# Secret-Sharing for **NP** from Indistinguishability Obfuscation*

Ilan Komargodski
Weizmann Institute of Science
ilan.komargodski@weizmann.ac.il

Moni Naor†
Weizmann Institute of Science
moni.naor@weizmann.ac.il

Eylon Yogev
Weizmann Institute of Science
eylon.yogev@weizmann.ac.il

March 2, 2014

**Abstract**

A computational secret-sharing scheme is a method that enables a dealer, that has a secret, to distribute this secret among a set of parties such that a "qualified" subset of parties can reconstruct the secret while any "unqualified" subset of parties cannot efficiently learn anything about the secret. The collection of "qualified" subsets is defined by a monotone Boolean function.

It has been a major open problem to understand which (monotone) functions can be realized by a computational secret-sharing schemes. Yao suggested a method for secret-sharing for any function that has a polynomial-size monotone circuit (a class which is strictly smaller than the class of monotone functions in **P**). Around 1990 Rudich raised the possibility of obtaining secret-sharing for all monotone functions in **NP**: In order to reconstruct the secret a set of parties must be "qualified" and provide a witness attesting to this fact.

Recently, there has been much excitement regarding the possibility of obtaining program obfuscation satisfying the "indistinguishability obfuscation" requirement: A transformation that takes a program and outputs an obfuscated version of it so that for any two functionally equivalent programs the output of the transformation is computationally indistinguishable.

Our main result is a construction of a computational secret-sharing scheme for *any* monotone function in **NP** assuming the existence of an efficient indistinguishability obfuscator for **P** and one-way functions. Furthermore, we show how to get the same result but relying on a weaker obfuscator: an efficient indistinguishability obfuscator for 3CNF formulas.

*Research supported in part by a grant from the I-CORE Program of the Planning and Budgeting Committee, the Israel Science Foundation and the Citi Foundation.

†Incumbent of the Judith Kleeman Professorial Chair.

# Contents

# 1 Introduction

A secret-sharing scheme is a method that enables a dealer, that has a secret piece of information, to distribute this secret among $n$ parties such that a "qualified" subset of parties has enough information to reconstruct the secret while any "unqualified" subset of parties learns nothing about the secret. A monotone collection of "qualified" subsets (i.e., subsets of parties that can reconstruct the secret) is known as an access structure, and is usually identified with its characteristic monotone function.[1] Besides being interesting in their own right, secret-sharing schemes are an important building block in many cryptographic protocols, especially those involving some notion of "qualified" sets (e.g., multi-party computation, threshold cryptography and Byzantine agreement). For more information we refer to the extensive survey of Beimel on secret-sharing schemes and their applications [Bei11].

A significant goal in constructing secret-sharing schemes is to *minimize* the amount of information distributed to the parties. We say that a secret-sharing scheme is *efficient* if the size of all shares is polynomial in the number of parties and the size of the secret.

Secret-sharing schemes were introduced in the late 1970s by Blakley [Bla79] and Shamir [Sha79] for the *threshold access structure*, i.e., where the subsets that can reconstruct the secret are all the sets whose cardinality is at least a certain threshold. Their constructions were fairly efficient both in the size of the shares and in the computation required for sharing and reconstruction. Ito, Saito and Nishizeki [ISN93] considered general access structures and showed that every monotone access structure has a (possibly *inefficient*) secret-sharing scheme that realizes it. In their scheme the size of the shares is proportional to the size of the DNF for the corresponding function. Benaloh and Leichter [BL88] proved that if an access structure can be described by a polynomial-size monotone *formula*, then it has an efficient secret-sharing scheme. The most general class for which secret sharing is known was suggested by Karchmer and Wigderson [KW93] who showed that if the access structure can be described by a polynomial-size monotone *span program* (for instance, undirected connectivity in a graph), then it has an efficient secret-sharing scheme. Beimel and Ishai [BI05] proposed a secret-sharing scheme for an access structure which is conjectured to lie outside **NC**. On the other hand, there are no known lower bounds that show that there exists an access structure that requires only inefficient secret-sharing schemes.[2]

In the secret-sharing schemes considered above the security is guaranteed even if the parties are computationally unbounded. These secret-sharing schemes are known as perfect secret-sharing schemes. A natural variant, known as computational secret-sharing schemes, is to allow only computationally limited dealers and parties, i.e., they are probabilistic algorithms that run in polynomial-time. More precisely, a computational secret-sharing scheme is a secret-sharing scheme in which there exists an *efficient* dealer that generates the shares such that a "qualified" subset of parties can *efficiently* reconstruct the secret, however, an "unqualified" subset that pulls its shares together but has only limited (i.e., polynomial) computational power and attempts to reconstruct the secret should fail (with high probability). Krawczyk [Kra93] presented a computational secret-sharing scheme for threshold access structures that is more efficient (in terms of the size of the shares) than

[1]It is most sensible to consider only *monotone* sets of "qualified" subsets of parties. A set $M$ of subsets is called monotone if $A \in M$ and $A \subseteq A'$, then $A' \in M$. It is hard to imagine a meaningful method for sharing a secret to a set of "qualified" subsets that does not satisfy this property.

[2]Moreover, there are not even non-constructive lower bounds for secret-sharing schemes. The usual counting arguments (e.g., arguments that show that most functions require large circuits) do not work here since one needs to enumerate over the sharing and reconstruction algorithms whose complexity may be larger than the share size.

the perfect secret-sharing schemes given by Blakley and Shamir [Bla79, Sha79]. In an unpublished work (mentioned in [Bei11], see also Vinod et al. [VNS+03]), Yao showed an efficient computational secret-sharing scheme (assuming the existence of one-way functions) for access structures whose characteristic function can be computed by a polynomial-size monotone *circuit* (as opposed to the *perfect* secret-sharing of Benaloch and Leichter [BL88] for polynomial-size monotone *formulas*). There are access structures which are known to have an efficient *computational* secret-sharing schemes but are not known to have efficient *perfect* secret-sharing schemes, e.g., directed connectivity.[3] Yao's scheme does not include all monotone access structures with an efficient algorithm to determine eligibility. One notable example where no efficient secret-sharing is known is matching in a graph.[4] Thus, a major open problem is to understand *which access structures have efficient computational secret-sharing schemes, and what cryptographic assumptions are required for that.*

Around 1990 Steven Rudich raised the possibility of obtaining secret-sharing schemes for an even more general class of access structures than **P**: monotone functions in **NP**, also known as **mNP**.[5] An access structure that is defined by a function in **mNP** is called an **mNP** access structure. Intuitively, a secret-sharing scheme for an **mNP** access structure is defined (in the natural way) as following: for the "qualified" subsets there is a witness attesting to this fact and *given* the witness it should be possible to reconstruct the secret. On the other hand, for the "unqualified" subsets there is no witness, and so it should not be possible to reconstruct the secret. For example, consider the Hamiltonian access structure. In this access structure the parties correspond to edges of the complete undirected graph, and a set of parties $X$ is said to be "qualified" if and only if the corresponding set of edges contains a Hamiltonian cycle and the set of parties knows a witness attesting to this fact.

Rudich observed that if $\mathbf{NP} \neq \mathsf{co}\mathbf{NP}$, then there is no *perfect* secret-sharing scheme for the Hamiltonian access structure in which the sharing of the secret can be done efficiently (i.e., in polynomial-time).[6] This (conditional) impossibility result motivates looking for *computational* secret-sharing schemes for the Hamiltonian access structure and other **mNP** access structures. Furthermore, Rudich showed that the construction of a computational secret-sharing schemes for the Hamiltonian access structure gives rise to a protocol for oblivious transfer. More precisely, Rudich showed that if one-way functions exist and there is a *computational* secret-sharing scheme for the Hamiltonian access structure (i.e., with efficient sharing and reconstruction), then efficient protocols for oblivious transfer exist.[7] In particular, constructing a computational secret-sharing scheme for the Hamiltonian access structure assuming one-way functions will resolve a major open problem in cryptography and prove that Minicrypt=Cryptomania, to use Impagliazzo's terminology [Imp95].

In the decades since Rudich raised the possibility of access structures beyond **P** not much has happened. This changed with the work on Witness Encryption by Garg et al. [GGSW13], where

[3]In the access structure for directed connectivity, the parties correspond to edge slots in the complete *directed* graph and the "qualified" subsets are those edges that connect two distinguished nodes $s$ and $t$.

[4]In the access structure for matching the parties correspond to edge slots in the complete graph and the "qualified" subsets are those edges that *contain* a perfect matching. Even though matching is in **P**, it is known that there is no monotone circuit that computes it [Raz85].

[5]Rudich raised it in private communication with the second author around 1990 and was not written to the best of our knowledge; a description of some of Rudich's results can be found in Beimel's survey [Bei11] and in [Nao06].

[6]Moreover, it is possible to show that if $\mathbf{NP} \not\subseteq \mathsf{co}\mathbf{AM}$, then there is no *statistical* secret-sharing scheme for the Hamiltonian access structure in which the sharing of the secret can be done efficiently [Nao06].

[7]The resulting reduction is *non*-black-box. Also, note that the results of Rudich apply for any other monotone **NP**-complete problem as well.

the goal is to encrypt a message relative to a statement $x \in L$ for a language $L \in \mathbf{NP}$ such that: Anyone holding a witness to the statement can decrypt the message, however, if $x \notin L$, then it is computationally hard to decrypt. It is relatively simple to show that secret-sharing for an $\mathbf{NP}$ language $L$ implies witness encryption for the statement $x \in L$ (see Garg et al.). A byproduct of the proposed construction of Garg et al. was a construction of a computational secret-sharing scheme for a *specific* monotone $\mathbf{NP}$-complete language (3-EXACT COVER [Gol08, Proposition 2.25]) based on assumptions closely related to multilinear maps. However, it is unclear whether one can use a secret-sharing scheme for a specific (monotone) $\mathbf{NP}$-complete language in order to achieve secret-sharing schemes for any language in $\mathbf{mNP}$. We discuss this more in Appendix C.

Since the publication of Garg et al. [GGSW13] a proposed construction for another fascinating primitive emerged: *indistinguishability obfuscation*. In this paper, we construct a secret-sharing scheme for *every* $\mathbf{mNP}$ access structure assuming the existence of efficient indistinguishability obfuscation for all of $\mathbf{P}$ and one-way functions. In particular, our result gives an alternative construction of an efficient protocol for oblivious transfer assuming indistinguishability obfuscation and one-way functions than the one by Sahai and Waters [SW13]. We discuss the notion of indistinguishability obfuscation next and then state our main result.

**Obfuscation.** The study of methods to transform a program (say described as a Boolean circuit) into a form that is executable but otherwise completely unintelligible is a central research direction in cryptography. The latter task, known as obfuscation is an open problem in computer security from both practical and theoretical point of view.

The theoretical study of obfuscation was initiated by Barak et al. [BGI+12]. They studied several notions of obfuscation, primarily focusing on virtual black-box obfuscation. Virtual black-box obfuscation requires that anything that can be efficiently computed from the obfuscated program, can also be computed efficiently from black-box (i.e., input-output) access to the program. Their main result was that this notion of obfuscation cannot always be achieved. Indeed, they presented an explicit family of circuits that provably cannot be virtual black-box obfuscated (based on one-way functions). Other variants of definitions for obfuscation were introduced and proven to be impossible in some cases by Goldwasser and Kalai [GK05] and Goldwasser and Rothblum [GR07].

Barak et al. [BGI+12] considered also an alternative notion of obfuscation called indistinguishability obfuscation (henceforth $i\mathcal{O}$). An indistinguishability obfuscator guarantees that if two circuits compute the same function, then their obfuscated version (outputs of the obfuscator) are computationally indistinguishable. This definition is weaker than the virtual black-box one and hence may bypass the impossibility results shown for the latter. Indeed, (as shown by Barak et al.) it is easy to build *inefficient* indistinguishability obfuscators by outputting a "canonical" circuit which is equivalent to the original circuit. Apparently, one disadvantage of indistinguishability obfuscation is that it does not give an intuitive guarantee that the circuit hides information. For some functionalities, this is a major drawback.

Recently, the work of Garg et al. [GGH+13] proposed the first candidate construction of obfuscators. They show that, under new computational assumptions closely related to multilinear maps, their construction satisfies the notion of indistinguishability obfuscation. Different variants of this construction that are secure in idealized algebraic models have been proposed in [BGK+13, BR14b].

Following the work of Garg et al. it has been shown by Sahai and Waters [SW13], Hohenberger et al. [HSW13] and Boneh and Zhandry [BZ13] that $i\mathcal{O}$ can be combined with one-way functions to construct many powerful primitives such as public-key encryption, identity-based encryption, attribute-based encryption (via witness encryption), NIZK arguments, CCA encryption, deniable encryption with non-negligible advantage and traitor-tracing schemes with very short messages (note that the latter two primitives did not have any known construction). We note that without further assumptions, we cannot prove that $i\mathcal{O}$ implies one-way functions. Indeed, if $\mathbf{P} = \mathbf{NP}$ then one-way functions do not exists, but $i\mathcal{O}$ does exist since the canonizing $i\mathcal{O}$ from above can be implemented efficiently. (Recently, Moran and Rosen [MR13] showed that if $\mathbf{NP} \neq \mathsf{co}\mathbf{RP}$ and efficient indistinguishability obfuscators exist, then one-way functions exist.) Therefore, we do not expect to build many "cryptographically interesting" tools just from $i\mathcal{O}$, but usually need to combine it with other assumptions.

