J Cryptol (2017) 30:444–469
DOI: 10.1007/s00145-015-9226-0

Journal of
CRYPTOLOGY

Secret-Sharing for NP[*]

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

Moni Naor
Weizmann Institute of Science, Rehovot, Israel
moni.naor@weizmann.ac.il

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

Communicated by Omer Reingold.

Received 7 January 2015 / Revised 21 October 2015
Online publication 14 January 2016

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 efficiently 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 scheme. 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, Garg et al. (Symposium on theory of computing conference, STOC, pp 467–476, 2013) put forward the concept of $\mathsf{witness\ encryption}$, where the goal is to encrypt a message relative to a statement $x \in L$ for a language $L \in \mathsf{NP}$ 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. Garg et al. showed how to construct several cryptographic primitives from witness encryption and gave a candidate construction. One can show that computational secret-sharing implies witness encryption for the same language. Our main result is the converse: we give a construction of a computational secret-sharing scheme for *any* monotone function in NP assuming witness encryption for NP and one-way functions. As a consequence we get a completeness theorem for secret-sharing: compu-

[*]Research supported in part by a Grant from the Israel Science Foundation, the I-CORE Program of the Planning and Budgeting Committee, BSF and the Israeli Ministry of Science and Technology. Moni Naor is the incumbent of the Judith Kleeman Professorial Chair.

tational secret-sharing scheme for any *single* monotone NP-complete function implies a computational secret-sharing scheme for *every* monotone function in NP.

Keywords. Secret-sharing, Witness encryption, Obfuscation.

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 [2].

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 [8] and Shamir [32] 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 [22] 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 DNF (resp. CNF) formula size of the corresponding function. Benaloh and Leichter [7] 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 [25] 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 [6] 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]

[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.

Computational Secret-Sharing In the secret-sharing schemes considered above the security is guaranteed information theoretically, that is, 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 [24] presented a computational secret-sharing scheme for threshold access structures that is more efficient (in terms of the size of the shares) than the perfect secret-sharing schemes given by Blakley and Shamir [8,32]. In an unpublished work (mentioned in [2], see also Vinod et al. [34]), Yao showed an efficient computational secret-sharing scheme 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 [7] for polynomial-size monotone *formulas*). Yao's construction assumes the existence of pseudorandom generators, which can be constructed from any one-way function [20]. 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 answer the following question:

> Which access structures have efficient computational secret-sharing schemes, and what cryptographic assumptions are required for that?

Secret-Sharing for NP 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

[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 [31].

[5] Rudich raised it in private communication with the second author around 1990 and was not written to the best of our knowledge; some of Rudich's results can be found in Beimel's survey [2] and in Naor's presentation [29].

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 $\mathsf{NP} \neq \mathsf{coNP}$, 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 [21].

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. [15], where the goal is to encrypt a message relative to a statement $x \in L$ for a language $L \in \mathsf{NP}$ 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. Garg et al. showed how to construct several cryptographic primitives from witness encryption and gave a candidate construction.

A by-product of the proposed construction of Garg et al. was a construction of a computational secret-sharing scheme for a *specific* monotone NP-complete language. However, understanding whether one can use a secret-sharing scheme for any single (monotone) NP-complete language in order to achieve secret-sharing schemes for any language in mNP was an open problem. One of our main results is a positive answer to this question. Details follow.

Our Results In this paper, we construct a secret-sharing scheme for *every* mNP access structure assuming witness encryption for NP and one-way functions. In addition, we give two variants of a formal definition for secret-sharing for mNP access structures (indistinguishability and semantic security) and prove their equivalence.

Theorem 1.1. *Assuming witness encryption for* NP *and one-way functions, there is an efficient computational secret-sharing scheme for every* mNP *access structure.*

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,

[6]Moreover, it is possible to show that if $\mathsf{NP} \not\subseteq \mathsf{coAM}$, then there is no *statistical* secret-sharing scheme for the Hamiltonian access structure in which the sharing of the secret can be done efficiently [29].

[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.

negligibly close to 1), then our scheme from Theorem 1.1 actually gives a secret-sharing scheme for every monotone functions in MA.[8]

As a corollary, using the fact that a secret-sharing scheme for a language implies witness encryption for that language and using the completeness of witness encryption,[9] we obtain a completeness theorem for secret-sharing.

Corollary 1.2. (Completeness of Secret-Sharing) *Let L be a monotone language that is* NP*-complete (under Karp/Levin reductions) and assume that one-way functions exist. If there exists a computational secret-sharing scheme for the access structure defined by L, then there are computational secret-sharing schemes for* every mNP *access structure.*

1.1. *On Witness Encryption and Its Relation to Obfuscation*

Witness encryption was introduced by Garg et al. [15]. They gave a formal definition and showed how witness encryption can be combined with other cryptographic primitives to construct public-key encryption (with efficient key generation), identity-based encryption and attribute-based encryption. Lastly, Garg et al. presented a candidate construction of a witness encryption scheme which they assumed to be secure. In a more recent work, a new construction of a witness encryption scheme was proposed by Gentry, Lewko and Waters [17].

