set system with n' entries such that the set system $\mathbb{S}^j$ is over a universe set U_j. Let $U = \bigcup_{j\in[\ell]} U_j$. We associate the set system $\mathbb{S}^j$ with the j'th input bit of BP. Instead of indexing the elements of the set system $\mathbb{S}^j$ with integers from the range $[n']$, we use the indexes of the the steps of the branching program BP that inspect the j'th input. Namely,

$$\mathbb{S}^j = \left\{S^j_{k,b}\right\}_{k\in\mathsf{ind}(j),b\in\{0,1\}} \ .$$

The arithmetic circuit $F_{i,u,v,b}$ corresponds to the set $S^{\mathsf{inp}(i)}_{i,b}$. Note that for every $i \in [n]$ and every $b \in \{0,1\}$, it is indeed the case that $S^{\mathsf{inp}(i)}_{i,b} \in \mathbb{S}^{\mathsf{inp}(i)}$. This follows from the way we defined the set $\mathsf{ind}(j)$ for input bit $j \in [\ell]$, and from the way the elements of $\mathbb{S}^j$ are indexed.

The obfuscated program. The obfuscated program produced by $i\mathcal{O}$ simply contains the puzzle:

$$Z \leftarrow \mathsf{Gen}(1^\lambda, U, \vec{S}, \vec{F}) \ ,$$

where:

$$\vec{S} = \left(S^{\mathsf{inp}(i)}_{i,b}\right)_{i\in[n],u,v\in[w],b\in\{0,1\}} \quad , \quad \vec{F} = (F_{i,u,v,b})_{i\in[n],u,v\in[w],b\in\{0,1\}} \ .$$

To evaluate this obfuscated program on an input $x \in \{0,1\}^\ell$ we consider an arithmetic circuit G^x over the following input variables:

$$\{c_{i,u,v,b}\}_{i\in[n],u,v\in[w],b\in\{0,1\}} \ ,$$

were we think of the input variable $c_{i,u,v,b}$ as taking the output of the circuit $F_{i,u,v,b}$ in the circuit $G^x[\vec{F}]$. For $i \in [n]$ and $b \in \{0,1\}$ consider the $w \times w$ matrix of variables $C_{i,b} = (c_{i,u,v,b})_{u,v\in[w]}$. We let G^x be the arithmetic circuit that evaluates the expression

$$\left(\prod_{i=1}^{n} C_{i,x_{\mathsf{inp}(i)}}\right)[1,1] \ .$$

The evaluation of the obfuscated program on the input x output

$$1 - \mathsf{Ver}(Z, G^x) \ .$$

4.1 Analysis

Theorem 4.1. *If* $(\mathsf{Gen},\mathsf{Ver})$ *is a secure polynomial jigsaw puzzle scheme, the algorithm* $i\mathcal{O}$ *constructed above is a secure indistinguishability obfuscator for* $\mathbf{NC}^1$ *circuits.*

The proof of Theorem 4.1 appears in Appendix A.

5 Polynomial Jigsaw Puzzles from $i\mathcal{O}$

In this section we construct an s-bounded Polynomial Jigsaw Puzzles scheme $(\mathsf{Gen}, \mathsf{Ver})$ based on an indistinguishability obfuscator $i\mathcal{O}$ for circuits. See Section 1.2 for a high-level overview of the construction.

The puzzle generation algorithm Gen**.** The algorithm Gen takes as input:

1. The security parameter 1^λ.
2. A universe set U.
3. A vector $\vec{S} \in (2^U)^m$ of sets such that no entry of $\vec{S}$ is empty.
4. A vector $\vec{F}$ of m arithmetic circuits over the variables $(x_1, \cdots, x_k)$, where each circuit is of total degree at most d.