**Our Results.** We formally definite secret-sharing for **mNP** access structures, give two variants (indistinguishability and semantic security) and prove their equivalence. Our main result is a computational secret-sharing scheme for all **mNP** access structures assuming the existence of an efficient indistinguishability obfuscator for **P** and one-way functions.

**Theorem 1.1** (Informal)**.** *If efficient $i\mathcal{O}$ exists for* **P** *and one-way functions exist, then for every* **mNP** *access structure there is an efficient computational secret-sharing scheme.*

Moreover, we show that the same result holds even when assuming the existence of an efficient indistinguishability obfuscator for a smaller class of circuits: 3CNF formulas (for more information we refer to Section 4.2). In particular, a simple candidate for an indistinguishability obfuscator for 3CNF formulas that is provably secure in an idealized algebraic model was recently suggested by Brakerski and Rothblum [BR14a].

We remark that if we relax the requirement of computational secret-sharing such that a "qualified" subset of parties can reconstruct the secret with very high probability (say, negligibly close to 1), then our scheme from Theorem 1.1 actually gives a secret-sharing scheme for every monotone functions in **MA**.

## 1.1 Other Related Work

A different model of secret-sharing for **mNP** access structures was suggested by Vinod et al. [VNS+03]. Specifically, they relaxed the requirements of secret-sharing by introducing a semi-trusted third party $T$ who is allowed to interact with the dealer and the parties. They require that $T$ does not learn anything about the secret and the participating parties. In this model, they constructed an efficient secret-sharing scheme for any **mNP** access structures (that is also efficient in terms of the round complexity of the parties with $T$) assuming the existence of efficient oblivious transfer protocols.

## 1.2 Main Idea

Let $M$ be an **mNP** access structure on $n$ parties $\mathcal{P} = \{\mathsf{p}_1, \ldots, \mathsf{p}_n\}$ with a verifier $V_M$. We think of $V_M$ as a polynomial-size circuit. Recall that indistinguishability obfuscation ($i\mathcal{O}$) is a functionality preserving (randomized) transformation on circuits that assures that if two circuits $C_1$ and $C_2$ agree on every input and have the same size, then $i\mathcal{O}(C_1)$ is computationally indistinguishable from $i\mathcal{O}(C_2)$.

Let $\mathsf{Com}$ be a commitment scheme. A secret-sharing scheme consists of a setup phase in which the dealer distributes secret shares to the parties. For $i \in [n]$ the share of party $\mathsf{p}_i$ is composed of the following parts:

1. A secret opening $r_i$ for a commitment to the value $i$.

2. A circuit $i\mathcal{O}(C)$ where $C = C^{S,\mathsf{c}_1,\ldots,\mathsf{c}_n}$ has the following hardwired:[8]

   (a) For every $i \in [n]$ the commitment $\mathsf{c}_i = \mathsf{Com}(i, r_i)$ (of party $\mathsf{p}_i$) to the value $i$ with the opening $r_i$.

   (b) The secret $S$.

   The circuit $C$ gets as an input a subset of parties $X = \{\mathsf{p}_{i_1}, \ldots, \mathsf{p}_{i_k}\}$ and the corresponding list of alleged openings $r'_{i_1}, \ldots, r'_{i_k}$ and an alleged witness $w$ of $X$ for $V_M$. The circuit $C$ verifies that $r'_{i_1}, \ldots, r'_{i_k}$ are correct openings (i.e., it verifies that $\mathsf{c}_{i_j} = \mathsf{Com}(i_j, r'_{i_j})$ for every $j \in [k]$) and that $V_M(X, w) = 1$. If all the tests pass, it outputs the secret $S$; otherwise, it outputs $\mathsf{NUL}$.

Clearly, if $i\mathcal{O}$ and $\mathsf{Com}$ are efficient, then the generation of the shares is efficient. Moreover, the reconstruction procedure is the natural one: Given a subset of parties $X \subseteq \mathcal{P}$ such that $M(X) = 1$ and a valid witness $w$ such that $V_M(X, w) = 1$, evaluate $i\mathcal{O}(C)$ on $X$ and $w$. The tests of $C$ will pass and $i\mathcal{O}(C)$ will output the secret $S$, by the definition of the circuit $C$.

As for the *security* of this scheme, we want to show that it is impossible to extract (or even learn anything about) the secret having a subset of parties $X$ for which $M(X) = 0$ (i.e., an "unqualified" subset of parties). Let $D$ be an algorithm that extracts the secret given $X$. Roughly speaking, we will use the ability to extract the secret in order to solve the following task: we are given a list of $n$ unopened string commitments $\mathsf{c}_1, \ldots, \mathsf{c}_n$ and a promise that it either corresponds to the values $A_0 = \{1, \ldots, n\}$ or it corresponds to the values $A_1 = \{n+1, \ldots, 2n\}$ and we need to decide which is the case. Succeeding in this task would break the security guarantee of the commitment scheme.

We sample $n$ openings $r_1, \ldots, r_n$ uniformly at random and create a new circuit $C'$ such that $C' = C^{S,\mathsf{c}'_1,\ldots,\mathsf{c}'_n}$ as above, where we replace the commitments corresponding to parties not in $X$ with commitments from the input as follows:

$$\forall i \in [n]: \; \mathsf{c}'_i = \begin{cases} \mathsf{Com}(i, r_i) & \text{if } \mathsf{p}_i \in X \\ \mathsf{c}_i & \text{otherwise.} \end{cases}$$

For $i \in [n]$ we set the share of party $\mathsf{p}_i$ to be $\langle r_i, i\mathcal{O}(C') \rangle$. We run $D$ with this new set of shares. If we are in the case where $\mathsf{c}_1, \ldots, \mathsf{c}_n$ corresponds to $A_0$, then $D$ is unable to distinguish between $C$ and $C'$ and, hence, will be able to extract the secret. On the other hand, if $\mathsf{c}_1, \ldots, \mathsf{c}_n$ corresponds to $A_1$, then the circuit $C'$ is equivalent to the $\mathsf{NUL}$ circuit. In this case, it is computationally hard to extract the secret from $i\mathcal{O}(C')$ since it is computationally indistinguishable from $i\mathcal{O}(Z)$ where $Z$ is a canonical $\mathsf{NUL}$ circuit. Hence, if $D$ is able to extract the secret $S$, then we deduce that $\mathsf{c}_1, \ldots, \mathsf{c}_n$ correspond to $A_0$ and, otherwise we conclude that $\mathsf{c}_1, \ldots, \mathsf{c}_n$ correspond to $A_1$. We refer to Section 4 for the complete description of the scheme and the proof of its security.

[8] Note that the circuit is the same circuit for all the parties. Hence, the circuit is not at all secret and can be implemented only once and placed in a "shared storage" that all the parties have access to.

# 2 Preliminaries

We start with some general notation. We denote by $[n]$ the set of numbers $\{1, 2, \ldots, n\}$. Throughout the paper we use $n$ as our security parameter. We denote by $\mathbf{U}_n$ the uniform distribution on $n$ bits. For a distribution or random variable $R$ we write $r \leftarrow R$ to denote the operation of sampling a random element $r$ according to $R$. For a set $S$, we write $s \stackrel{\mathsf{R}}{\leftarrow} S$ to denote the operation of sampling an $s$ uniformly at random from the set $S$. We denote by $\mathsf{neg} : \mathbb{N} \to \mathbb{R}$ a function such that for every positive integer $c$ there exists an integer $N_c$ such that for all $n > N_c$, $\mathsf{neg}(n) < 1/n^c$.

Throughout this paper we deal with Boolean circuits. We denote by $|C|$ the size of a circuit $C$ and define it as the number of wires in $C$. Furthermore, we assume that the circuits have fan-in 2.

## 2.1 Monotone NP

A function $f : 2^{[n]} \to \{0, 1\}$ is said to be monotone if for every $X \subseteq [n]$ such that $f(X) = 1$ it also holds that $\forall Y \subseteq [n]$ such that $X \subseteq Y$ it holds that $f(Y) = 1$.

A monotone Boolean circuits is a Boolean circuit with AND and OR gates (without negations). A non-deterministic circuit is a Boolean circuit whose inputs are divided into two parts: standard inputs and non-deterministic inputs. A non-deterministic circuit accepts a standard input if and only if there is some setting of the non-deterministic input that causes the circuit to evaluate to 1. A monotone non-deterministic circuit is a non-deterministic circuit where the monotonicity requirement applies only to the standard inputs, that is, every path from a standard input wire to the output wire does not have a negation gate.

**Definition 2.1** ([GS92])**.** *We say that a function $L$ is in* **mNP** *if there exists a uniform family of polynomial-size monotone non-deterministic circuit that computes $L$.*

**Lemma 2.2** ([GS92, Theorem 2.2])**.** $\mathbf{mNP} = \mathbf{NP} \cap \mathsf{mono}$, *where* mono *is the set of all monotone functions.*

## 2.2 Computational Indistinguishability

**Definition 2.3.** *Two sequences of random variables $X = \{X_n\}_{n\in\mathbb{N}}$ and $Y = \{Y_n\}_{n\in\mathbb{N}}$ are computationally indistinguishable if for every probabilistic polynomial-time algorithm $A$ there exists an integer $N$ such that for all $n \geq N$,*

$$|\Pr[A(X_n) = 1] - \Pr[A(Y_n) = 1]| \leq \mathsf{neg}(n).$$

*where the probabilities are over $X_n$, $Y_n$ and the internal randomness of $A$.*

## 2.3 Secret-Sharing

A perfect (resp., computational) secret-sharing scheme involves a dealer who has a secret, a set of $n$ parties, and a collection $A$ of "qualified" subsets of parties called the access structure. A secret-sharing scheme for $A$ is a method by which the dealer (resp., efficiently) distributes shares to the parties such that (1) any subset in $A$ can (resp., efficiently) reconstruct the secret from its shares, and (2) any subset not in $A$ cannot (resp., efficiently) reveal any partial information on the secret. For more information on secret-sharing schemes we refer to [Bei11] and references therein.

Throughout this paper we deal with secret-sharing schemes for access structures over $n$ parties $\mathcal{P} = \mathcal{P}_n = \{\mathsf{p}_1, \ldots, \mathsf{p}_n\}$.

**Definition 2.4** (Access structure)**.** *An* access structure *$M$ on $\mathcal{P}$ is a monotone set of subset of $\mathcal{P}$. That is, for all $X \in M$ it holds that $X \subseteq \mathcal{P}$ and for all $X \in M$ and $X'$ such that $X \subseteq X' \subseteq \mathcal{P}$ it holds that $X' \in M$.*

We may think of $M$ as a characteristic function $M : 2^{\mathcal{P}} \to \{0,1\}$ that outputs 1 given as input $X \subseteq \mathcal{P}$ if and only if $X$ is in the access structure.

Many different definitions for secret-sharing schemes appeared in the literature. Some of the definitions were not stated formally and in some cases rigorous security proofs were not given. Bellare and Rogaway [BR07] survey many of these different definitions and recast them in the tradition of provable-security cryptography. They also provide some proofs for well-known secret-sharing schemes that were previously unanalyzed. We refer to [BR07] for more information.

## 2.4 Indistinguishability Obfuscation

We say that two circuits $C$ and $C'$ are *equivalent* and denote it by $C \equiv C'$ if they compute the same function (i.e., $\forall x : C(x) = C'(x)$).

**Definition 2.5** (Indistinguishability Obfuscator)**.** *Let $\mathcal{C} = \{\mathcal{C}_n\}_{n \in \mathbb{N}}$ be a class of polynomial-size circuits, where $\mathcal{C}_n$ is a set of circuits operating on inputs of length $n$. A uniform algorithm $i\mathcal{O}$ is called an* indistinguishability obfuscator *for the class $\mathcal{C}$ if it takes as input a circuit in $\mathcal{C}$ and outputs a new circuit so that following properties are satisfied:*

1. *Preserving Functionality:*

   *For any input length $n \in \mathbb{N}$ and any $C \in \mathcal{C}_n$ we have that*

   $$\Pr\left[C \equiv i\mathcal{O}(C)\right] = 1,$$ [9]

   *where the probability is over the internal randomness of $i\mathcal{O}$.*

2. *Polynomial Slowdown:*

   *There is a polynomial $p(\cdot)$ such that for any input length $n \in \mathbb{N}$ and every circuit $C \in \mathcal{C}_n$ it holds that $|i\mathcal{O}(C)| \le p(|C|)$.*

3. *Indistinguishable Obfuscation:*

   *For any probabilistic polynomial-time algorithm $D$ there exists an $n_0$ such that for any $n \ge n_0$ and any two* equivalent *circuits $C_1, C_2 \in \mathcal{C}_n$ of size $k$, it holds that*

   $$\left|\Pr\left[D\left(i\mathcal{O}\left(C_1\right)\right) = 1\right] - \Pr\left[D\left(i\mathcal{O}\left(C_2\right)\right) = 1\right]\right| \le \mathsf{neg}(k).$$

*We say that $i\mathcal{O}$ is* efficient *if it runs in polynomial-time.*

**Remark.** Our definition of Rudich secret-sharing (that is given in Section 3) is uniform. However, we note that we use a *non-uniform* definition of indistinguishability obfuscation (given in Definition 2.5) since this is the most common definition in the literature.

[9] We could also define indistinguishability obfuscator $i\mathcal{O}$ with imperfect completeness, i.e., where $\Pr\left[C \equiv i\mathcal{O}(n, C)\right] \ge 1 - \mathsf{neg}(n)$. In this case, the same proof shows that our secret-sharing scheme is secure but with imperfect completeness.