Shortly after the paper of Garg et al. [15] a candidate construction of indistinguishability obfuscation was proposed by Garg et al. [14]. An indistinguishability obfuscator is an algorithm that guarantees that if two circuits compute the same function, then their obfuscations are computationally indistinguishable. The notion of indistinguishability obfuscation was originally proposed in the seminal work of Barak et al. [3,4].

Recently, there have been two significant developments regarding indistinguishability obfuscation: first, candidate constructions for obfuscators for all polynomial-time programs were proposed [5,11,14,16,30] and second, intriguing applications of indistinguishability obfuscation when combined with other cryptographic primitives[10] have been demonstrated (see, e.g. [12,14,33]).

As shown by Garg et al. [14], indistinguishability obfuscation implies witness encryption for all NP, which, as we show in Theorem 1.1, implies secret-sharing for all mNP. In fact, using the completeness of witness encryption (see Footnote 9), even an indistinguishability obfuscator for $\mathsf{3CNF}$ formulas (for which there is a simple candidate construction [10]) implies witness encryption for all NP. Understanding whether witness encryption is strictly weaker than indistinguishability obfuscation is an important open problem.

A summary of the known relations between the above-mentioned objects can be found in “Appendix 3”.

[8] The class MA is defined similarly to the verifier-based definition of NP where the verifier is allowed to use randomness.

[9] Using standard Karp/Levin reductions between NP-complete languages, one can transform a witness encryption scheme for a single NP-complete language to a witness encryption scheme for any other language in NP.

[10] See [23] for a thorough discussion of the need in additional hardness assumptions on top of $i\mathcal{O}$.

1.2. *Other Related Work*

A different model of secret-sharing for mNP access structures was suggested by Vinod et al. [34]. 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.

Following this work, Komargodski and Zhandry [26] studied two extensions of secret-sharing for NP. In the first, which they call *distributed* secret-sharing, there is no trusted dealer at all, and instead the role of the dealer is distributed among the parties themselves. The second, which they call *functional* secret-sharing, incorporates some of the features of functional encryption into secret-sharing by providing more fine-grained access to the secret: Qualified subsets of parties do not learn the secret, but instead learn some function applied to the secret, with each set of parties potentially learning a *different* function. Komargodski and Zhandry [26] showed that both of these extensions are *equivalent* to other recent primitives.

1.3. *Main Idea*

Let Com be a perfectly binding commitment scheme. Let $M \in \mathsf{mNP}$ be an access structure on n parties $\mathcal{P} = \{\mathsf{p}_1, \ldots, \mathsf{p}_n\}$. For a sequence of commitments $\vec{\mathsf{c}} = \mathsf{c}_1, \ldots, \mathsf{c}_n$ and a sequence of (alleged) openings $\vec{r} = r_1, \ldots, r_n$, let $x = x_{\vec{\mathsf{c}},\vec{r}}$ be a string whose i^{th} bit is defined as

$$\forall i \in [n]: \quad x_i = \begin{cases} 1 & \text{if } r_i \neq \perp \text{ and } \mathsf{Com}(i, r_i) = \mathsf{c}_i, \\ 0 & \text{otherwise.} \end{cases}$$

We define a language M', related to M, consisting of all sequences of commitments $\vec{\mathsf{c}}$ for which there exists a sequence openings $\vec{r}$ and a witness w for $x_{\vec{\mathsf{c}},\vec{r}}$ being in M.

For the language M' denote by $(\mathsf{Encrypt}_{M'}, \mathsf{Decrypt}_{M'})$ the witness encryption scheme for M'. A secret-sharing scheme for the access structure M consists of a setup phase in which the dealer distributes secret shares to the parties. First, the dealer samples uniformly at random n openings $r_1, \ldots, r_n$. Then, the dealer computes a witness encryption ct of the message S with respect to the instance $(\mathsf{c}_1 = \mathsf{Com}(1, r_1), \ldots, \mathsf{c}_n = \mathsf{Com}(n, r_n))$ of the language M', namely $\mathsf{ct} = \mathsf{Encrypt}_{M'}((\mathsf{c}_1, \ldots, \mathsf{c}_n), S)$. Finally, the share of party p_i is set to be $\langle r_1, \mathsf{ct}\rangle$.

Clearly, if $\mathsf{Encrypt}_{M'}$ and 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, decrypt ct using the shares of the parties X and w. By the completeness of the witness encryption scheme, given a valid subset of parties X and a valid witness w the decryption will output the secret S.