The algorithm Gen proceeds as follows:

1. Gen chooses a finite field F of characteristic at least $2 \cdot d \cdot 2^s$. Note that the description of elements in F is of size $\mathrm{poly}(\lambda)$.
2. Let ℓ be as follows
$$\ell = s(\lambda) + \lambda \ .$$
Gen samples a matrix $R \in F^{\ell \times k}$ of random field elements.
3. Thinking of every row of R as vector of k input elements to one of the circuits in $\vec{F}$, Gen computes the matrix $M \in F^{\ell \times m}$ of evaluations given by
$$M[i,j] = \vec{F}[j](R[i]) \ .$$
4. Gen constructs a circuit $\mathsf{V}[\lambda, U, \vec{S}, M]$ (or just V for short) as follows:
   (a) V takes as input the description of size at most $s(\lambda)$ of an arithmetic circuit G taking m input elements.
   (b) V tests that G is $\vec{S}$-respecting (with respect to the universe set U). If it is not $\vec{S}$-respecting, V outputs 0.
   (c) For every $i \in [\ell]$, V evaluates the circuit G on the inputs $M[i]$ (where we think of the row $M[i]$ as vector of m input elements to the circuit G).
   (d) If all ℓ evaluation are 0, V outputs 1 otherwise it outputs 0.
5. Gen obtains the obfuscated circuit $Z = i\mathcal{O}(\mathsf{V}[\lambda, U, \vec{S}, M])$ and outputs Z.

The solution verification algorithm Ver**.** The algorithm Ver takes as input a puzzle Z and a description of an arithmetic circuit G. Ver evaluates the obfuscated circuit Z on the description of G and output the result.

5.1 Analysis

Theorem 5.1. *If $i\mathcal{O}$ is a secure indistinguishability obfuscator for $\mathbf{P}/poly$, the algorithms $(\mathsf{Gen}, \mathsf{Ver})$ constructed above define a secure polynomial jigsaw puzzle scheme.*

Proof. We prove that the scheme defined by the algorithms $(\mathsf{Gen}, \mathsf{Ver})$ satisfies the correctness and security properties.

Correctness. Let $\lambda \in \mathbb{N}$ be a security parameter. Let U be a universe set. Let $\vec{S} \in (2^U)^m$ be a vector of m sets such that no entry of $\vec{S}$ is empty. Let $\vec{F}$ be vector of m arithmetic circuits over the variables $(x_1, \cdots, x_k)$, where each circuit is of total degree at most d.

$$\vec{F} = (F_1, \ldots F_m) \ .$$

Let:

$$Z = \mathsf{Gen}(1^\lambda, U, \vec{S}, \vec{F}) \ .$$

We prove that with overwhelming probability $1 - 2^{-\lambda}$ over the choice of Z the following holds: for every arithmetic circuit G of size at most $s(\lambda)$

$$\mathsf{Ver}(Z, G) = 1 \quad \Leftrightarrow \quad \Big(G \text{ is } \vec{S}\text{-respecting} \quad \wedge \quad G[\vec{F}] \equiv 0\Big) \ .$$

Let $R \in F^{\ell \times k}$ be the random matrix of field elements sampled by $\mathsf{Gen}(1^\lambda, U, \vec{S}, \vec{F})$ and let $M \in F^{\ell \times m}$ be the computed matrix of evaluations. By construction and by the correctness of $i\mathcal{O}$ we have that with overwhelming probability over the coins of $i\mathcal{O}$, the obfuscated circuit Z computes the same function as the circuit $\mathsf{V}[\lambda, U, \vec{S}, M]$ constructed by Gen. Therefore, it is sufficient to prove that

$$\Pr_R \Big[\forall G : \mathsf{V}[\lambda, U, \vec{S}, M](G) = 1 \quad \Leftrightarrow \quad \Big(G \text{ is } \vec{S}\text{-respecting} \quad \wedge \quad G[\vec{F}] \equiv 0\Big)\Big] \geq 1 - 2^\lambda \ .$$