### 2.5 Commitment Schemes

A *non-interactive* statistically binding commitment scheme can be constructed based on any one-way permutation [Blu82]. Naor [Nao91] showed a construction of an *interactive* (two-round) statistically-binding commitment scheme based on any one-way function. For simplicity of presentation we will define commitment schemes in this paper to be non-interactive; however, all of our results still hold if the non-interactive commitment is replaced by Naor's construction.

**Definition 2.6** (Commitment Scheme)**.** *A polynomial-time computable function* $\mathsf{Com}\colon \{0,1\} \times \{0,1\}^n \to \{0,1\}^{p(n)}$ *(where* $p(\cdot)$ *is some polynomial) is a bit commitment scheme if it satisfies the following properties:*

1. *Computational Hiding:*

   *The random variables* $\mathsf{Com}(0; \mathbf{U}_n)$ *and* $\mathsf{Com}(1; \mathbf{U}_n)$ *are computationally indistinguishable.*

2. *Statistical Binding:*

   *The supports of the above random variables are disjoint.*

One can convert a bit commitment scheme into a *string* commitment scheme by concatenating independent commitments for each of the input bits. Thus, for $x = x_1 \cdots x_\ell \in \{0,1\}^\ell$ and $r = r^{(1)} \cdots r^{(\ell)} \in \{0,1\}^{n\ell}$ we define $\mathsf{Com}(x; r) = \mathsf{Com}(x_1; r^{(1)}) \cdots \mathsf{Com}(x_\ell; r^{(\ell)})$. We say that $\mathsf{Com}(x; r)$ is the commitment of the value $x$ with the opening $r$.

## 3 The Definition of Rudich Secret-Sharing

In this section we formally define computational secret-sharing for access structures realizing monotone functions in **NP**, which we call *Rudich secret-sharing*. Even though secret-sharing for functions in **NP** were considered in the past [VNS+03, Bei11, GGSW13], no formal definition was given. Our definition consists of two requirements: completeness and security. The *completeness* requirement assures that a "qualified" subset of parties that wishes to reconstruct the secret and *knows* the witness will be successful. The *security* requirement guarantees that as long as the parties form an "unqualified" subset, they are unable to learn the secret.

Note that the security requirement stated above is possibly hard to check efficiently: For some access structures in **mNP** (e.g., monotone **NP**-complete problems) it might be computationally hard to verify that the parties form an "unqualified" subset. Next, in Definition 3.1 we give a *uniform* definition of secret-sharing for **NP**.

**Definition 3.1** (Rudich secret-sharing)**.** *Let* $M : 2^{\mathcal{P}} \to \{0,1\}$ *be an* **mNP** *access structure with a verifier* $V_M$*. A secret-sharing scheme* $\mathcal{S}$ *for* $M$ *consists of a setup procedure* $\mathsf{SETUP}$ *and a reconstruction procedure* $\mathsf{RECON}$ *that satisfy the following requirements:*

1. $\mathsf{SETUP}(1^n, S)$ *gets as input a secret* $S$ *and distributes a share for each party. For* $i \in [n]$ *denote by* $\Pi(S,i)$ *the random variable that corresponds to the share of party* $\mathsf{p}_i$*. Furthermore, for* $X \subseteq \mathcal{P}$ *we denote by* $\Pi(S,X)$ *the random variable that corresponds to the set of shares of parties in* $X$*.*

2. *Completeness:*

*If* $\mathsf{RECON}(1^n, \Pi(S,X), w)$ *gets as input the shares of a "qualified" subset of parties and a valid witness, and outputs the shared secret. Namely, for* $X \subseteq \mathcal{P}$ *if* $M(X) = 1$*, then for any valid witness* $w$ *such that* $V_M(X, w) = 1$*, it holds that:*

$$\Pr\left[\mathsf{RECON}(1^n, \Pi(S,X), w) = S\right] = 1,$$

*where the probability is over the internal randomness of the scheme and of* $\mathsf{RECON}$*.*

3. *Indistinguishability of the Secret:*

   *For every pair of probabilistic polynomial-time algorithms* $(\mathsf{Samp}, D)$ *where* $\mathsf{Samp}(1^n)$ *defines a distribution over pairs of secrets* $S_0, S_1$*, a subset of parties* $X$ *and auxiliary information* $\sigma$*, it holds that*

$$\begin{aligned} |\Pr\left[M(X) = 0 \wedge D(1^n, S_0, S_1, \Pi(S_0, X), \sigma) = 1\right] - \\ \Pr\left[M(X) = 0 \wedge D(1^n, S_0, S_1, \Pi(S_1, X), \sigma) = 1\right]| \le \mathsf{neg}(n), \end{aligned}$$

   *where the probability is over the internal randomness of the scheme, the internal randomness of* $D$ *and the distribution* $(S_0, S_1, X, \sigma) \leftarrow \mathsf{Samp}(1^n)$*.*

   *That is, for every pair of probabilistic polynomial-time algorithms* $(\mathsf{Samp}, D)$ *such that* $\mathsf{Samp}$ *chooses two secrets* $S_0, S_1$ *and a subset of parties* $X \subseteq \mathcal{P}$*, if* $M(X) = 0$ *then* $D$ *is unable to distinguish (with noticeable probability) between the shares of* $X$ *generated by* $\mathsf{SETUP}(S_0)$ *and the shares of* $X$ *generated by* $\mathsf{SETUP}(S_1)$*.*

**Notation.** For ease of notation, $1^n$ and $\sigma$ are omitted when they are clear from the context.

### 3.1 An Alternative Definition: Semantic Security

The security requirement (i.e., the third requirement) of a Rudich secret-sharing scheme that is given in Definition 3.1 is phrased in the spirit of *computational indistinguishability*. A different approach is to define the security of a Rudich secret-sharing in the spirit of *semantic security*. As in many cases (e.g., encryption [GM84]), it turns out that the two definitions are equivalent.

**Definition 3.2** (Rudich secret-sharing - semantic security version)**.** *Let* $M : 2^{\mathcal{P}} \to \{0,1\}$ *be an* **mNP** *access structure with verifier* $V_M$*. A secret-sharing scheme* $\mathcal{S}$ *for* $M$ *consists of a setup procedure* $\mathsf{SETUP}$ *and a reconstruction procedure* $\mathsf{RECON}$ *as in Definition 3.1 and has the following property* instead *of the* indistinguishability of the secret *property:*

3 *Unlearnability of the Secret:*

   *For every pair of probabilistic polynomial-time algorithms* $(\mathsf{Samp}, D)$ *where* $\mathsf{Samp}(1^n)$ *defines a distribution over a secret* $S$*, a subset of parties* $X$ *and auxiliary information* $\sigma$*, and for every efficiently computable function* $f : \{0,1\}^* \to \{0,1\}^*$ *it holds that there exists a probabilistic polynomial-time algorithm* $D'$ *(called a* simulator*) such that*

$$\begin{aligned} |\Pr\left[M(X) = 0 \wedge D(1^n, \Pi(S,X), \sigma) = f(S)\right] - \\ \Pr\left[M(X) = 0 \wedge D'(1^n, X, \sigma) = f(S)\right]| \le \mathsf{neg}(n), \end{aligned}$$

   *where the probability is over the internal randomness of the scheme, the internal randomness of* $D$ *and* $D'$*, and the distribution* $(S, X, \sigma) \leftarrow \mathsf{Samp}(1^n)$*.*

*That is, for every pair of probabilistic polynomial-time algorithms* $(\mathsf{Samp}, D)$ *such that* Samp *chooses a secret* $S$ *and a subset of parties* $X \subseteq \mathcal{P}$*, if* $M(X) = 0$ *then* $D$ *is unable to learn anything about* $S$ *that it could not learn without access to the secret shares of* $X$*.*

**Theorem 3.3.** *Definition 3.2 and Definition 3.1 are equivalent.*

We defer the proof of Theorem 3.3 to Appendix A.

### 3.2 Definition of Adaptive Security

Our definition of Rudich secret-sharing only guarantees security against static adversaries. That is, the adversary chooses a subset of parties before it sees any of the shares. In other words, the selection is done *independently* of the sharing process and hence, we may think of it as if the sharing process is done *after* Samp chooses $X$.

A stronger security guarantee would be to require that even an adversary that chooses its set of parties in an *adaptive* manner based on the shares it has seen so far is unable to learn the secret (or any partial information about it). Namely, the adversary chooses the parties one by one depending on the secret share of the previously chosen parties.

The security proof of our scheme (which is given in Section 4) does not hold under this stronger requirement. It would be interesting to strengthen it to the adaptive case as well. One problem that immediately arises in an analysis of our scheme against adaptive adversaries is that of *selective decommitment* (cf. [DNRS03]), that is when an adversary sees a collection of commitments and can select a subset of them and receive their openings. The usual proofs of security of commitment schemes are not known to hold in this case.

## 4 Rudich Secret-Sharing from $i\mathcal{O}$

In this section we prove the main theorem of this paper. We show how to construct a Rudich secret-sharing scheme for any **mNP** access structure assuming the existence of efficient indistinguishability obfuscation for **P** and one-way functions. In Section 4.2 we strengthen this result and show a related construction that only uses an indistinguishability obfuscator for 3CNF formulas (as opposed to all of **P**).

**Theorem 1.1** (Restated)**.** *If an efficient indistinguishability obfuscator exists for all of* **P** *and one-way functions exist, then there is an efficient Rudich secret-sharing scheme for* any **mNP** *access structure.*

Let $\mathcal{P} = \{\mathsf{p}_1, \dots, \mathsf{p}_n\}$ be a set of $n$ parties and let $M : 2^{\mathcal{P}} \to \{0,1\}$ be an **mNP** access structure with a verifier $V_M$ (see Definition 2.1). We view $V_M$ as a polynomial-size circuit. Let $i\mathcal{O}$ be an efficient indistinguishability obfuscator and let $\mathsf{Com} : [2n] \times \{0,1\}^n \to \{0,1\}$ be a commitment scheme.

**The Scheme.** For every $i \in [n]$, the *share* of party $\mathsf{p}_i$ is composed of 2 components: (1) $r_i \in \{0,1\}^n$ - an opening of a commitment to the value $i$, and (2) an *obfuscated* circuit $\mathcal{O}(C)$. The circuit $C$ to be obfuscated has the following hardwired: the secret $S$ and the commitments of all parties (i.e., $\mathsf{c}_i = \mathsf{Com}(i, r_i)$ for $i \in [n]$). We stress that the openings $r_1, \dots, r_n$ of the commitments are *not* hardwired into the circuit. The input to the circuit $C$ consists of alleged $k$ openings $r'_{i_1}, \dots, r'_{i_k}$

corresponding to parties $\mathsf{p}_{i_1}, \ldots, \mathsf{p}_{i_k}$ where $k, i_1, \ldots, i_k \in [n]$ and an alleged witness $w$. The circuit $C$ first checks that the openings are valid, i.e., verifies that for every $j \in [k] : \mathsf{c}_{i_j} = \mathsf{Com}(i_j, r'_{i_j})$. Then, it verifies that the given $w$ is a valid witness, i.e., that $V_M(X, w) = 1$. If all the tests pass, $C$ outputs the secret $S$; otherwise, if any of the tests fail, the circuit $C$ outputs $\mathsf{NUL}$. The secret-sharing scheme is formally described in Figure 1 and the circuit $C$ is formally described in Figure 2.

**The Rudich Secret-Sharing Scheme $\mathcal{S}$ for $M$**

**The $\mathsf{SETUP}$ Procedure:**

*Input*: A secret $S$.

Let $i\mathcal{O}$ be an efficient indistinguishability obfuscator (see Definition 2.5).
Let $\mathsf{Com}\colon [2n] \times \{0,1\}^n \to \{0,1\}^{q(n)}$ be a string commitment scheme where $q(\cdot)$ is a polynomial (see Definition 2.6).

1. For $i \in [n]$:
   (a) Sample uniformly at random an opening $r_i \in \{0,1\}^n$.
   (b) Compute the commitment $\mathsf{c}_i = \mathsf{Com}(i, r_i)$.
2. Compute the circuit $C$ from Figure 2 where $C = C^{S,\mathsf{c}_1,\ldots,\mathsf{c}_n}$ consists of the following hardwired: the secret $S$ and the list of commitments $\mathsf{c}_1, \ldots, \mathsf{c}_n$.
3. Set the share of party $\mathsf{p}_i$ to be $\Pi(S, i) = \langle r_i, i\mathcal{O}(C) \rangle$.

**The $\mathsf{RECON}$ Procedure:**

*Input*: A non-empty subset of parties $X \subseteq \mathcal{P}$ together with their shares and a witness $w$ of $X$ for $M$.

1. Let $i\mathcal{O}(C)$ be the obfuscated circuit in the shares of $X$.
2. Evaluate the circuit $i\mathcal{O}(C)$ with the shares of $X$ and $w$ and return its output.

Figure 1: Rudich secret-sharing scheme for **NP**.

Observe that if $i\mathcal{O}$ and $\mathsf{Com}$ are both probabilistic polynomial-time algorithms, then the scheme is efficient (i.e., $\mathsf{SETUP}$ and $\mathsf{RECON}$ are probabilistic polynomial-time algorithms). $\mathsf{SETUP}$ generates $n$ commitments and an obfuscated circuit of polynomial size. $\mathsf{RECON}$ only evaluates this polynomial-size obfuscated circuit once.