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 X be such that $M(X) = 0$ and let D be an algorithm that extracts the secret given the shares of parties corresponding to 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 witness encryption ct' such that $\mathsf{ct}' = \mathsf{Encrypt}_{M'}((\mathsf{c}'_1, \ldots, \mathsf{c}'_n), S)$ 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 p_i to be $\langle r_i, \mathsf{ct}' \rangle$. 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 ct and ct' 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 there is no valid witness to decrypt ct' (since the commitment scheme is perfectly binding). Therefore, by the security of the witness encryption scheme, it is computationally hard to learn anything about the secret S from ct'. 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.

The above gives intuition for proving security in the non-uniform setting. To see this, we assume that there exists an X such that $M(X) = 0$ and the distinguisher D can extract the secret from the shares of X. Our security definition (see Sect. 3) is uniform and requires the distinguisher D to find such an X *and* extract the secret with noticeable probability. In the uniform case, we first run D to get X and must make sure that $M(X) = 0$. Otherwise, if $M(X) = 1$, in both cases (that $\mathsf{c}_1, \ldots, \mathsf{c}_n$ correspond to A_0 or to A_1) it is easy to extract the secret and thus we might be completely fooled. The problem is that M is a language in mNP and, in general, it could be hard to test whether $M(X) = 0$. We overcome this by sampling many subsets X and use D to estimate which one to use. For more information we refer to Sect. 4.1.

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 λ 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 $s \overset{\mathsf{R}}{\leftarrow} S$ to denote the operation of sampling an s uniformly at random from the set S. A function $\mathsf{neg}: \mathbb{N} \to \mathbb{R}$ is *negligible* if for every constant $c > 0$ there exists an integer N_c such that $\mathsf{neg}(\lambda) < \lambda^{-c}$ for all $\lambda > N_c$.

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 circuit 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. ([19]) 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. ([19, Theorem 2.2]) $\mathsf{mNP} = \mathsf{NP} \cap \mathsf{mono}$*, where* mono *is the set of all monotone functions.*

2.2. *Computational Indistinguishability*

Definition 2.3. Two sequences of random variables $X = \{X_\lambda\}_{\lambda \in \mathbb{N}}$ and $Y = \{Y_\lambda\}_{\lambda \in \mathbb{N}}$ are computationally indistinguishable if for every probabilistic polynomial-time algorithm A, there exists a negligible function $\mathsf{neg}(\cdot)$, such that for any $\lambda \in \mathbb{N}$ it holds that

$$\left|\Pr[A(1^\lambda, X_\lambda) = 1] - \Pr[A(1^\lambda, Y_\lambda) = 1]\right| \leq \mathsf{neg}(\lambda),$$

where the probabilities are over X_λ, Y_λ 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 [2] 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 subsets 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 [9] 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 [9] for more information.

2.4. *Witness Encryption*

Definition 2.5. (*Witness encryption* [15,17]) A $\mathsf{witness\ encryption}$ scheme for an NP language L (with a corresponding relation R) consists of the following two polynomial-time algorithms:

$\mathsf{Encrypt}(1^\lambda, x, M)$: Takes as input a security parameter 1^λ, a string x and a message M, and outputs a ciphertext ct.

$\mathsf{Decrypt}(\mathsf{ct}, w)$: Takes as input a ciphertext ct and a string w, and outputs a message M or the symbol $\perp$.

These algorithms satisfy the following two conditions:

1. *Completeness (Correctness)* For any security parameter λ, any $M \in \{0,1\}^*$ and any $x \in L$ such that $R(x, w)$ holds, we have that

$$\Pr[\mathsf{Decrypt}(\mathsf{Encrypt}(1^\lambda, x, M), w) = M] = 1.$$

2. *Soundness (Security)* For any probabilistic polynomial-time adversary A and any polynomial $p(\cdot)$, there exists a negligible function $\mathsf{neg}(\cdot)$, such that for any $\lambda \in \mathbb{N}$, any $x \notin L$ and any two equal-length messages M_1 and M_2 such that $|x|, |M_1| \leq p(\lambda)$, we have that

$$\left|\Pr[A(\mathsf{Encrypt}(1^\lambda, x, M_1)) = 1] - \Pr[A(\mathsf{Encrypt}(1^\lambda, x, M_2)) = 1]\right| \leq \mathsf{neg}(\lambda).$$

Remark. Our definition of Rudich secret-sharing (that is given in Sect. 3) is uniform. The most common definition of witness encryption in the literature is a non-uniform one (i.e. the security holds for *any* instance and pair of messages, and not only for ones that can be found efficiently by the adversary). To achieve our notion of security for Rudich secret-sharing it is enough to use a witness encryption scheme in which the messages and instance are chosen uniformly.