Fix an arithmetic circuit G of size at most $s(\lambda)$. If G is not $\vec{S}$-respecting, then by construction,

$$\Pr\Big[\mathsf{V}[\lambda, U, \vec{S}, M](G) = 0\Big] = 1 \ .$$

Therefore, for the rest of the proof we focus on the case where G is $\vec{S}$-respecting. By construction we have that for every $i \in [\ell]$

$$G(M[i]) = G[\vec{F}](R[i]) \ .$$

If $G[\vec{F}] \equiv 0$ then for every $i \in [\ell]$

$$\Pr_R \Big[G(M[i]) = G[\vec{F}](R[i]) = 0\Big] = 1 \ ,$$

and hence

$$\Pr_R \Big[\mathsf{V}[\lambda, U, \vec{S}, M](G) = 1\Big] = 1 \ .$$

Since G is of size at most s, by Fact 2.1 G is of total degree at most 2^s. Since for every $j \in [m]$, F_j is of total degree at most d, the circuit $G[\vec{F}]$ is of total degree at most $d \cdot 2^s$. Recall that the field F is of size at least $2 \cdot d \cdot 2^s$. If $G[\vec{F}] \not\equiv 0$ then, by Fact 2.1 G, the same holds over F. Therefore, by the Schwartz-Zippel lemma, for every $i \in [\ell]$

$$\Pr_R \Big[G(M[i]) = G[\vec{F}](R[i]) = 0\Big] \leq \frac{d \cdot 2^s}{|F|} \leq \frac{1}{2} \ ,$$

and hence

$$\Pr_R \Big[\mathsf{V}[\lambda, U, \vec{S}, M](G) = 1\Big] \leq 2^\ell = 2^{s+\lambda} \ .$$

Combining all the above cases we have that:

$$\Pr_R\left[\mathsf{V}[\lambda, U, \vec{S}, M](G) = 1 \quad \not\Leftrightarrow \quad \Big(G \text{ is } \vec{S}\text{-respecting} \quad \wedge \quad G[\vec{F}] \equiv 0\Big)\right] \leq 2^{s+\lambda} \ .$$

Since the description size of the circuit G is bounded by s, there are at most 2^s possible choices for G and hance by Union bound

$$\Pr_R\left[\exists G : \mathsf{V}[\lambda, U, \vec{S}, M](G) = 1 \quad \not\Leftrightarrow \quad \Big(G \text{ is } \vec{S}\text{-respecting} \quad \wedge \quad G[\vec{F}] \equiv 0\Big)\right] \leq 2^{\lambda} \ ,$$

as required.

Security.

Let:

$$\left\{U_\lambda, \vec{S}_\lambda, \left(\vec{F}^1_\lambda, \vec{F}^2_\lambda\right)\right\}_{\lambda\in\mathbb{N}} \ ,$$

be a polynomial size ensemble such that the following conditions are satisfied

1. For every $i \in [m]$, $\vec{F}^1_\lambda[i]$ and $\vec{F}^2_\lambda[i]$ are arithmetic circuits of the same size and total degree.
2. For every $\lambda \in \mathbb{N}$ and every $\vec{S}_\lambda$-respecting arithmetic circuit G of size at most $s(\lambda)$

$$G[\vec{F}^1_\lambda] \equiv 0 \quad \Leftrightarrow \quad G[\vec{F}^2_\lambda] \equiv 0 \ .$$

For every $\lambda \in \mathbb{N}$ and let:

$$Z^1_\lambda = \mathsf{Gen}(1^\lambda, U_\lambda, \vec{S}_\lambda, \vec{F}^1_\lambda) \quad , \quad Z^2_\lambda = \mathsf{Gen}(1^\lambda, U_\lambda, \vec{S}_\lambda, \vec{F}^2_\lambda) \ .$$

By construction and by Condition 1 we have that the Z^1_λ and Z^2_λ are obfuscations of circuits of the same size. By the correctness of the scheme (proven above) and by Condition 2 we have that with overwhelming probability over the choice of Z^1_λ and Z^2_λ, the two circuits are functionally equivalent. Therefore, by the security of $i\mathcal{O}$

$$\left\{Z^1_\lambda\right\}_{\lambda\in\mathbb{N}} \approx_c \left\{Z^2_\lambda\right\}_{\lambda\in\mathbb{N}} \ ,$$

as required. □

References

[AB15] Benny Applebaum and Zvika Brakerski. Obfuscating circuits via composite-order graded encoding. In *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II*, pages 528–556, 2015.