**Completeness.** In the next lemma we show that the scheme is complete. That is, whenever the scheme is given a qualified $X \subseteq \mathcal{P}$ and a valid witness $w$ of $X$ for $V_M$, it is possible to successfully reconstruct the secret.

**Lemma 4.1.** *Let $M \in$ **NP** be an **mNP** access structure with a verifier $V_M$. Let $\mathcal{S} = \mathcal{S}_M$ be the scheme from Figure 1 instantiated with $M$. For every subset of parties $X \subseteq \mathcal{P}$ such that $M(X) = 1$*

**The Circuit** $C^{S,\mathsf{c}_1,\ldots,\mathsf{c}_n}$

*Hardwired into the circuit*: The secret $S$ and the commitments of all parties $\mathsf{c}_1, \ldots, \mathsf{c}_n$.

*Input to the circuit*:

1. The secret shares corresponding to a subset of parties $X$. Namely, it receives a sequence of $n$ values $r'_1, \ldots, r'_n \in \{0,1\}^n \cup \mathsf{NUL}$ such that for any $i \in [n]$ if $\mathsf{p}_i \in X$, then $r'_i$ is the alleged opening of party $\mathsf{p}_i$, and otherwise $r'_i = \mathsf{NUL}$.
2. An alleged witness $w$.

*Algorithm:*

1. Execute the following tests:
   (a) For every $i \in [n]$ such that $r_i \neq \mathsf{NUL}$, verify that the opening $r'_i$ is valid. That is, verify that $\mathsf{c}_i = \mathsf{Com}(i, r'_i)$.
   (b) Verify that the given alleged witness $w$ is a valid witness. That is, verify that $V_M(X, w) = 1$.
2. If any of the above tests fails, output $\mathsf{NUL}$; otherwise, output the secret $S$.

Figure 2: The circuit to be obfuscated from Figure 1.

*and any witness $w$ such that $V_M(X, w) = 1$ it holds that*

$$\Pr\left[\mathsf{RECON}(\Pi(S, X), w) = S\right] = 1.$$

**Proof.** Recall the definition of the (deterministic) algorithm $\mathsf{RECON}$ from Figure 1: $\mathsf{RECON}$ gets as input the shares of a subset of parties $X = \{\mathsf{p}_{i_1}, \ldots, \mathsf{p}_{i_k}\}$ for $k, i_1, \ldots, i_k \in [n]$ and a valid witness $w$. Recall that the shares of the parties in $X$ consist of $k$ openings for the corresponding commitments and an obfuscated circuit $i\mathcal{O}(C)$. $\mathsf{RECON}$ evaluates the circuit $i\mathcal{O}(C)$ given the openings of parties in $X$ and the witness $w$.

Note that $i\mathcal{O}$ perfectly preserves functionality, that is the output of $i\mathcal{O}(C)$ is identical to the output of $C$ on every input. Hence, we analyze the output of $C$ given $X$ and $w$. The verifications in Step 1a pass trivially (since the openings to the commitments are valid) and since $w$ is a valid witness $V_M(X, w) = 1$, the verification in Step 1b passes (see Figure 2). We get that $C$ (as well as $\mathsf{RECON}$) outputs the secret $S$. □

**Indistinguishability of the Secret.** We show that our scheme is secure. More precisely, we show that given an "unqualified" set of parties $X \subseteq \mathcal{P}$ as input (i.e., $M(X) = 0$), with overwhelming probability, any probabilistic polynomial-time algorithm cannot distinguish the shared secret from another. To this end, we assume towards a contradiction that such an algorithm exists and use it to efficiently solve the following task: given two lists of $n$ commitments and a promise that one of them corresponds to the values $\{1, \ldots, n\}$ and the other corresponds to the values $\{n+1, \ldots, 2n\}$, identify which one corresponds to the values $\{1, \ldots, n\}$. The following lemma shows that solving this task efficiently can be used to break the hiding property of the commitment scheme.

**Lemma 4.2.** *Let* $\mathsf{Com}\colon [2n] \times \{0,1\}^n \to \{0,1\}^{q(n)}$ *be a commitment scheme where* $q(\cdot)$ *is a polynomial. If there exist* $\varepsilon = \varepsilon(n) > 0$ *and a probabilistic polynomial-time algorithm* $D$ *for which*

$$|\Pr[D(\mathsf{Com}(1, \mathbf{U}_n), \ldots, \mathsf{Com}(n, \mathbf{U}_n)) = 1] - \Pr[D(\mathsf{Com}(n, \mathbf{U}_n), \ldots, \mathsf{Com}(2n, \mathbf{U}_n)) = 1]| \geq \varepsilon,$$

*then there exist a probabilistic polynomial-time algorithm* $D'$ *and* $x, y \in [2n]$ *such that*

$$\left|\Pr[D'(\mathsf{Com}(x, \mathbf{U}_n)) = 1] - \Pr[D'(\mathsf{Com}(y, \mathbf{U}_n)) = 1]\right| \geq \varepsilon/n.$$

The proof of the lemma follows from a standard hybrid argument. See full details in Appendix B.

At this point we are ready to prove the security of our scheme. That is, we show that the ability to break the security of our scheme translates to the ability to break the commitment scheme (using Lemma 4.2).

**Lemma 4.3.** *Let* $\mathcal{P} = \{\mathsf{p}_1, \ldots, \mathsf{p}_n\}$ *be a set of* $n$ *parties. Let* $M : 2^{\mathcal{P}} \to \{0,1\}$ *be an* **mNP** *access structure. If there exist a non-negligible* $\varepsilon = \varepsilon(n)$ *and a pair of probabilistic polynomial-time algorithms* $(\mathsf{Samp}, D)$ *such that for* $(S_0, S_1, X, \sigma) \leftarrow \mathsf{Samp}(1^n)$ *it holds that*

$$\Pr[M(X) = 0 \wedge D(S_0, S_1, \Pi(S_0, X)) = 1] - \Pr[M(X) = 0 \wedge D(S_0, S_1, \Pi(S_1, X)) = 1] \geq \varepsilon,$$

*then there exists a probabilistic algorithm* $D'$ *that runs in polynomial-time in* $n/\varepsilon$ *such that for sufficiently large* $n$

$$|\Pr[D'(\mathsf{Com}(1, \mathbf{U}_n), \ldots, \mathsf{Com}(n, \mathbf{U}_n)) = 1] - \Pr[D'(\mathsf{Com}(n+1, \mathbf{U}_n), \ldots, \mathsf{Com}(2n, \mathbf{U}_n)) = 1]| \geq \varepsilon/10 - \mathsf{neg}(n).$$

The proof of Lemma 4.3 appears in Section 4.1.

Using Lemma 4.3 we can prove Theorem 1.1, the main theorem of this section. The *completeness* requirement (Item 2 in Definition 3.1) follows directly from Lemma 4.1. The *indistinguishability of the secret* requirement (Item 3 in Definition 3.1) follows by combining Lemmas 4.2 and 4.3 together with the hiding property of the commitment scheme. Section 4.1 is devoted to the proof of Lemma 4.3.

## 4.1 Main Proof of Security

Let $M$ be an **mNP** access structure, $(\mathsf{Samp}, D)$ be a pair of algorithms and $\varepsilon > 0$ be a function of $n$, as in the Lemma 4.3. We are given a list of (unopened) string commitments $\mathsf{c}_1, \ldots, \mathsf{c}_n \in \{\mathsf{Com}(z_i, r)\}_{r \in \{0,1\}^n}$, where for $Z = \{z_1, \ldots, z_n\}$ either $Z = \{1, \ldots, n\} \triangleq A_0$ or $Z = \{n+1, \ldots, 2n\} \triangleq A_1$. Our goal is to construct an algorithm $D'$ that distinguishes between the two cases (using $\mathsf{Samp}$ and $D$) with non-negligible probability (that is related to $\varepsilon$). Recall that $\mathsf{Samp}$ chooses two secrets $S_0, S_1$ and $X \subseteq \mathcal{P}$ and then $D$ gets as input the secret shares of parties in $X$ for one of the secrets. By assumption, for $(S_0, S_1, X) \leftarrow \mathsf{Samp}(1^n)$ we have that

$$|\Pr[M(X) = 0 \wedge D(S_0, S_1, \Pi(S_0, X)) = 1] - \Pr[M(X) = 0 \wedge D(S_0, S_1, \Pi(S_1, X)) = 1]| \geq \varepsilon. \quad (1)$$

Roughly speaking, the algorithm $D'$ that we define creates a new set of shares using $\mathsf{c}_1, \ldots, \mathsf{c}_n$ such that: If $\mathsf{c}_1, \ldots, \mathsf{c}_n$ are commitments to $Z = A_0$ then $D$ is able to recover the secret; otherwise, (if $Z = A_1$) it is computationally hard to recover the secret. Thus, $D'$ can distinguish between the two cases by running $D$ on the new set of shares and acting according to its output.

We begin by describing a useful subroutine we call $\mathsf{D_{ver}}$. The inputs to $\mathsf{D_{ver}}$ are $n$ string commitments $\mathsf{c}_1, \ldots, \mathsf{c}_n$, two secrets $S_0, S_1$ and a subset of $k \in [n]$ parties $X$. Assume for ease of notations that $X = \{\mathsf{p}_1, \ldots, \mathsf{p}_k\}$. $\mathsf{D_{ver}}$ first chooses $b$ uniformly at random from the set $\{0, 1\}$ and samples uniformly at random $n$ openings $r_1, \ldots, r_n$ from the distribution $\mathbf{U}_n$. Then, $\mathsf{D_{ver}}$ computes the circuit $C'_b = C^{S_b, \mathsf{Com}(1,r_1), \ldots, \mathsf{Com}(k,r_k), \mathsf{c}_{k+1}, \ldots, \mathsf{c}_n}$ (see Figure 2) and sets for every $i \in [n]$ the share of party $\mathsf{p}_i$ to be $\langle r_i, i\mathcal{O}(C'_b) \rangle$. Finally, $\mathsf{D_{ver}}$ emulates the execution of $D$ on the set of shares $\Pi'(S_b, X)$. If the output of $D$ equals to $b$, then $\mathsf{D_{ver}}$ outputs 1 (meaning the input commitments correspond to $Z = A_0$); otherwise, $\mathsf{D_{ver}}$ outputs 0 (meaning the input commitments correspond to $Z = A_1$).

The naïve implementation of $D'$ is to run $\mathsf{Samp}$ to generate $S_0, S_1$ and $X$, run $\mathsf{D_{ver}}$ with the given string commitments, $S_0, S_1$ and $X$, and output accordingly. This, however, does not work. To see this, recall that the assumption (eq. (1)) only guarantees that $D$ is able to distinguish between the two secrets when $M(X) = 0$. However, it is possible that with high probability (yet smaller than $1 - 1/\mathsf{poly}(n)$) over $\mathsf{Samp}$ it holds that $M(X) = 1$, in which we do not have any guarantee on $D$. Hence, simply running $\mathsf{Samp}$ and $\mathsf{D_{ver}}$ might fool us in outputting the wrong answer.

The first step to solve this is to observe that, by the assumption in eq. (1), $\mathsf{Samp}$ generates an $X$ such that $M(X) = 0$ with (non-negligible) probability at least $\varepsilon$. By this observation, notice that by running $\mathsf{Samp}$ for $\Theta(n/\varepsilon)$ iterations we are assured that with very high probability (specifically, $1 - \mathsf{neg}(n)$) there exists an iteration in which $M(X) = 0$. All we are left to do is to recognize in which iteration $M(X) = 0$ and only in that iteration we run $\mathsf{D_{ver}}$ and output accordingly.

However, in general it might be computationally difficult to test for a given $X$ whether $M(X) = 0$ or not. To overcome this, we observe that we need something much simpler than testing if $M(X) = 0$ or not. All we actually need is a procedure that we call $\mathsf{B}$ that checks if $\mathsf{D_{ver}}$ is a good distinguisher (between commitments to $A_0$ and commitments to $A_1$) for a given $X$. One the one hand, by the assumption, we are assured that this is indeed the case if $M(X) = 0$. On the other hand, if $M(X) = 1$ and $\mathsf{D_{ver}}$ is biased, then simply running $\mathsf{D_{ver}}$ and outputting accordingly is enough. Thus, our goal is to estimate the bias of $\mathsf{D_{ver}}$. The latter is implemented efficiently by running $\mathsf{D_{ver}}$ independently $\Theta(n/\varepsilon)$ times on both inputs (i.e., with $Z = A_0$ and with $Z = A_1$) and counting the number of "correct" answers.

Recapping, our construction of $D'$ is as follows: $D'$ runs for $\Theta(n/\varepsilon)$ iterations such that in each iteration it runs $\mathsf{Samp}(1^n)$ and gets two secrets $S_0, S_1$ and a subset of parties $X$. Then, it estimates the *bias* of $\mathsf{D_{ver}}$ for that specific $X$ (independently of the input). If the bias is large enough, $D'$ evaluates $\mathsf{D_{ver}}$ with the input of $D'$, the two secrets $S_0, S_1$ and the subset of parties $X$ and outputs its output. The formal description of $D'$ is given in Figure 3.