2.5. *Commitment Schemes*

In our construction we need a non-interactive commitment scheme such that commitments of different strings have disjoint support. Since the dealer in the setup phase of a secret-sharing scheme is not controlled by an adversary (i.e. it is honest), we can relax the foregoing requirement and use non-interactive commitment schemes that work in the CRS (common random string) model. Moreover, since the domain of input strings is small (it is of size $2n$), issues of non-uniformity can be ignored. Thus, we use the following definition:

Definition 2.6. (*Commitment scheme in the CRS model*) A polynomial-time computable function $\mathsf{Com}\colon \{0,1\}^n \times \{0,1\}^\lambda \times \{0,1\}^m \to \{0,1\}^*$, where $n = \mathsf{poly}(\lambda)$ is the length of the string to commit, λ is the length of the randomness, $m = \mathsf{poly}(\lambda)$ is the length of the CRS. We say that Com is a (non-interactive perfectly binding) commitment scheme in the CRS model if for any two inputs $x_1, x_2 \in \{0,1\}^n$ such that $x_1 \neq x_2$ it holds that:

1. *Computational Hiding* Let $\mathsf{crs} \leftarrow \{0,1\}^m$ be chosen uniformly at random. The random variables $\mathsf{Com}(x_1, \mathbf{U}_\lambda, \mathsf{crs})$ and $\mathsf{Com}(x_2, \mathbf{U}_\lambda, \mathsf{crs})$ are computationally indistinguishable (given crs).
2. *Perfect Binding* With all but negligible probability over the CRS the supports of the above random variables are disjoint.

Commitment schemes that satisfy the above definition, in the CRS model, can be constructed based on any pseudorandom generator [27] (which can be based on any one-way functions [20]). For simplicity, throughout the paper we ignore the CRS and simply write $\mathsf{Com}(\cdot,\cdot)$. We say that $\mathsf{Com}(x, r)$ is the $\mathsf{commitment}$ of the value x with the $\mathsf{opening}$ 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 schemes for functions in NP were considered in the past [2,15,34], 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. In Sect. 3.1 we give an alternative definition and show their equivalence.

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

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

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

   where the probability is over the internal randomness of the scheme and of RECON.
3. Indistinguishability of the Secret:
   For every pair of probabilistic polynomial-time algorithms (Samp, D) and every polynomial $p(\cdot)$, where $\mathsf{Samp}(1^\lambda, n)$ defines a distribution over pairs of secrets S_0, S_1 of the same length, a subset of parties $X \subseteq \mathcal{P}_n$ and auxiliary information σ, there exists a negligible function $\mathsf{neg}(\cdot)$ such that

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

   for every λ and n such that $n \leq p(\lambda)$, where the probability is over the internal randomness of the scheme, the internal randomness of D and the distribution $(S_0, S_1, X, \sigma) \leftarrow \mathsf{Samp}(1^\lambda, n)$.
   That is, for every pair of probabilistic polynomial-time algorithms (Samp, D) such that Samp chooses two secrets S_0, S_1 and a subset of parties $X \subseteq \mathcal{P}_n$, 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^λ, n, and σ 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 [18]), it turns out that the two definitions are equivalent.

Definition 3.2. (*Rudich secret-sharing—semantic security version*) Let $M : 2^{\mathcal{P}_n} \to \{0, 1\}$ be an mNP access structure with verifier V_M. A secret-sharing scheme $\mathcal{S}$ for M consists of a setup procedure SETUP and a reconstruction procedure 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 (Samp, D) and every polynomial $p(\cdot)$, where $\mathsf{Samp}(1^\lambda, n)$ defines a distribution over a secret S, a subset

of parties $X \subseteq \mathcal{P}_n$ and auxiliary information σ, there exists a negligible function $\mathsf{neg}(\cdot)$ such that for every efficiently computable function $f : \{0,1\}^* \to \{0,1\}^*$ there exists a probabilistic polynomial-time algorithm D' (called a *simulator*), such that

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

for every λ and n such that $n \le p(\lambda)$, where the probability is over the internal randomness of the scheme, the internal randomness of D and D', and the distribution $(S, X, \sigma) \leftarrow \mathsf{Samp}(1^\lambda, n)$.
That is, for every pair of probabilistic polynomial-time algorithms (Samp, D) such that Samp chooses a secret S and a subset of parties $X \subseteq \mathcal{P}_n$, 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. *Definitions 3.2 and 3.1 are equivalent.*

We defer the proof of Theorem 3.3 to “Appendix 1”.

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 shares of the previously chosen parties.

The security proof of our scheme (which is given in Sect. 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. [13]), 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 Witness Encryption

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 witness encryption for NP and one-way functions.