[AGIS14] Prabhanjan Vijendra Ananth, Divya Gupta, Yuval Ishai, and Amit Sahai. Optimizing obfuscation: Avoiding barrington's theorem. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, pages 646–658, 2014.

[Bar86] David A. Mix Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in nc1. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, pages 1–5, 1986.

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, pages 440–456, 2005.

[BCKP14] Nir Bitansky, Ran Canetti, Yael Tauman Kalai, and Omer Paneth. On virtual grey box obfuscation for general circuits. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II*, pages 108–125, 2014.

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *CRYPTO*, pages 1–18, 2001.

[BGK+14] Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks. In *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, pages 221–238, 2014.

[BR14] Zvika Brakerski and Guy N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. In *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings*, pages 1–25, 2014.

[CLT13] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In *CRYPTO (1)*, pages 476–493, 2013.

[CLT15] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. New multilinear maps over the integers. *IACR Cryptology ePrint Archive*, 2015:162, 2015.

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *EUROCRYPT*, pages 1–17, 2013.

[GGH+13b] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, 2013.

[GGH15] Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. In *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II*, pages 498–527, 2015.

[GLSW14] Craig Gentry, Allison B. Lewko, Amit Sahai, and Brent Waters. Indistinguishability obfuscation from the multilinear subgroup elimination assumption. *IACR Cryptology ePrint Archive*, 2014:309, 2014.

[Kil88] Joe Kilian. Founding cryptography on oblivious transfer. In *STOC*, pages 20–31, 1988.

[PST14] Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation from semantically-secure multilinear encodings. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, pages 500–517, 2014.

[SZ14] Amit Sahai and Mark Zhandry. Obfuscating low-rank matrix branching programs. *IACR Cryptology ePrint Archive*, 2014:773, 2014.

[Zim15] Joe Zimmerman. How to obfuscate programs directly. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*, pages 439–467, 2015.

A Proof of Theorem 4.1

Proof. We start by proving that $i\mathcal{O}$ preserves functionality and then we prove that it is a secure indistinguishability obfuscator.

Correctness. Let BP be matrix branching program describing the input circuit.

$$\mathsf{BP} = \left(\mathsf{inp}(i), B_{i,0}, B_{i,1}\right)_{i=1}^{n} \ ,$$

and let $x \in \{0,1\}^{\ell}$ be an input. Using the properties of the straddling set system it is straightforward to verify that the circuit G^x is $\vec{S}$-respecting. Therefore, by the correctness of the jigsaw puzzle scheme, the obfuscated program output 1 iff the expression $G^x[\vec{F}]$ is equivalent to zero. Expending the expressions computed in the construction we have that

$$\begin{aligned}
\prod_{i=1}^{n} C_{i,x_{\mathsf{inp}(i)}} &\equiv \prod_{i=1}^{n} \alpha_{i,x_{\mathsf{inp}(i)}} \cdot \tilde{B}_{i,x_{\mathsf{inp}(i)}} \\
&\equiv \prod_{i=1}^{n} \alpha_{i,x_{\mathsf{inp}(i)}} \cdot R_{i-1} \cdot B_{i,x_{\mathsf{inp}(i)}} \cdot \mathrm{adj}(R_i) \\
&\equiv \left(\prod_{i=1}^{n} \alpha_{i,x_{\mathsf{inp}(i)}} \cdot \det(R_i)\right) \cdot \left(\prod_{i=1}^{n} B_{i,x_{\mathsf{inp}(i)}}\right) \ .
\end{aligned}$$