**Analysis of $D'$.** We prove the following lemma which is a restatement of Lemma 4.3.

**Lemma 4.3** (Restated)**.** *Let $\mathsf{c}_1, \ldots, \mathsf{c}_n \in \{\mathsf{Com}(z_i, r)\}_{r \in \{0,1\}^n}$ be a list of string commitments, where for $Z = \{z_1, \ldots, z_n\}$ either $Z = \{1, \ldots, n\} \triangleq A_0$ or $Z = \{n+1, \ldots, 2n\} \triangleq A_1$. Assuming eq. (1), it holds that*

$$|\Pr[D'(\mathsf{c}_1, \ldots, \mathsf{c}_n) = 1 \mid Z = A_0] - \Pr[D'(\mathsf{c}_1, \ldots, \mathsf{c}_n) = 1 \mid Z = A_1]| \geq \varepsilon/10 - \mathsf{neg}(n).$$

**The algorithm $D'$**

*Input*: A sequence of commitments $\mathsf{c}_1, \ldots, \mathsf{c}_n$ where $\forall i \in [n]$: $\mathsf{c}_i \in \{\mathsf{Com}(z_i, r)\}_{r \in \{0,1\}^n}$ and for $Z = \{z_1, \ldots, z_n\}$ either $Z = \{1, \ldots, n\} \triangleq A_0$ or $Z = \{n+1, \ldots, 2n\} \triangleq A_1$.

1. Do the following for $T = n/\varepsilon$ times:
   (a) $S_0, S_1, X \leftarrow \mathsf{Samp}(1^n)$.
   (b) Run $\mathsf{bias} \leftarrow \mathsf{B}(S_0, S_1, X)$.
   (c) If $\mathsf{bias} = 1$:
      i. Run $\mathsf{resD} \leftarrow \mathsf{D}_{\mathsf{ver}}(\mathsf{c}_1, \ldots, \mathsf{c}_n, S_0, S_1, X)$.
      ii. Output $\mathsf{resD}$ (and HALT).
2. Output 0.

**The sub-procedure $\mathsf{B}$**

*Input*: Two secrets $S_0$, $S_1$ and a subset of parties $X \subseteq \mathcal{P}$.

1. Set $q_0, q_1 \leftarrow 0$. Run $T_{\mathsf{B}} = 4n/\varepsilon$ times:
   (a) $q_0 \leftarrow q_0 + \mathsf{D}_{\mathsf{ver}}(\mathsf{Com}(1, \mathbf{U}_n), \ldots, \mathsf{Com}(n, \mathbf{U}_n), S_0, S_1, X)$.
   (b) $q_1 \leftarrow q_1 + \mathsf{D}_{\mathsf{ver}}(\mathsf{Com}(n+1, \mathbf{U}_n), \ldots, \mathsf{Com}(2n, \mathbf{U}_n), S_0, S_1, X)$.
2. If $|q_0 - q_1| > n$, output 1.
3. Output 0.

**The sub-procedure $\mathsf{D}_{\mathsf{ver}}$**

*Input*: A sequence of commitments $\mathsf{c}_1, \ldots, \mathsf{c}_n$, two secrets $S_0$, $S_1$ and a subset of parties $X \subseteq \mathcal{P}$.

1. Choose $b \in \{0,1\}$ uniformly at random.
2. For $i \in [n]$: Sample $r_i \xleftarrow{\mathsf{R}} \mathbf{U}_n$ and let $\mathsf{c}'_i = \begin{cases} \mathsf{Com}(i, r_i) & \text{if } \mathsf{p}_i \in X \\ \mathsf{c}_i & \text{otherwise.} \end{cases}$
3. Compute $C'_b = C^{S_b, \mathsf{c}'_1, \ldots, \mathsf{c}'_n}$ as in Figure 2.
4. For $i \in [n]$ let the new share of party $\mathsf{p}_i$ be $\Pi'(S_b, i) = \langle r_i, i\mathcal{O}(C'_b) \rangle$.
5. Return 1 if $D(S_0, S_1, \Pi'(S_b, X)) = b$ and 0 otherwise.

Figure 3: The description of the algorithm $D'$.

We begin with the analysis of the procedure $\mathsf{D}_{\mathsf{ver}}$. In the next two claims we show that assuming that $M(X) = 0$, then $\mathsf{D}_{\mathsf{ver}}$ is a good distinguisher between the case $Z = A_0$ and the case $Z = A_1$. Specifically, the first claim states that $\mathsf{D}_{\mathsf{ver}}$ answers correctly given input $Z = A_0$ with probability at least $1/2 + \varepsilon/2$ while in the second claim we show that $\mathsf{D}_{\mathsf{ver}}$ is unable to do much better than merely guessing given input $Z = A_1$ (assuming $M(X) = 0$).

**Claim 4.4.** *For* $(S_0, S_1, X) \leftarrow \mathsf{Samp}(1^n)$ *it holds that*

$$|\Pr[\mathsf{D}_{\mathsf{ver}}(\mathsf{c}_1, \ldots, \mathsf{c}_n, S_0, S_1, X) = 1 \mid M(X) = 0 \wedge Z = A_0] - 1/2| \geq \varepsilon/2.$$

**Proof.** By the definition of $\mathsf{D}_{\mathsf{ver}}$ (see Figure 3) we have that $\mathsf{D}_{\mathsf{ver}}(\mathsf{c}_1, \ldots, \mathsf{c}_k, S_0, S_1, X) = 1$ if and only if $D(S_0, S_1, \Pi'(S_b, X)) = b$ for $b \xleftarrow{\mathsf{R}} \{0,1\}$. Since $b$ is chosen uniformly at random from $\{0,1\}$, it is enough to show that

$$\begin{aligned}\varepsilon \leq &| \Pr\left[D(S_0, S_1, \Pi'(S_1, X)) = 1 \mid M(X) = 0\right] \\ &- \Pr\left[D(S_0, S_1, \Pi'(S_0, X), \sigma) = 1 \mid M(X) = 0\right] |.\end{aligned}$$

Using the assumption (see eq. (1)), for $(S_0, S_1, X) \leftarrow \mathsf{Samp}(1^n)$ it holds that

$$\begin{aligned}\varepsilon \leq &| \Pr[M(X) = 0 \wedge D(S_0, S_1, \Pi(S_1, X)) = 1] \\ &- \Pr[M(X) = 0 \wedge D(S_0, S_1, \Pi(S_0, X)) = 1] | \\ \leq &| \Pr[D(S_0, S_1, \Pi(S_1, X), \sigma) = 1 \mid M(X) = 0] \\ &- \Pr[D(S_0, S_1, \Pi(S_0, X)) = 1 \mid M(X) = 0] |.\end{aligned}$$

Notice that since $Z = A_0$ we have that the sequence $(\mathsf{Com}(1, \mathbf{U}_n), \ldots, \mathsf{Com}(n, \mathbf{U}_n))$ is *identically* distributed as the sequence $(\mathsf{c}'_1, \ldots, \mathsf{c}'_n)$. Hence, for any $b \in \{0,1\}$ it holds that $\Pi'(S_b, X)$ is identically distributed as $\Pi(S_b, X)$. Hence,

$$\begin{aligned}\varepsilon \leq &| \Pr\left[D(S_0, S_1, \Pi'(S_1, X)) = 1 \mid M(X) = 0\right] \\ &- \Pr\left[D(S_0, S_1, \Pi'(S_0, X), \sigma) = 1 \mid M(X) = 0\right] |,\end{aligned}$$

as required. □

**Claim 4.5.** *For* $(S_0, S_1, X) \leftarrow \mathsf{Samp}(1^n)$ *it holds that*

$$|\Pr[\mathsf{D}_{\mathsf{ver}}(\mathsf{c}_1, \ldots, \mathsf{c}_n, S_0, S_1, X) = 1 \mid M(X) = 0 \wedge Z = A_1] - 1/2| \leq \mathsf{neg}(n).$$

**Proof.** Recall that $\mathsf{D}_{\mathsf{ver}}(\mathsf{c}_1, \ldots, \mathsf{c}_n, S_0, S_1, X) = 1$ if and only if for $b$ chosen uniformly at random from $\{0,1\}$ it holds that $D(S_0, S_1, \Pi'(S_b, X)) = b$.

Recall that for $b \in \{0,1\}$ and $i \in [n]$ the new share of party $\mathsf{p}_i$ denoted by $\Pi'(S_b, i)$ consists of the pair $\langle r_i^b, i\mathcal{O}(C'_b)\rangle$ where $r_i^b$ is chosen uniformly at random from $\mathbf{U}_n$. To prove the claim we show that $i\mathcal{O}(C'_0)$ and $i\mathcal{O}(C'_1)$ are computationally indistinguishable.

To this end, we show that if $Z = A_1$, then it holds that $C'_0$ is equivalent to the $\mathsf{NUL}$ circuit. The same proof shows that $C'_1$ is equivalent to the $\mathsf{NUL}$ circuit, as well. Fix an input to $C'_0$ and let $X' \subseteq \mathcal{P}$ be the corresponding set of parties. Recall that $C'_0$ verifies that the given openings are valid, runs the verifier on the input and if any test fails, it outputs $\mathsf{NUL}$.

If $X' \not\subseteq X$, then there exists an $i \in [n]$ such that $\mathsf{p}_i \in X'$ and $\mathsf{p}_i \notin X$. In this case, the opening verification test of $C'_0$ will fail: Since $Z = A_1$, for every $i$ such that $\mathsf{p}_i \notin X$ the commitment $\mathsf{c}_i$ (that is hardwired in the circuit $C'_0$) is a commitment to the value $n + i$ (and not $i$). Recall that the distributions $\mathsf{Com}(i, \mathbf{U}_n)$ and $\mathsf{Com}(j, \mathbf{U}_n)$ are *disjoint* for every $i \neq j$. Hence, any opening for the commitment $\mathsf{c}_i$ and the value $i$ is *invalid*, i.e., any opening $r'_i$ will fail the test $\mathsf{c}_i \stackrel{?}{=} \mathsf{Com}(i, r'_i)$ (see Step 1a in Figure 2).

Otherwise, if $X' \subseteq X$, then since $M$ is monotone and $M(X) = 0$ it holds that $M(X') = 0$. Therefore, there is no witness for $X'$, hence, the verifier $V_M$ will reject causing $C'_0$ to output $\mathsf{NUL}$ (see Step 1b in Figure 2).

In conclusion, for any input, $C'_0$ (resp., $C'_1$) outputs $\mathsf{NUL}$. Since both $C'_0$ and $C'_1$ are equivalent to the $\mathsf{NUL}$ circuit and they are of the same size, their obfuscations are computationally indistinguishable from one another (see Definition 2.5) and the claim follows. □

Next, we continue with two claims connecting $\mathsf{D_{ver}}$ and $\mathsf{B}$. Before we state these claims, we introduce a useful notation regarding the bias of the procedure $\mathsf{D_{ver}}$. We denote by $\mathsf{bias}(S_0, S_1, X)$ the advantage of $\mathsf{D_{ver}}$ in recognizing the case $Z = A_0$ over the case $Z = A_1$ given two secrets $S_0$ and $S_1$ and a subset of parties $X$. Namely, for any $S_0, S_1$ and $X$ denote

$$\begin{aligned}\mathsf{bias}(S_0, S_1, X) = |\Pr[&\mathsf{D_{ver}}(\mathsf{Com}(1, \mathbf{U}_n), \ldots, \mathsf{Com}(n, \mathbf{U}_n), S_0, S_1, X) = 1] \\ &- \Pr[\mathsf{D_{ver}}(\mathsf{Com}(n+1, \mathbf{U}_n), \ldots, \mathsf{Com}(2n, \mathbf{U}_n), S_0, S_1, X) = 1]|.\end{aligned}$$

The first claim states that if $\mathsf{D_{ver}}$ is biased (in the sense that $\mathsf{bias}(S_0, S_1, X)$ is large enough), then $\mathsf{B}$ almost surely notices that and outputs 1, and vice-versa, i.e., if $\mathsf{D_{ver}}$ is unbiased (in the sense that $\mathsf{bias}(S_0, S_1, X)$ is small enough), then $\mathsf{B}$ almost surely notices that and outputs 0.