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

The SETUP Procedure:

Input: A secret S.

Let M' be the language as described above, and let $(\mathsf{Encrypt}_{M'}, \mathsf{Decrypt}_{M'})$ be a witness encryption for M' (see Definition 2.5).

```
1. For $i \in [n]$:
    (a) Sample uniformly at random an opening $r_i \in \{0,1\}^\lambda$.
    (b) Compute the commitment $\mathsf{c}_i = \mathsf{Com}(i, r_i)$.
2. Compute $\mathsf{ct} \leftarrow \mathsf{Encrypt}_{M'}((\mathsf{c}_1, \ldots, \mathsf{c}_n), S)$.
3. Set the share of party $\mathsf{p}_i$ to be $\Pi(S, i) = \langle r_i, \mathsf{ct} \rangle$.
```

The 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 $\mathsf{ct}$ be the witness encryption in the shares of $X$.
2. For any $i \in [n]$ let $r'_i = \begin{cases} r_i & \text{if } p_i \in X \\ \perp & \text{otherwise.} \end{cases}$
3. Output $\mathsf{Decrypt}_{M'}(\mathsf{ct}, (r'_1, \ldots, r'_n, w))$.
```

Fig. 1. Rudich secret-sharing scheme for NP.

Theorem 1.1 (Restated) *Assuming witness encryption for* NP *and one-way functions, there is an efficient computational secret-sharing scheme for every* mNP *access structure.*

Let $\mathcal{P} = \{\mathsf{p}_1, \ldots, \mathsf{p}_n\}$ be a set of n parties and let $M : 2^{\mathcal{P}} \rightarrow \{0, 1\}$ be an mNP access structure. We view M either as a function or as a language. For a language L in NP let $(\mathsf{Encrypt}_L, \mathsf{Decrypt}_L)$ be a witness encryption scheme and let $\mathsf{Com} : [2n] \times \{0, 1\}^\lambda \rightarrow \{0, 1\}^{q(\lambda)}$ be a commitment scheme, where $q(\cdot)$ is a polynomial.

The Scheme For any sequence of strings $\vec{\mathsf{c}} = \mathsf{c}_1, \ldots, \mathsf{c}_n$, where $\mathsf{c}_i \in \{0, 1\}^{q(\lambda)}$, and $\vec{r} = r_1, \ldots, r_n$, where $r_i \in \{0, 1\}^\lambda \cup \{\perp\}$, let $x = x_{\vec{\mathsf{c}},\vec{r}} \in \{0, 1\}^n$ be a string whose i^{th} bit is defined as

$$x_i = \begin{cases} 1 & \text{if } r_i \neq \perp \text{ and } \mathsf{Com}(i, r_i) = \mathsf{c}_i, \\ 0 & \text{otherwise.} \end{cases}$$

We define a language M', related to M, consisting of all sequences of strings $\vec{\mathsf{c}}$ for which there exists a sequence $\vec{r}$ and a witness w for $x_{\vec{\mathsf{c}},\vec{r}}$ being in M.

For every $i \in [n]$, the *share* of party p_i is composed of two components: (1) $r_i \in \{0, 1\}^\lambda$ - an opening of a commitment to the value i, and (2) a witness encryption ct. The witness encryption encrypts the secret S with respect to the commitments of all parties $\{\mathsf{c}_i = \mathsf{Com}(i, r_i)\}_{i \in [n]}$. To reconstruct the secret given a subset of parties X we simply decrypt ct given the corresponding openings of X and the witness w that indeed $M(X) = 1$. The secret-sharing scheme is formally described in Fig. 1.

Observe that if the witness encryption scheme and Com are both efficient, then the scheme is efficient (i.e. SETUP and RECON are probabilistic polynomial-time algorithms). SETUP generates n commitments and a witness encryption of polynomial size. RECON only decrypts this witness encryption.

Completeness The next lemma states 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, it is possible to successfully reconstruct the secret.

Lemma 4.1. *Let $M \in \mathsf{NP}$ be an mNP access structure. Let $\mathcal{S} = \mathcal{S}_M$ be the scheme from Fig. 1 instantiated with M. For every subset of parties $X \subseteq \mathcal{P}$ such that $M(X) = 1$ and any valid witness w it holds that*

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

Proof. Recall the definition of the algorithm RECON from Fig. 1: 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 a witness encryption ct. RECON decrypts ct given the openings of parties in X and the witness w.

By the completeness of the witness encryption scheme, the output of the decryption procedure on ct, given a valid X and a valid witness, is S (with probability 1). □

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

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

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 2”.

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* $\varepsilon = \varepsilon(\lambda)$ *and a pair of probabilistic polynomial-time algorithms* (Samp, D) *such that for* $(S_0, S_1, X) \leftarrow \mathsf{Samp}(1^\lambda, n)$ *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* λ/ε *such that for sufficiently large n*

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

The proof of Lemma 4.3 appears in Sect. 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, (Samp, D) be a pair of algorithms and $\varepsilon = \varepsilon(\lambda) > 0$, as in Lemma 4.3. We are given a list of (unopened) string commitments $\mathsf{c}_1, \ldots, \mathsf{c}_n \in \{\mathsf{Com}(z_i, r)\}_{r \in \{0,1\}^\lambda}$, 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 Samp and D) with non-negligible probability (that is related to ε). Recall that 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 $(S_0, S_1, X) \leftarrow \mathsf{Samp}(1^\lambda, n)$ we have that

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

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 witness encryption ct'_b of the message S_b with respect to the instance $\mathsf{Com}(1, r_1), \ldots, \mathsf{Com}(k, r_k), \mathsf{c}_{k+1}, \ldots, \mathsf{c}_n$ of M' (see Fig. 1) and sets for every $i \in [n]$ the share of party p_i to be $\Pi'(S_b, i) = \langle r_i, \mathsf{ct}'_b\rangle$. Finally, $\mathsf{D_{ver}}$ emulates the execution of D on the set of shares of X ($\Pi'(S_b, X)$). 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 $Z = A_1$).

A naive implementation of D' is to run 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 $1 - 1/\mathsf{poly}(\lambda)$) over Samp it holds that $M(X) = 1$, in which we do not have any guarantee on D. Hence, simply running 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), Samp generates an X such that $M(X) = 0$ with (non-negligible) probability at least ε. By this observation, notice that by running Samp for $\Theta(\lambda/\varepsilon)$ iterations we are assured that with very high probability (specifically, $1 - \mathsf{neg}(\lambda)$) 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 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. On 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 $\Theta(\lambda/\varepsilon)$ 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 $\Theta(\lambda/\varepsilon)$ iterations such that in each iteration it runs $\mathsf{Samp}(1^\lambda, n)$ 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 Fig. 2.

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

Lemma 4.3 (Restated) *Let* $\mathsf{c}_1, \ldots, \mathsf{c}_n \in \{\mathsf{Com}(z_i, r)\}_{r\in\{0,1\}^\lambda}$ *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*