Therefore, by the correctness of the branching program

$$\begin{aligned}
G^x[\vec{F}] &\equiv \left(\prod_{i=1}^{n} C_{i,x_{\mathsf{inp}(i)}}\right)[1,1] \\
&\equiv \left(\prod_{i=1}^{n} \alpha_{i,x_{\mathsf{inp}(i)}} \cdot \det(R_i)\right) \cdot \left(\prod_{i=1}^{n} B_{i,x_{\mathsf{inp}(i)}}\right)[1,1] \\
&\equiv \left(\prod_{i=1}^{n} \alpha_{i,x_{\mathsf{inp}(i)}} \cdot \det(R_i)\right) \cdot (1 - \mathsf{BP}(x)) \ .
\end{aligned}$$

Overall, the expression $G^x[\vec{F}]$ is equivalent to zero iff $\mathsf{BP}(x) = 1$, as require.

Security. Let $\mathsf{BP}^1, \mathsf{BP}^2$ be a pair of matrix branching program describing two functionally equivalent circuits of the same size.

$$\mathsf{BP}^1 = \left(\mathsf{inp}(i), B^1_{i,0}, B^1_{i,1}\right)_{i=1}^{n} \quad , \quad \mathsf{BP}^2 = \left(\mathsf{inp}(i), B^2_{i,0}, B^2_{i,1}\right)_{i=1}^{n} \ .$$

Note that since the matrix branching programs are oblivious, both program use the same function inp. Let Z^1 and Z^2 be the puzzles obfuscating of the programs BP^1 and BP^2 respectively, where

$$Z^1 \leftarrow \mathsf{Gen}(1^{\lambda}, \vec{S}, \vec{F}^1) \quad , \quad Z^2 \leftarrow \mathsf{Gen}(1^{\lambda}, \vec{S}, \vec{F}^2) \ ,$$

and

$$\vec{F}^1 = \left(F^1_{i,u,v,b}\right)_{i\in[n],u,v\in[w],b\in\{0,1\}} \quad , \quad \vec{F}^2 = \left(F^2_{i,u,v,b}\right)_{i\in[n],u,v\in[w],b\in\{0,1\}} \ .$$

Note that both puzzles use the seme set vector $\vec{S}$ since this vector depends only on the function inp.

By construction, we have that for every i, u, v, b, the circuits $F^1_{i,u,v,b}$ and $F^2_{i,u,v,b}$ are of the same size and total degree. By the security of the polynomial jigsaw puzzle scheme, it is enough to prove that for every $\vec{S}_{\lambda}$-respecting arithmetic circuit G of size at most $s(\lambda)$

$$G[\vec{F}^1] \equiv 0 \quad \Leftrightarrow \quad G[\vec{F}^2] \equiv 0 \ . \tag{2}$$

In fact, we prove that (2) holds even for G that is of size larger than s.

Recall that we denote the input variables of G by

$$\left\{c_{i,u,v,b}\right\}_{i\in[n],u,v\in[w],b\in\{0,1\}} \quad .$$

Next we state a lemma on the structure of G. Roughly, the lemma says that the expression G can be written as a sum of monomials, where each monomial is the product of exactly one variable of the form $c_{i,u,v,b}$ for every $i \in [n]$. Furthermore, all these variables are "consistent with some input", meaning that for every two variables $c_{i,u,v,b}$ and $c_{i',u',v',b'}$ that belong to the same monomial, if the levels i and i' of the branching program read the same input bit, that is, if $\mathsf{inp}(i) = \mathsf{inp}(i')$, then $b = b'$.