**Claim 4.6.** *For* $(S_0, S_1, X) \leftarrow \mathsf{Samp}(1^n)$,

1. $\Pr[\mathsf{B}(S_0, S_1, X) = 1 \mid \mathsf{bias}(S_0, S_1, X) \geq \varepsilon/3] \geq 1 - \mathsf{neg}(n)$
2. $\Pr[\mathsf{B}(S_0, S_1, X) = 1 \mid \mathsf{bias}(S_0, S_1, X) \leq \varepsilon/10] \leq \mathsf{neg}(n)$

**Proof.** Recall that $\mathsf{B}$ runs for $T_{\mathsf{B}}$ *independent* iterations such that in each iteration it executes $\mathsf{D_{ver}}$ twice: Once with $\mathsf{Com}(1, \mathbf{U}_n), \ldots, \mathsf{Com}(n, \mathbf{U}_n)$ and once with $\mathsf{Com}(n+1, \mathbf{U}_n), \ldots, \mathsf{Com}(2n, \mathbf{U}_n)$. For $i \in [T_{\mathsf{B}}]$, let $I_0^i$ be an indicator random variable that takes the value 1 if and only if in the $i$-th iteration $\mathsf{D_{ver}}(\mathsf{Com}(1, \mathbf{U}_n), \ldots, \mathsf{Com}(n, \mathbf{U}_n), S_0, S_1, X) = 1$. Similarly, denote by $I_1^i$ an indicator random variable that takes the value 1 if and only if in the $i$-th iteration $\mathsf{D_{ver}}(\mathsf{Com}(n+1, \mathbf{U}_n), \ldots, \mathsf{Com}(2n, \mathbf{U}_n), S_0, S_1, X) = 1$. When $\mathsf{B}$ finishes, it holds that $q_0 = \sum_{i=1}^{T} I_0^i$ and $q_1 = \sum_{i=1}^{T} I_1^i$. Furthermore, if $\mathsf{bias}(S_0, S_1, X) \geq \varepsilon/3$ we get that $\mathbb{E}[|q_0 - q_1|] \geq (\varepsilon/3) \cdot T_{\mathsf{B}}$. By Chernoff's bound (see [AS08, §A.1]) we get that

$$\Pr[|q_0 - q_1| > 3/4 \cdot ((\varepsilon/3) \cdot T_{\mathsf{B}})] \geq 1 - \exp\left(O(\varepsilon \cdot T_{\mathsf{B}})\right).$$

Similarly, if $\mathsf{bias}(S_0, S_1, X) \leq \varepsilon/10$ we get that $\mathbb{E}[|q_0 - q_1|] \leq (\varepsilon/10) \cdot T_{\mathsf{B}}$. By Chernoff's bound we get that

$$\Pr[|q_0 - q_1| > 2 \cdot ((\varepsilon/10) \cdot T_{\mathsf{B}})] \leq \exp\left(O(\varepsilon \cdot T_{\mathsf{B}})\right).$$

Recall that $\mathsf{B}$ outputs 1 if and only if $|q_0 - q_1| > n$. Plugging in $T_{\mathsf{B}} = 4n/\varepsilon$ both parts of the claim follow. □

In Claim 4.6 we proved that $\mathsf{B}$ is a good estimator for the bias of $\mathsf{D_{ver}}$. That is, we showed that if $\mathsf{D_{ver}}$ is very biased, then $\mathsf{B}$ is 1 (with high probability) and vice-versa (i.e., that if $\mathsf{D_{ver}}$ is unbiased, then $\mathsf{B}$ is most likely to be 0). Denote by $\mathsf{BAD}$ the event in which $\mathsf{B}(S_0, S_1, X) = 1$ and $\mathsf{bias}(S_0, S_1, X) \leq \varepsilon/10$. In the next claim we show that the probability that $\mathsf{BAD}$ happens in any iteration of $D'$ is negligible.

**Claim 4.7.** *Denote by* $\mathsf{BAD}^i$ *the event that* $\mathsf{BAD}$ *happens in iteration* $i \in [T]$*.*

$$\Pr\left[\forall i:\ \neg\mathsf{BAD}^i\right] \geq 1 - \mathsf{neg}(n).$$

**Proof.** Since the $T$ iteration are independent and implemented identically it holds that

$$\Pr\left[\exists i:\ \mathsf{BAD}^i\right] = \sum_{i=1}^{T} \Pr\left[\mathsf{BAD}^i\right] = T \cdot \Pr\left[\mathsf{BAD}\right].$$

Observe that

$$\begin{aligned}\Pr\left[\mathsf{BAD}\right] &= \Pr\left[\mathsf{B}(S_0, S_1, X) = 1 \wedge \mathsf{bias}(S_0, S_1, X) \leq \varepsilon/10\right] \\ &\leq \Pr\left[\mathsf{B}(S_0, S_1, X) = 1 \mid \mathsf{bias}(S_0, S_1, X) \leq \varepsilon/10\right] \leq \mathsf{neg}(n).\end{aligned}$$

Hence, we get that $\Pr\left[\exists i:\ \mathsf{BAD}^i\right] \leq (n/\varepsilon) \cdot \mathsf{neg}(n) \leq \mathsf{neg}(n)$. □

The next claim states that if $X$ is such that $M(X) = 0$, then $B$ outputs 1 with very high probability. The idea is to combine Claims 4.4 and 4.5 that assure that if $M(X) = 0$, then $\mathsf{D}_{\mathsf{ver}}$ is biased (i.e., $\mathsf{bias}$ is large), with Claim 4.6 that assures that if the $\mathsf{bias}$ is large, then $\mathsf{B}$ almost surely outputs 1.

**Claim 4.8.** *For* $(S_0, S_1, X) \leftarrow \mathsf{Samp}(1^n)$*,*

$$\Pr\left[\mathsf{B}(S_0, S_1, X) = 1 \mid M(X) = 0\right] \geq 1 - \mathsf{neg}(n).$$

**Proof.** Let $(S_0, S_1, X) \leftarrow \mathsf{Samp}(1^n)$. By the definition of $\mathsf{B}$ it holds that $\mathsf{B}(S_0, S_1, X) = 1$ if and only if $q_0 - q_1 > n$. Thus, it is enough to show that

$$\Pr[q_0 - q_1 > n \mid M(X) = 0] \geq 1 - \mathsf{neg}(n).$$

Using Claims 4.4 and 4.5 we get that

$$\Pr[\mathsf{bias}(S_0, S_1, X) \geq \varepsilon/2 - \mathsf{neg}(n) \mid M(X) = 0] \geq 1 - \mathsf{neg}(n).$$

Plugging this into Claim 4.6 the claim follows. □

At this point we are finally ready to prove Lemma 4.3.

**Proof of Lemma 4.3.** Recall that our goal is to lower bound the following expression:

$$|\Pr[D'(\mathsf{c}_1, \ldots, \mathsf{c}_n) = 1 \mid Z = A_0] - \Pr[D'(\mathsf{c}_1, \ldots, \mathsf{c}_n) = 1 \mid Z = A_1]|.$$

Notice that one property of $M$ that follows from the assumption in eq. (1) is that $\Pr[M(X) = 0] \geq \varepsilon$ (where the probability if over $\mathsf{Samp}$). Combining this fact with the fact that $D'$ makes $T = n/\varepsilon$ iterations of $\mathsf{B}$ and $\Pr\left[\mathsf{B}(S_0, S_1, X) = 1 \mid M(X) = 0\right] \geq 1 - \mathsf{neg}(n)$ (by Claim 4.8), we get that $D'$ reaches Step 2 with negligible probability. In other words, with probability $1 - \mathsf{neg}(n)$ there is an iteration in which $X$ is chosen such that $M(X) = 0$ and $\mathsf{B}$ outputs 1. For the rest of the proof we assume that this is indeed the case (and lose a negligible additive term).

Furthermore, using Claim 4.7 we may also assume that in every iteration $\mathsf{BAD}$ does not happen. That is, in every iteration either $\mathsf{B}$ outputs 0 or $\mathsf{bias}$ is larger than $\varepsilon/10$. Recall that $D'$ ignores all the iteration in which $\mathsf{B}$ outputs 0. Moreover, we assumed that there is an iteration in which $\mathsf{B}$ outputs 1. In that iteration, it must be the case that the $\mathsf{bias}$ is larger than $\varepsilon/10$ which completes the proof.

□

## 4.2 Rudich Secret-Sharing from $i\mathcal{O}$ for 3CNF Formulas

In this section we show how to strengthen Theorem 1.1 and give a related construction of Rudich secret-sharing scheme for every **mNP** access structure that requires only an indistinguishability obfuscator for 3CNF formulas. The main result of this section is stated next.

**Theorem 4.9.** *If an efficient indistinguishability obfuscator exists for the family of* 3CNF *formulas and one-way functions exist, then there is an efficient Rudich secret-sharing scheme for* any **mNP** *access structure.*

The proof of Theorem 4.9 relies on the fact that it is possible to (efficiently and uniformly) transform every Boolean circuit $C$ that computes a function in **P** into a 3CNF formula $\widehat{F}$ such that $\widehat{F}(g(x)) = C(x)$, where $g$ is some efficiently computable function that translates an input of $C$ into an input of $\widehat{F}$ (see Lemma 4.10). Roughly speaking, we use this translation to transform the circuit in the secret-sharing scheme (see Figures 1 and 2) into a 3CNF formula and let the reconstruction procedure compute the transformation $g$.

For completeness, in the next lemma we show how to translate a general circuit into a 3CNF formula with the desired properties described above.

**Lemma 4.10.** *There exists a uniform polynomial-time algorithm* ToCNF *such that for any $n$,* ToCNF *takes as input a polynomial-size circuit $C : \{0,1\}^n \to \{0,1\}$ and outputs a* 3CNF *formula $\widehat{F}$ such that:*

1. *$\widehat{F}$ is of size $O(|C|)$ and has $|C|$ variables.*
2. *There exists an efficiently computable function $g \triangleq g^C : \{0,1\}^n \to \{0,1\}^{|C|}$ such that $\forall x \in \{0,1\}^n : \widehat{F}(g(x)) = C(x)$.*

**Proof.** Let $t = |C|$. The algorithm ToCNF define $t$ variables $w_1, \ldots, w_t$ that correspond to the wires of $C$. Then, for every gate in $C$, it computes the corresponding 3CNF formula that verifies consistency between the value of the two input variables and the output variable depending on the gate. Finally, ToCNF outputs the AND of these 3CNF formulas (which is a 3CNF formula).

The function $g$ on input $x$ evaluates $C(x)$ and outputs the value of the computation on each wire. Since $C$ is of polynomial-size, $g$ is efficiently computable. Moreover, by the definition of $g$, $\forall x \in \{0,1\}^n : \widehat{F}(g(x)) = C(x)$. □

Next, we prove the main theorem of this section: Theorem 4.9.

**Proof Sketch of Theorem 4.9.** The construction of the Rudich secret-sharing scheme is very similar to the construction given in Figure 1. Assume for simplicity that the secret is a single bit (to deal with arbitrary long secret we deal with each bit separately). We describe the differences of the new scheme.

Let $C$ be the circuit from the setup procedure (see Figure 1) and $\widehat{C}$ be the circuit $C$ *without* the secret and the commitments hardwired. Let $\widehat{F} \triangleq \text{ToCNF}(\widehat{C})$ and $\widehat{g} \triangleq g^{\widehat{C}}$ be as in Lemma 4.10, such that $\forall x : \widehat{F}(\widehat{g}(x)) = \widehat{C}(x)$. Set $F$ to be the 3CNF formula $\widehat{F}$ with the secret and the commitments hardwired: That is, fix the variables in $\widehat{F}$ that correspond to the wires of the secret and the commitments in $\widehat{C}$. It is easy to see that by fixing the corresponding coordinates in $\widehat{g}$ we get a function $g$ such that $\forall x : F(g(x)) = C(x)$. In the SETUP procedure, instead of obfuscating $C$, we obfuscate the 3CNF formula $F$. We modify the RECON procedure accordingly: first compute $g(X)$, evaluate the obfuscation of $F$ on $g(X)$ and output that value.

The completeness of the new scheme follows immediately from Lemma 4.10 since the circuit $F(g(\cdot))$ is equivalent to the circuit $C(\cdot)$. The proof that the new scheme is secure follows similar lines to the proof that the original scheme is secure. Specifically, the main difference is that the reconstruction algorithm also evaluates $g$ which only depends on the circuit $\widehat{C}$ and not the hardwired values. In particular, $g$ holds no information about the secret. □

# 5 Conclusions and Open Problems

We have shown a construction of a secret-sharing scheme for any **mNP** access structure. In fact, our construction yields the first computational secret-sharing scheme for *all* monotone functions in **P** (recall that not every monotone function in **P** can be computed by a polynomial-size monotone circuit, see e.g., Razborov's lower bound for matching [Raz85]). Our result seems to strengthen the view of indistinguishability obfuscation as a "central hub" for cryptography [SW13].

Our construction only requires indistinguishability obfuscation for 3CNF formulas. As we have mentioned, a simple candidate for such an obfuscator for 3CNF formulas that is provably secure in an idealized algebraic model was recently suggested by Brakerski and Rothblum [BR14a]. It may be easier to achieve a construction of a 3CNF obfuscator in the standard model based on standard hardness assumptions than an obfuscator for **P** (see [BR14a, BR14b]). In fact, there is no impossibility result for virtual black-box obfuscation for 3CNFs. We conclude with several open problems:

- Is there a secret-sharing scheme for **mNP** (even for specific monotone **NP**-complete problems) that relies only on standard hardness assumptions or at least falsifiable ones?
- Is there a way to use secret-sharing for monotone **P** to achieve secret-sharing for monotone **NP** (in a black-box manner)?
- Construct a Rudich secret-sharing scheme for every access structure in **mNP** that is secure against *adaptive* adversaries (see Section 3.2 for a discussion).

  Under a stronger obfuscation assumption, i.e., virtual black-box obfuscation or even extractability obfuscation (cf. [BGI+12, BCP14]), Zvika Brakerski observed that our construction is secure against adaptive adversaries as well.
- Is there a general way to transform any protocol that uses a trusted (third) party $T$ into one that does not use $T$ but uses an indistinguishability obfuscator instead?


# Acknowledgements

We thank Zvika Brakerski for many helpful discussions and insightful ideas. The second author thanks Steven Rudich for sharing with him his ideas on secret sharing beyond **P**.

## References


[AS08] Noga Alon and Joel Spencer. *The Probabilistic Method.* John Wiley, third edition, 2008.

[BCP14] Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability obfuscation. In *TCC*, volume 8349 of *Lecture Notes in Computer Science*, pages 52–73. Springer, 2014.

[Bei11] Amos Beimel. Secret-sharing schemes: A survey. In *IWCC*, volume 6639 of *Lecture Notes in Computer Science*, pages 11–46. Springer, 2011.

[BGI$^+$12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *Journal of the ACM*, 59(2):6, 2012. Preliminary version appeared in CRYPTO 2001.