```
The algorithm $D'$

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

  1. Do the following for $T = \lambda/\varepsilon$ times:
     (a) $S_0, S_1, X \leftarrow \mathsf{Samp}(1^\lambda, n)$.
     (b) Run $\mathsf{resB} \leftarrow \mathsf{B}(S_0, S_1, X)$.
     (c) If $\mathsf{resB} = 1$:
         i. Run $\mathsf{resD} \leftarrow \mathsf{D_{ver}}(\mathsf{c}_1,\dots,\mathsf{c}_n, S_0, S_1, X)$.
         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 $T_{\mathsf{B}} = 4\lambda/\varepsilon$ times:
     (a) $q_0 \leftarrow q_0 + \mathsf{D_{ver}}(\mathsf{Com}(1, \mathbf{U}_n),\dots,\mathsf{Com}(n, \mathbf{U}_n), S_0, S_1, X)$.
     (b) $q_1 \leftarrow q_1 + \mathsf{D_{ver}}(\mathsf{Com}(n+1, \mathbf{U}_n),\dots,\mathsf{Com}(2n, \mathbf{U}_n), S_0, S_1, X)$.
  2. If $|q_0 - q_1| > n$, output 1.
  3. Output 0.

The sub-procedure $\mathsf{D_{ver}}$

Input: A sequence of commitments $\mathsf{c}_1,\dots,\mathsf{c}_n$, 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 $r_i \overset{\mathsf{R}}{\leftarrow} \mathbf{U}_n$ 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 $\mathsf{ct}'_b \leftarrow \mathsf{Encrypt}_{M'}((\mathsf{c}'_1,\dots,\mathsf{c}'_n), S_b)$.
  4. For $i \in [n]$ let the new share of party $\mathsf{p}_i$ be $\Pi'(S_b, i) = \langle r_i, \mathsf{ct}'_b\rangle$.
  5. Return 1 if $D(S_0, S_1, \Pi'(S_b, X)) = b$ and 0 otherwise.
```

Fig. 2. The description of the algorithm D'.

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

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

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

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

Proof. By the definition of $\mathsf{D}_{\mathsf{ver}}$ (see Fig. 2) 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 $b \overset{\mathsf{R}}{\leftarrow} \{0, 1\}$. Since b is chosen uniformly at random from $\{0, 1\}$, it is enough to show that

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

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

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

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 $\left(\mathsf{c}'_1, \ldots, \mathsf{c}'_n\right)$. Hence, for any $b \in \{0, 1\}$ it holds that $\Pi'(S_b, X)$ is identically distributed as $\Pi(S_b, X)$. Hence,

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

as required. □

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

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

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 p_i denoted by $\Pi'(S_b, i)$ consists of the pair $\langle r_i^b, \mathsf{ct}'_b)\rangle$ where r_i^b is chosen uniformly at random from $\mathbf{U}_n$. To prove the claim we show that ct'_0 and ct'_1 are computationally indistinguishable.

To this end, we show that if $Z = A_1$ and $M(X) = 0$, then there is *no* witness attesting to the fact that $\mathsf{c}'_1, \ldots, \mathsf{c}'_n$ is in M'. Fix $X \subseteq \mathcal{P}$ such that $M(X) = 0$ and let $(\{r'_i\}_{i \in [n]}, w) \in (\{0, 1\}^\lambda)^n \times \{0, 1\}^*$ be a possible witness. Let X' be the set of parties that correspond to the r'_i's for which $r'_i \neq \perp$.

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 witness is invalid since for every i such that $\mathsf{p}_i \notin X$ the commitment c_i 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 c_i and the value i is *invalid*, i.e. any opening r'_i will fail the test $\mathsf{c}_i \overset{?}{=} \mathsf{Com}(i, r'_i)$.

Otherwise, if $X' \subseteq X$, then since M is monotone and $M(X) = 0$ it holds that $M(X') = 0$. Therefore, the witness is invalid for X'.