Lemma A.1. *If G is $\vec{S}$-respecting then there exists indexes*

$$\left\{u^x_{j,i}, v^x_{j,i}\right\}_{x\in\{0,1\}^\ell, j\in\mathbb{N}, i\in[n]} \quad ,$$

such that

$$G \equiv \sum_{x\in\{0,1\}^\ell} \sum_{j} \prod_{i\in[n]} c_{i,u^x_{j,i},v^x_{j,i},x_{\mathsf{inp}(i)}} \quad .$$

Proof sketch. We consider a circuit G' which is equivalent to G except that G' is computed as a sum of monomial. That is, all multiplication gates in G' appear before all addition gates. Note that the circuit G' may be of exponential size. First we argue that if G is $\vec{S}$-respecting then G' is also $\vec{S}$-respecting. To prove this we transform the circuit G into the circuit G' in a sequence of steps, such that after each step the circuit remains $\vec{S}$-respecting. In each step we find one addition gate and one multiplication gate that together compute an expression of the form $a \cdot (b + c)$, and replace them with an addition gate and two multiplication gates computing the equivalent expression $(a \cdot b) + (a \cdot c)$. It is straightforward to verify that if the circuit was $\vec{S}$-respecting before this modification it will remain so after the modification.

It remains to prove that the circuit G' has the desired form. That is, every monomial in G' is of the form

$$\prod_{i\in[n]} c_{i,u_i,v_i,x_{\mathsf{inp}(i)}} \quad ,$$

for some $x \in \{0,1\}^\ell$ and some indexes $\{u_i, v_i\}_{i\in[n]}$. (Note that in the lemma statement we also summed over an index j. This index is to summing over all the monomials that correspond to the same input x.)

Since G' is $\vec{S}$-respecting, by the properties of an $\vec{S}$-respecting circuit, every sub-circuit of G' computing a single monomial must also be set respecting. Using again the properties of an $\vec{S}$-respecting circuit, we have that the sets associated with the monomial's input variables must form an *exact cover* of the universe set U. Since U is the disjoint union of ℓ straddling set systems $\{U_j\}_{j\in[\ell]}$, every exact cover of U is a union of exact covers for the sets $\{U_j\}$. By the properties of the straddling set systems, the only exact covers of U_j are of the form

$$\left\{S^j_{i,b}\right\}_{i\in\mathsf{ind}(j),b\in\{0,1\}} \quad .$$

Since in our construction, the variable $c_{i,u,v,b}$ is associated with the set $S^{\mathsf{inp}(i)}_{i,b}$, the set of variables that belong to every monomial must be of the form

$$\bigcup_{j\in[\ell]} \left\{c_{i,u_i,v_i,b_j}\right\}_{i\in\mathsf{ind}(j),b_j\in\{0,1\}} \quad ,$$

or equivalently, for $x \in \{0,1\}^\ell$ such that $x_j = b_j$ for every $j \in [\ell]$, the set of variables that belong to every monomial is of the form

$$\left\{c_{i,u_i,v_i,x_{\mathsf{inp}(i)}}\right\}_{i\in[n]} \quad ,$$

as required.

□

For $d \in \{1,2\}$ recall that the expression computed by the composed circuit $G[\vec{F}^d]$ is derived by replacing every variable $c_{i,u,v,b}$ in G with the output of the circuit $F^d_{i,u,v,b}$. Also recall that the circuit $F^d_{i,u,v,b}$ computes the expression

$$F^d_{i,u,v,b} \equiv \alpha_{i,b} \cdot \tilde{B}^d_{i,b}[u,v] \ ,$$

where the matrix $\tilde{B}^d_{i,b}$ is derived from the matrix $B^d_{i,b}$ as in (1). By Lemma A.1 we can therefore write the expression $G[\vec{F}^d]$ in the form

$$G[\vec{F}^d] \equiv \sum_{x \in \{0,1\}^\ell} \sum_j \prod_{i \in [n]} \alpha_{i,x_{\mathsf{inp}(i)}} \cdot \tilde{B}^d_{i,x_{\mathsf{inp}(i)}}[u^x_{j,i}, v^x_{j,i}] \ .$$

We denote by H^d_x the following polynomial over the variables X

$$H^d_x = \sum_j \prod_{i \in [n]} \tilde{B}^d_{i,x_{\mathsf{inp}(i)}}[u^x_{j,i}, v^x_{j,i}] \ .$$