[BGK$^+$13] Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks. *IACR Cryptology ePrint Archive*, 2013:631, 2013.

[BI05] Amos Beimel and Yuval Ishai. On the power of nonlinear secrect-sharing. *SIAM Journal on Discrete Mathematics*, 19(1):258–280, 2005.

[BL88] Josh Cohen Benaloh and Jerry Leichter. Generalized secret sharing and monotone functions. In *CRYPTO*, volume 403 of *Lecture Notes in Computer Science*, pages 27–35. Springer, 1988.

[Bla79] George R. Blakley. Safeguarding cryptographic keys. *Proceedings of the AFIPS National Computer Conference*, 22:313–317, 1979.

[Blu82] Manuel Blum. Coin flipping by telephone - a protocol for solving impossible problems. In *COMPCON*, pages 133–137. IEEE Computer Society, 1982.

[BR07] Mihir Bellare and Phillip Rogaway. Robust computational secret sharing and a unified account of classical secret-sharing goals. In *ACM Conference on Computer and Communications Security*, pages 172–184. ACM, 2007.

[BR14a] Zvika Brakerski and Guy N. Rothblum. Black-box obfuscation for d-cnfs. In *ITCS*, pages 235–250. ACM, 2014.

[BR14b] Zvika Brakerski and Guy N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. In *TCC*, pages 1–25, 2014.

[BZ13] Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. *IACR Cryptology ePrint Archive*, 2013:642, 2013.

[DNRS03] Cynthia Dwork, Moni Naor, Omer Reingold, and Larry J. Stockmeyer. Magic functions. *Journal of the ACM*, 50(6):852–921, 2003.

[GGH$^+$13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, pages 40–49, 2013.

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In *STOC*, pages 467–476. ACM, 2013.

[GK05] Shafi Goldwasser and Yael Tauman Kalai. On the impossibility of obfuscation with auxiliary input. In *FOCS*, pages 553–562. IEEE Computer Society, 2005.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[Gol08] Oded Goldreich. *Computational complexity - a conceptual perspective*. Cambridge University Press, 2008.

[GR07] Shafi Goldwasser and Guy N. Rothblum. On best-possible obfuscation. In *TCC*, volume 4392 of *Lecture Notes in Computer Science*, pages 194–213. Springer, 2007.

[GS92] Michelangelo Grigni and Michael Sipser. Monotone complexity. In *Proceedings of LMS workshop on Boolean function complexity*, volume 169, pages 57–75. Cambridge University Press, 1992.

[HSW13] Susan Hohenberger, Amit Sahai, and Brent Waters. Replacing a random oracle: Full domain hash from indistinguishability obfuscation. *IACR Cryptology ePrint Archive*, 2013:509, 2013.

[Imp95] Russell Impagliazzo. A personal view of average-case complexity. In *Structure in Complexity Theory Conference*, pages 134–147. IEEE Computer Society, 1995.

[ISN93] Mitsuru Ito, Akira Saito, and Takao Nishizeki. Multiple assignment scheme for sharing secret. *Journal of Cryptology*, 6(1):15–20, 1993.

[Kra93] Hugo Krawczyk. Secret sharing made short. In *CRYPTO*, volume 773 of *Lecture Notes in Computer Science*, pages 136–146. Springer, 1993.

[KW93] Mauricio Karchmer and Avi Wigderson. On span programs. In *Structure in Complexity Theory Conference*, pages 102–111. IEEE Computer Society, 1993.

[LL78] Nancy A. Lynch and Richard J. Lipton. On structure preserving reductions. *SIAM Journal on Computing*, 7(2):119–126, 1978.

[MR13] Tal Moran and Alon Rosen. There is no indistinguishability obfuscation in pessiland. *IACR Cryptology ePrint Archive*, 2013:643, 2013.

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, 1991.

[Nao06] Moni Naor. Secret sharing for access structures beyond P, 2006. Slides: `http://www.wisdom.weizmann.ac.il/~naor/PAPERS/minicrypt.html`.

[Raz85] Alexander A. Razborov. Lower bounds for the monotone complexity of some Boolean functions. *Dokl. Ak. Nauk. SSSR*, 281:798–801, 1985. English translation in: *Soviet Math. Dokl.* Vol 31, pp. 354-357, 1985.

[Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[Ste95] Iain A. Stewart. Complete problems for monotone NP. *Theoretical Computer Science*, 145(1&2):147–157, 1995.

[SV85] Sven Skyum and Leslie G. Valiant. A complexity theory based on boolean algebra. *Journal of the ACM*, 32(2):484–502, 1985.

[SW13] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: Deniable encryption, and more. *IACR Cryptology ePrint Archive*, 2013:454, 2013.

[VNS+03] V. Vinod, Arvind Narayanan, K. Srinathan, C. Pandu Rangan, and Kwangjo Kim. On the power of computational secret sharing. In *INDOCRYPT*, volume 2904 of *Lecture Notes in Computer Science*, pages 162–176. Springer, 2003.

# A Proof of Theorem 3.3

In this section we prove that Definition 3.1 is equivalent to Definition 3.2.

**Proof that Definition 3.2 implies Definition 3.1.** Let $\mathcal{S}$ be a Rudich secret-sharing scheme satisfying Definition 3.2 and assume towards contradiction that it does *not* satisfy Definition 3.1. That is, there is a pair of probabilistic polynomial-time algorithms $(\mathsf{Samp}, D)$ and a non-negligible $\varepsilon$ such that for $(S_0, S_1, X, \sigma) \leftarrow \mathsf{Samp}(1^n)$ it holds that

$$\begin{aligned} |\Pr[M(X) = 0 \wedge D(1^n, S_0, S_1, \Pi(S_0, X), \sigma) = 1] - \\ \Pr[M(X) = 0 \wedge D(1^n, S_0, S_1, \Pi(S_1, X), \sigma) = 1]| \geq \varepsilon. \end{aligned} \tag{2}$$

For a bit $b$ chosen uniformly at random from $\{0,1\}$, we have that

$$\begin{aligned} &\Pr[M(X) = 0 \wedge D(1^n, S_0, S_1, \Pi(S_b, X), \sigma) = b] = \\ &\quad \frac{1}{2}(\Pr[D(1^n, S_0, S_1, \Pi(S_0, X), \sigma) = 0 \mid M(X) = 0] \cdot \Pr[M(X) = 0] \\ &\quad + \Pr[M(X) = 0 \wedge D(1^n, S_0, S_1, \Pi(S_1, X), \sigma) = 1]) = \\ &\quad \frac{1}{2}(\Pr[M(X) = 0] - \Pr[M(X) = 0 \wedge D(1^n, S_0, S_1, \Pi(S_0, X), \sigma) = 1] \\ &\quad + \Pr[M(X) = 0 \wedge D(1^n, S_0, S_1, \Pi(S_1, X), \sigma) = 1]). \end{aligned}$$

Plugging in eq. (2) we get that

$$|\Pr[M(X) = 0 \wedge D(1^n, S_0, S_1, \Pi(S_b, X), \sigma) = b] - 1/2 \cdot (\Pr[M(X) = 0])| \geq \varepsilon/2.$$

Assume that $\mathsf{Samp}$ generates secrets in $[2^t]$ for some $t > 0$. Let $\mathcal{F} = \{f_i \colon [2^t] \to \{0,1\} \mid i \in [t] \wedge \forall x \in [2^t] : \ f_i(x) = \mathsf{bin}(x)_i\}$ be the set of all dictator functions, where $\mathsf{bin}(x)$ denotes the binary representation of $x$ of length $t$ (with leading zeroes if needed). We define a sampling algorithm $\mathsf{Samp}'$ as follows: $\mathsf{Samp}'(1^n)$ first runs $\mathsf{Samp}(1^n)$ and gets two secrets $S_0, S_1$, a subset of parties $X$ and auxiliary information $\sigma$. Then, $\mathsf{Samp}'$ chooses a bit $b \in \{0,1\}$ uniformly at random and outputs $(S_b, X, \sigma')$, where $\sigma' = \langle S_0, S_1, \sigma \rangle$. The algorithm $D'$ emulates the execution of $D$ with inputs $S_0, S_1$, $\Pi(S_b, X)$ and $\sigma'$. Note that $D'$ does not know the bit $b$. Denote by $\mathcal{F}' \subseteq \mathcal{F}$ the set of function $f \in \mathcal{F}$ for which $f(S_0) \neq f(S_1)$. Observe that with probability strictly larger than 0 over a random choice of $f$ from $\mathcal{F}$ it holds that $f \in \mathcal{F}'$ (i.e., $\mathcal{F}'$ is not empty). Then, over the randomness of $\mathsf{Samp}'$ we have that for any $f \in \mathcal{F}'$

$$|\Pr\left[M(X) = 0 \wedge D'(1^n, \Pi(S_b, X), \sigma') = f(S_b)\right] - 1/2 \cdot \Pr[M(X) = 0]| \geq \varepsilon/2. \tag{3}$$

On the other hand, since $X$ does not have any information about $S_0, S_1$ and $b$ is chosen uniformly at random from $\{0,1\}$, for any algorithm $D''$ and every $f \in \mathcal{F}'$ it holds that

$$\Pr\left[D''(1^n, X, \sigma') = f(S_b)\right] = 1/2.$$

Thus,

$$\Pr\left[M(X) = 0 \wedge D''(1^n, X, \sigma') = f(S_b)\right] = 1/2 \cdot \Pr[M(X) = 0]. \tag{4}$$

Combining eqs. (3) and (4) we get that for any $f \in \mathcal{F}'$:

$$|\Pr\left[M(X) = 0 \wedge D'(1^n, \Pi(S_b, X), \sigma') = f(S_b)\right] - \Pr\left[M(X) = 0 \wedge D''(1^n, X, \sigma') = f(S_b)\right]| \geq \varepsilon/2$$

which contradicts the unlearnability requirement of Definition 3.2. □

**Proof that Definition 3.1 implies Definition 3.2.** Let $\mathcal{S}$ be a Rudich secret-sharing scheme satisfying Definition 3.1. Fix a pair of algorithms $(\mathsf{Samp}, D)$ and a function $f$ as in Definition 3.2. We define a simulator $D'$ as follows:

$$D'(1^n, X, \sigma) = D(1^n, \Pi(0, X), \sigma).$$

We prove that this simulator satisfies the *unlearnability of the secret* requirement in Definition 3.2. Namely, we show that

$$|\Pr[M(X) = 0 \wedge D(1^n, \Pi(S, X), \sigma) = f(S)] - \Pr[M(X) = 0 \wedge D'(1^n, X, \sigma) = f(S)]| \leq \mathsf{neg}(n).$$

Towards this end, assume towards contradiction that there exists a non-negligible $\varepsilon = \varepsilon(n)$ such that

$$|\Pr[M(X) = 0 \wedge D(1^n, \Pi(S, X), \sigma) = f(S)] - \Pr[M(X) = 0 \wedge D'(1^n, X, \sigma) = f(S)]| \geq \varepsilon.$$

Plugging in the definition of $D'$ we have that

$$|\Pr[M(X) = 0 \wedge D(1^n, \Pi(S, X), \sigma) = f(S)] - \Pr[M(X) = 0 \wedge D(1^n, \Pi(0, X), \sigma) = f(S)]| \geq \varepsilon.$$

Next, we define a pair of algorithms $(\mathsf{Samp}'', D'')$ that are good distinguishers between two secrets which, in turn, contradicts the *indistinguishability of the secret* requirement from Definition 3.1 that $\mathcal{S}$ satisfies. The sampling algorithm $\mathsf{Samp}''$ simply runs $\mathsf{Samp}$ to get $(S, X, \sigma)$ and output $(0, S, X, \sigma)$. The distinguisher $D''$ is defined as follows: For every $b \in \{0,1\}:\ D''(1^n, S_0, S_1, \Pi(S_b, X), \sigma) = 1$ if and only if $D(1^n, \Pi(S_b, X), \sigma) = f(S_1)$. Using this $D''$ we get that

$$|\Pr[M(X) = 0 \wedge D''(1^n, S_0, S_1, \Pi(S_1, X), \sigma) = 1] - \Pr[M(X) = 0 \wedge D''(1^n, S_0, S_1, \Pi(S_0, X), \sigma) = 1]| \geq \varepsilon,$$

which contradicts the indistinguishability assumption. □

## B Proof of Lemma 4.2

In this section we prove the following lemma.