In conclusion, since $M'(\mathsf{c}_1, \ldots, \mathsf{c}_n) = 0$, the witness encryptions of S_0 and S_1 are computationally indistinguishable from one another (see Definition 2.5) and the claim follows. $\square$

Next, we continue with two claims connecting $\mathsf{D}_{\mathsf{ver}}$ and B. Before we state these claims, we introduce a useful notation regarding the bias of the procedure $\mathsf{D}_{\mathsf{ver}}$. We denote by $\mathsf{bias}(S_0, S_1, X)$ the advantage of $\mathsf{D}_{\mathsf{ver}}$ 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

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

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

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

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

Proof. Recall that B runs for T_{B} *independent* iterations such that in each iteration it executes $\mathsf{D}_{\mathsf{ver}}$ 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 $\mathsf{D}_{\mathsf{ver}}(\mathsf{Com}(1, \mathbf{U}_n), \ldots, \mathsf{Com}(n, \mathbf{U}_n), S_0, S_1, X) = 1$. Similarly, denote by I_1^i an indicator random variable that takes the value 1 if and only if in the i-th iteration $\mathsf{D}_{\mathsf{ver}}(\mathsf{Com}(n+1, \mathbf{U}_n), \ldots, \mathsf{Com}(2n, \mathbf{U}_n), S_0, S_1, X) = 1$. When 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 [1, § 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 B outputs 1 if and only if $|q_0 - q_1| > n$. Plugging in $T_{\mathsf{B}} = 4\lambda/\varepsilon$ both parts of the claim follow. $\square$

In Claim 4.6 we proved that B is a good estimator for the bias of $\mathsf{D}_{\mathsf{ver}}$. That is, we showed that if $\mathsf{D}_{\mathsf{ver}}$ is very biased, then B is 1 (with high probability) and vice versa (i.e. that if $\mathsf{D}_{\mathsf{ver}}$ is unbiased, then B is most likely to be 0). Denote by 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 BAD happens in any iteration of D' is negligible.

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

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

Proof. Since the T iterations are independent and implemented identically it holds that

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

Observe that

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

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

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. bias is large), with Claim 4.6 that assures that if the bias is large, then B almost surely outputs 1.

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

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

Proof. Let $(S_0, S_1, X) \leftarrow \mathsf{Samp}(1^\lambda, n)$. By the definition of 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

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

Using Claims 4.4 and 4.5 we get that

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

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 Samp). Combining this fact with the fact that D' makes $T = \lambda/\varepsilon$ iterations of B and $\Pr\left[\mathsf{B}(S_0, S_1, X) = 1 \mid M(X) = 0\right] \geq 1 - \mathsf{neg}(\lambda)$ (by Claim 4.8), we get that D' reaches Step 2 with negligible probability. In other words, with probability $1 - \mathsf{neg}(\lambda)$ there is an iteration in which X is chosen such that $M(X) = 0$ and 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 BAD does not happen. That is, in every iteration either B outputs 0 or bias is larger than $\varepsilon/10$. Recall that D' ignores all the iteration in which B outputs 0. Moreover, we assumed that there is an iteration in which B outputs 1. In that iteration, it must be the case that the bias is larger than $\varepsilon/10$ which completes the proof. □