We denote by α_x the monomial

$$\alpha_x = \prod_{i \in [n]} \alpha_{i,x_{\mathsf{inp}(i)}} \ .$$

Now we can rewrite $G[\vec{F}^d]$ as

$$G[\vec{F}^d] \equiv \sum_{x \in \{0,1\}^\ell} \alpha_x \cdot H^d_x \ .$$

By the algebraic independence of the monomials $\{\alpha_x\}_{x \in \{0,1\}^\ell}$ we have that $G[\vec{F}^d] \equiv 0$ iff $H^d_x \equiv 0$ for every $x \in \{0,1\}^\ell$. Therefore, to prove (2) it is sufficient to prove that for every $x \in \{0,1\}^\ell$

$$H^1_x \equiv 0 \quad \Leftrightarrow \quad H^2_x \equiv 0 \ . \tag{3}$$

Fix $x \in \{0,1\}^\ell$. Before proving (3) we introduce some additional notation. Let d be a bound on the degree of the polynomials H^1_x and H^2_x. Let F be a field of size at lease $2 \cdot d$ and of characteristic higher than $4 \cdot n$. For $d \in \{1,2\}$, we denote by $\mathsf{RanEval}(H^d_x)$ the evaluation of the expression H^d_x on random elements from F. Clearly, if $H^d_x \equiv 0$, $\mathsf{RanEval}(H^d_x)$ is supported entirely on 0. However, if $H^d_x \not\equiv 0$, by the Schwartz-Zippel lemma

$$\Pr[\mathsf{RanEval}_F(H^d_x) = 0] \leq \frac{d}{|F|} \leq \frac{1}{2} \ .$$

Next we state a lemma that follows directly from Kilian's randomization technique [Kil88].

Lemma A.2. *There exists a simulator* S *such that for every* $x \in \{0,1\}^\ell$ *and for every* $d \in \{1,2\}$ *the statistical distance between the random variables* $\mathsf{RanEval}(H^d_x)$ *and* $\mathsf{S}(\mathsf{BP}^d(x))$ *is less than* $\frac{1}{4}$.

Proof sketch. For $i \in [n]$, we denote by b_i the bit $x_{\mathsf{inp}(i)}$. Recall that the polynomial H^d_x is of the form

$$H^d_x = \sum_j \prod_{i \in [n]} \tilde{B}^d_{i,b_i}[u^x_{j,i}, v^x_{j,i}] \ . \tag{4}$$

where the matrix $\tilde{B}^d_{i,b_i}$ is given by (1)

$$\tilde{B}^d_{i,b_i} = R_{i-1} \cdot B^d_{i,b_i} \cdot \mathrm{adj}(R_i) \ .$$

When we choosing the entries of the matrices $R_1, \ldots, R_{n-1}$ randomly from the field F (recall that $R_0 = R_n = I_{w \times w}$), since the characteristic of F is $> 4 \cdot n$, every matrix R_i will be singular with probability $< \frac{1}{4 \cdot n}$. Therefore, with probability $> \frac{3}{4}$ all the matrices $R_1, \ldots, R_{n-1}$ are insertable. Given the output $\mathsf{BP}^d(x)$, the simulator S samples random matrices $\left\{\tilde{B}^d_{i,b_i}\right\}_{i \in [n]}$ conditioned on their product being consistent with the branching program output $\mathsf{BP}^d(x)$ (both should be either zero or a random non-zero element). Useing these sampled matrices, S evaluates H^d_x following (4). If follows from the proof of Kilian [Kil88] that conditioned on the event that all the matrices $R_1, \ldots, R_{n-1}$ are insertable, the random variables $\mathsf{RanEval}(H^d_x)$ and $\mathsf{S}(\mathsf{BP}^d(x))$ are identically distributed. Since the event occurs with probability $> \frac{3}{4}$, the claim follows. □

Since $\mathsf{BP}^1(x) = \mathsf{BP}^2(x)$ it follows from Lemma A.2 that

$$\left|\Pr[\mathsf{RanEval}(H^1_x) = 0] - \Pr[\mathsf{RanEval}(H^2_x) = 0]\right| < \frac{1}{2} \ .$$