**Lemma 4.2** (Restated)**.** *Let* $\mathsf{Com}\colon [2n] \times \{0,1\}^n \to \{0,1\}^{q(n)}$ *be a commitment scheme where* $q(\cdot)$ *is a polynomial. If there exist* $\varepsilon = \varepsilon(n) > 0$ *and a probabilistic polynomial-time algorithm* $D$ *for which*

$$|\Pr[D(\mathsf{Com}(1, \mathbf{U}_n), \ldots, \mathsf{Com}(n, \mathbf{U}_n)) = 1] - \Pr[D(\mathsf{Com}(n, \mathbf{U}_n), \ldots, \mathsf{Com}(2n, \mathbf{U}_n)) = 1]| \geq \varepsilon,$$

*then there exist a probabilistic polynomial-time algorithm* $D'$ *and* $x, y \in [2n]$ *such that*

$$\left|\Pr[D'(\mathsf{Com}(x, \mathbf{U}_n)) = 1] - \Pr[D'(\mathsf{Com}(y, \mathbf{U}_n)) = 1]\right| \geq \varepsilon/n.$$

**Proof.** Assume that there exists a polynomial-time algorithm $D$ and some $\varepsilon = \varepsilon(n)$ such that

$$|\Pr[D(\mathsf{Com}(1, \mathbf{U}_n), \ldots, \mathsf{Com}(n, \mathbf{U}_n)) = 1] - \Pr[D(\mathsf{Com}(n+1, \mathbf{U}_n), \ldots, \mathsf{Com}(2n, \mathbf{U}_n)) = 1]| \geq \varepsilon. \tag{5}$$

For $\sigma \in [2n]$ let $\mathsf{c}_\sigma$ be a random variable sampled according to the distribution $\mathsf{Com}(\sigma, \mathbf{U}_n)$. With this notation, eq. (5) can be rewritten as

$$\left|\Pr[D(\mathsf{c}_1, \ldots, \mathsf{c}_n) = 1] - \Pr[D(\mathsf{c}_{n+1}, \ldots, \mathsf{c}_{2n}) = 1]\right| \geq \varepsilon. \tag{6}$$

For $1 \leq i \leq n-1$ let $\mathcal{C}^{(i)}$ be the distribution induced by the sequence $\mathsf{c}_1, \ldots, \mathsf{c}_{n-i}, \mathsf{c}_{2n-i+1}, \ldots, \mathsf{c}_{2n}$. Moreover, let $\mathcal{C}^{(0)}$ be the distribution $\mathsf{c}_1, \ldots, \mathsf{c}_n$ and let $\mathcal{C}^{(n)}$ be the distribution $\mathsf{c}_{n+1}, \ldots, \mathsf{c}_{2n}$. Using this notation, eq. (6) can be rewritten as

$$\left|\Pr[D(\mathcal{C}^{(0)}) = 1] - \Pr[D(\mathcal{C}^{(k)}) = 1]\right| \geq \varepsilon.$$

By a hybrid argument, there exists an index $i \in [n]$ for which

$$\left|\Pr[D(\mathcal{C}^{(i-1)}) = 1] - \Pr[D(\mathcal{C}^{(i)}) = 1]\right| \geq \varepsilon/n.$$

Expanding the definition of $\mathcal{C}^{(i)}$,

$$|\Pr[D\left(\mathsf{c}_1, \ldots, \mathsf{c}_{n-i}, \mathsf{c}_{n-i+1}, \mathsf{c}_{2n-i+2}, \ldots, \mathsf{c}_{2n}\right) = 1] - \Pr[D(\mathsf{c}_1, \ldots, \mathsf{c}_{n-i}, \mathsf{c}_{2n-i+1}, \mathsf{c}_{2n-i+2}, \ldots, \mathsf{c}_{2n}) = 1]| \geq \varepsilon/n.$$

At this point, it follows that there exists $D'$ that distinguishes between $\mathsf{c}_{n-i+1}$ and $\mathsf{c}_{2n-i+1}$. Namely, for $x = n - i + 1$ and $y = 2n - i + 1$, it holds that

$$\left|\Pr[D'(\mathsf{Com}(x, \mathbf{U}_n)) = 1] - \Pr[D'(\mathsf{Com}(y, \mathbf{U}_n)) = 1]\right| \geq \varepsilon/n,$$

as required. □

# C On Completeness for Rudich Secret-Sharing

In this section we characterize which languages in **mNP** are the "hardest" for Rudich secret-sharing: languages for which the existence of a Rudich secret-sharing scheme implies a scheme for all **mNP**.

We observe that if one has a secret-sharing scheme for a language in **mNP** that is also complete for **mNP** under *monotone-circuit witness-preserving* reductions (to be defined next), then we could get a secret-sharing scheme for all **mNP**.

**Definition C.1** (Monotone-Circuit Witness-Preserving (MCWP) Reduction)**.** *Let $L : 2^{[n]} \to \{0,1\} \in$ **mNP** and $L' : 2^{[m]} \to \{0,1\} \in$ **mNP** be functions (or equivalently, sets of subsets of $[n]$ and $[m]$, respectively). $L'$ is said to be MCWP-reducible to $L$ if the following requirements hold:*

1. *There exists a uniform polynomial-time algorithm that generates a sequence of $n$ monotone circuits $\sigma_1, \ldots, \sigma_n : 2^{[m]} \to \{0,1\}$ such that for $x' \subseteq [m]$ and $x = (\sigma_1(x'), \ldots, \sigma_n(x')) \subseteq [n]$ it holds that $x' \in L'$ if and only if $x \in L$.*
2. *There exists an efficiently computable function $g : 2^{[m]} \times \{0,1\}^* \to \{0,1\}^*$ such that if $w'$ is a witness for $x' \in L'$, then $w = g(x', w')$ is a witness for $x = (\sigma_1(x'), \ldots, \sigma_n(x'))$.*

We emphasis that Definition C.1 is a strengthening of the usual definition of a reduction between **NP** problems in two ways. First, we require the reduction to be efficiently computable by a polynomial-size monotone circuit. Second, we require the reduction to provide an efficiently computable correspondence between witnesses.

In the next lemma, we show that having a secret-sharing scheme for a problem that is complete for **mNP** under MCWP-reductions, implies a secret-sharing scheme for all **mNP**.

**Lemma C.2.** *Let $L : 2^{[n]} \to \{0,1\} \in$ **mNP** be a function that defines an access structure. If there is a Rudich secret-sharing scheme for $L$ and a MCWP-reduction from $L' : 2^{[m]} \to \{0,1\} \in$ **mNP** to $L$, then there is a Rudich secret-sharing scheme for $L'$.*

**Proof Sketch.** Let $L$ and $L'$ be as in the lemma. Assume that $L$ is defined over parties $\mathcal{P} = \{\mathsf{p}_1, \ldots, \mathsf{p}_n\}$ and $L'$ is defined over parties $\mathcal{P}' = \{\mathsf{p}'_1, \ldots, \mathsf{p}'_m\}$. Our goal is to construct a secret-sharing scheme for $L'$ given a secret $S$. Since $L$ has a Rudich secret-sharing scheme $\mathcal{S}_L$ (see Definition 3.1), there exists a procedure $\mathsf{SETUP}_L$ that get as input the secret $S$ and generates secret shares for $\mathsf{p}_1, \ldots, \mathsf{p}_n$. Denote these secret-shares by $s_1 \triangleq \Pi(S,1), \ldots, s_n \triangleq \Pi(S,n)$. Moreover, there is a procedure $\mathsf{RECON}_L$ that given the shares of a subset of parties and a (valid) witness for them, reconstructs the secret.

Since there is a MCWP-reduction from $L'$ to $L$, then an instance of $L'$ can be transformed to an instance of $L$ using a sequence of $n$ polynomial-size monotone Boolean circuits $\sigma_1, \ldots, \sigma_n$, one for each party of $L$. In other words, each of these $n$ functions defines, using a polynomial-size monotone circuit, an access structure. Hence, we can use Yao's scheme (see also [VNS⁺03]) and get a secret-sharing scheme $\mathcal{S}_{\sigma_i}$ for each $i \in [n]$. That is, there is a sequence of $n$ setup and reconstruction procedures: $\mathsf{SETUP}_{\sigma_1}, \ldots, \mathsf{SETUP}_{\sigma_n}$ and $\mathsf{RECON}_{\sigma_1}, \ldots, \mathsf{RECON}_{\sigma_n}$, respectively.

The idea is as following. First, we run the $\mathsf{SETUP}_L(S)$ procedure with the secret $S$ as input and get secret shares for $\mathsf{p}_1, \ldots, \mathsf{p}_n$. Denote these shares by $\Pi_{\mathsf{p}_1}, \ldots, \Pi_{\mathsf{p}_n}$. By the definition of the reductions, the existence of each $\mathsf{p}_i$ after the reduction depends on $\sigma_i$. Thus, we use the secret shares of $\mathsf{p}_1, \ldots, \mathsf{p}_n$ as *secrets* for the setup procedures of the $\sigma_i$s. That is, for each $i \in [n]$ we run $\mathsf{SETUP}(\Pi_{\mathsf{p}_i})$ and get $\Pi^i_{\mathsf{p}'_1}, \ldots, \Pi^i_{\mathsf{p}'_m}$. Finally, for each $i \in [n]$ we define the secret share of party $\mathsf{p}'_i$ to be $\langle \Pi^1_{\mathsf{p}'_i}, \ldots, \Pi^n_{\mathsf{p}'_i} \rangle$. The precise description of the scheme is given in Figure 4.

**The Rudich Secret-Sharing Scheme for $L'$**

- Let $\mathsf{SETUP}_L$ and $\mathsf{RECON}_L$ be the setup and reconstruction procedures in the Rudich secret-sharing scheme for $L$ (on $n$ parties $\{\mathsf{p}_1, \ldots, \mathsf{p}_n\}$), respectively.
- For $i \in [n]$ let $\mathsf{SETUP}_{\sigma_i}$ and $\mathsf{RECON}_{\sigma_i}$ be the setup and reconstruction procedures in the secret-sharing scheme for $\sigma_i$ (on $m$ parties $\{\mathsf{p}'_1, \ldots, \mathsf{p}'_m\}$), respectively.
- Let $g : \{0,1\}^* \to \{0,1\}^*$ be the function that gets a subset of parties in $\mathcal{P}'$ and an alleged witness for $L'$ and outputs a corresponding witness for $L$.

**The $\mathsf{SETUP}$ Procedure:**

*Input*: A secret $S$.

1. Run $\mathsf{SETUP}_L(S)$ and get $\Pi_{\mathsf{p}_1}, \ldots, \Pi_{\mathsf{p}_n}$.
2. For $i \in [n]$ run $\mathsf{SETUP}_{\sigma_i}(\Pi_{\mathsf{p}_i})$ and get $\Pi^i_{\mathsf{p}'_1}, \ldots, \Pi^i_{\mathsf{p}'_m}$.
3. For $i \in [m]$ set the share of party $\mathsf{p}'_i$ to be $\langle \Pi^1_{\mathsf{p}'_i}, \ldots, \Pi^n_{\mathsf{p}'_i} \rangle$

**The $\mathsf{RECON}$ Procedure:**

*Input*: A non-empty subset of parties $X' \subseteq \mathcal{P}'$ together with their shares and a witness $w'$ of $X'$.

1. Let $X \subseteq \mathcal{P}$ be a set of parties such that $\mathsf{p}_i \in \mathcal{P}$ if and only if $\sigma_i(X') = 1$.
2. For $i \in [n]$ such that $\mathsf{p}_i \in X$ execute $\mathsf{RECON}_{\sigma_i}(X')$ and get $\Pi'_{\mathsf{p}_i}$.
3. Compute $w \leftarrow R(X', w')$.
4. Execute $\mathsf{RECON}_L$ with the shares $\{\Pi'_{\mathsf{p}_i}\}_{\mathsf{p}_i \in X}$ and the witness $w$.

Figure 4: Rudich secret-sharing scheme for **NP**

**Completeness (Sketch).** Assume that we are given a "qualified" subset of parties $X$ together with their shares and a corresponding valid witness $w'$. Let $X \subseteq \mathcal{P}$ be a subset of parties such that $\mathsf{p}_i \in X$ if $\Pi'_{\mathsf{p}_i} = \Pi_{\mathsf{p}_i}$. By the correctness of the reduction it must be the case that $X \in L$ and $w$ is a witness for $X$. Hence, by the correctness of the secret-sharing scheme for $L$, it must be the case that the scheme outputs the secret $S$.

**Security. (Sketch)** Assume that we are given an "unqualified" subset of parties $X$ together with their shares and an alleged witness. Let $X \subseteq \mathcal{P}$ be a subset of parties such that $\mathsf{p}_i \in X$ if $\Pi'_{\mathsf{p}_i} = \Pi_{\mathsf{p}_i}$. By the correctness of the reduction it must be the case that $X \notin L$. Hence, by the security of the secret-sharing scheme for $L$, it must be the case that the scheme does not outputs the secret $S$. □

**Completeness under MCWP-reductions.** As we have seen, having a Rudich secret-sharing scheme for a language in **mNP** that is complete under MCWP-reduction gives rise to schemes for all **mNP**. MCWP-reduction require two properties. The second property (i.e., that it is *witness preserving*) usually follows immediately from the correctness of the reduction (a thorough discussion is given by Lynch and Lipton [LL78]). However, the first property (of circuit monotonicity) is more subtle and harder to achieve.

A specific type of reductions that satisfies the first property of Definition C.1, called *monotone projection translations*, was introduced by Skyum and Valiant [SV85] and further studied by Stewart [Ste95]. A monotone projection of a Boolean function is a function obtained by substituting for each of its variables a variable or a constant.[10]

Stewart [Ste95] proved that the problem $\mathsf{DHam}_{s,t}$ of deciding whether a digraph has a Hamiltonian path between two specified vertices $s, t$ and the problem $\mathsf{CUB}$ of deciding whether a given graph has a cubic subgraph (i.e., a subgraph where each vertex has degree 3) are *complete* for **mNP** via monotone projection translations.

[10]More precisely, a function $f(x_1, \ldots, x_n)$ is said to be a monotone projection of a function $g(y_1, \ldots, y_m)$ if and only if there is a mapping $\sigma \colon \{y_1, \ldots, y_n\} \to \{0, 1, x_1, \ldots, x_n\}$ such that $f(x_1, \ldots, x_n) = g(\sigma(y_1), \ldots, \sigma(y_m))$.