Therefore (3) holds as requires. □

B Generalized Polynomial Jigsaw Puzzles

In this section we define the notion of generalized Polynomial Jigsaw Puzzles. The definition is similar to the definition of standard Polynomial Jigsaw Puzzles (Definition 3.2) except that instead of initializing the puzzle with a vector of sets $\vec{S}$ and requiring that every valid solution is $\vec{S}$-respecting, the puzzle is initialized with a description of a predicate V and valid solution are requires to satisfy this predicate.

We state Theorem B.1 which is a generalization of Theorem 5.1 for generalized Polynomial Jigsaw Puzzles. The construction and proof are essentially identical to these in Section 5.

Definition B.1 (Generalized polynomial jigsaw puzzle)**.** *A generalized polynomial jigsaw puzzle scheme consists of a pair of PPT algorithms* $(\mathsf{Gen}, \mathsf{Ver})$ *satisfying the following properties:*

Syntax *The puzzle generation algorithm* Gen *is given as input:*

- *A security parameter* 1^λ.
- *A universe set* U.
- *A solution validity predicate* V.
- *A vector* $\vec{F}$ *of* m *arithmetic circuits of over the same set of input variables* X.

Gen *outputs a puzzle* Z.

The solution verification algorithm Ver *takes a puzzle* Z *and a description of size at most* s *of an arithmetic circuit* G *taking* m *input variables.* Ver *outputs a bit.*

Correctness *For every security parameter* 1^λ*, every universe set* U*, predicate* V*, and every vector* $\vec{F}$ *of arithmetic circuits*

$$\vec{F} = (F_1, \ldots F_m) \ ,$$

following the syntax above, let

$$Z = \mathsf{Gen}(1^\lambda, U, V, \vec{F}) \ .$$

For every arithmetic circuit G *taking* m *input variables let* $G[\vec{F}]$ *be the composed circuit*

$$G[\vec{F}](X) = G(F_1(X), \ldots, F_m(X)) \ .$$

We require that with overwhelming probability over the choice of Z the following holds: for every arithmetic circuit G

$$\mathsf{Ver}(Z,G)=1 \quad \Leftrightarrow \quad \left(V_\lambda(G)=1 \quad \wedge \quad G[\vec{F}]\equiv 0\right) \ .$$

Security. *Let:*

$$\left\{U_\lambda, V_\lambda, \left(\vec{F}^1_\lambda, \vec{F}^2_\lambda\right)\right\}_{\lambda\in\mathbb{N}} \ ,$$

be a polynomial size ensemble such that:

1. *For every $i\in[m]$, $\vec{F}^1_\lambda[i]$ and $\vec{F}^2_\lambda[i]$ are arithmetic circuits of the same size and total degree.*
2. *For every $\lambda\in\mathbb{N}$ and every arithmetic circuit G such that $V_\lambda(G)=1$*

$$G[\vec{F}^1_\lambda]\equiv 0 \quad \Leftrightarrow \quad G[\vec{F}^2_\lambda]\equiv 0 \ .$$

We require that:

$$\left\{\mathsf{Gen}(1^\lambda, U_\lambda, V_\lambda, \vec{F}^1_\lambda)\right\}_{\lambda\in\mathbb{N}} \approx_c \left\{\mathsf{Gen}(1^\lambda, U_\lambda, V_\lambda, \vec{F}^2_\lambda)\right\}_{\lambda\in\mathbb{N}} \ .$$

We say the the polynomial jigsaw puzzle scheme is s-bounded for a polynomial $s=s(\lambda)$ if the description of the solution circuit G is required to be of size at most s (both in the correctness and in the security properties).

Theorem B.1. *Let s be a polynomial. Assuming indistinguishability obfuscation for $\mathbf{P}/poly$, there exist an s-bounded generalized polynomial jigsaw puzzle scheme.*