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 candidate 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 [31]). Our construction only requires witness encryption scheme for NP.

We conclude with several open problems:

- Is there a secret-sharing scheme for mNP that relies only on standard falsifiable hardness assumptions [28]?
- 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 Sect. 3.2 for a discussion). Under a stronger assumption, i.e. extractable witness encryption (in which if an algorithm is able to decrypt a ciphertext, then it is possible to extract a witness), Zvika Brakerski observed that our construction is secure against adaptive adversaries as well.
- Show a completeness theorem (similarly to Corollary 1.2) for secret-sharing schemes that are also secure against *adaptive* adversaries, as defined in Sect. 3.2.

Acknowledgements

We are grateful to Amit Sahai for suggesting to base our construction on witness encryption. 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. We thank the anonymous referees for many helpful remarks.

Appendix 1: 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 (Samp, D) and a non-negligible ε such that for $(S_0, S_1, X, \sigma) \leftarrow \mathsf{Samp}(1^n)$ it holds that

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

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

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

Plugging in Eq. (2) we get that

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

Assume that Samp generates secrets in $[2^t]$ for some $t > 0$. Let $\mathcal{F} = \{f_i : [2^t] \to \{0, 1\} \mid i \in [t] \wedge \forall x \in [2^t] : \ f_i(x) = \mathsf{bin}(x)_i\}$ 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 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 σ. Then, Samp' chooses a bit $b \in \{0, 1\}$ uniformly at random and outputs (S_b, X, σ'), 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 σ'. 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 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 $f \in \mathcal{F}'$:

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

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 (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

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

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

$$\begin{aligned}&|\Pr[M(X) = 0 \wedge D(1^n, \Pi(S, X), \sigma) = f(S)]\\&\quad - \Pr[M(X) = 0 \wedge D'(1^n, X, \sigma) = f(S)]| \geq \varepsilon.\end{aligned}$$

Plugging in the definition of D' we have that

$$\begin{aligned}&|\Pr[M(X) = 0 \wedge D(1^n, \Pi(S, X), \sigma) = f(S)]\\&\quad - \Pr[M(X) = 0 \wedge D(1^n, \Pi(0, X), \sigma) = f(S)]| \geq \varepsilon.\end{aligned}$$

Next, we define a pair of algorithms (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 Samp'' simply runs Samp to get (S, X, σ) 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

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

which contradicts the indistinguishability assumption. □

Appendix 2: Proof of Lemma 4.2

In this section we prove the following lemma.

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

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

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

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

For $\sigma \in [2n]$ let c_σ 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 $\mathcal{C}^{(i)}$,

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

At this point it follows that there exists D' that distinguishes between c_{n-i+1} and 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. □

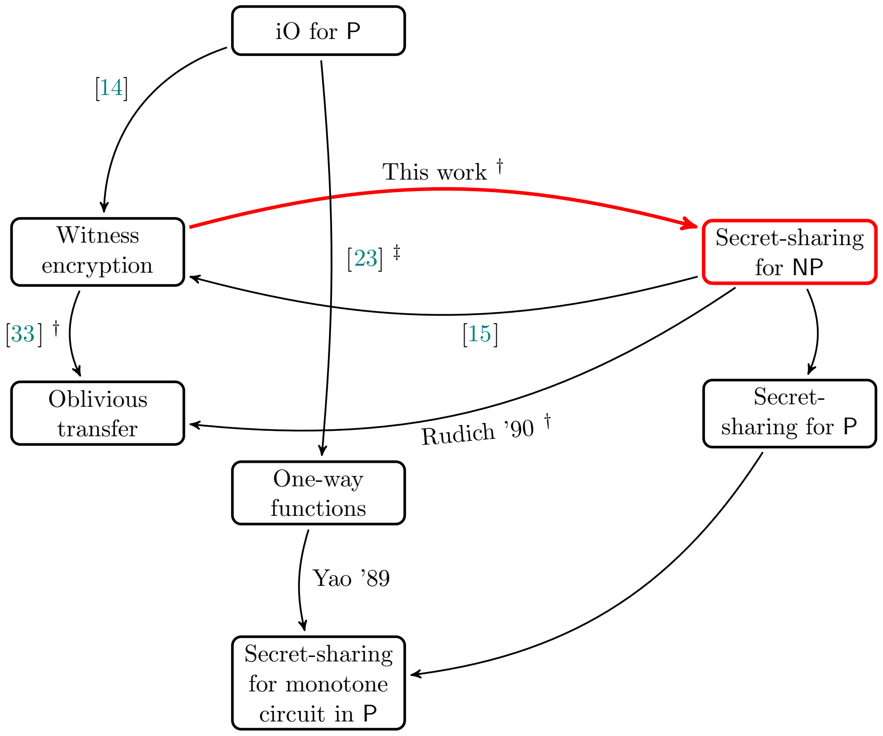

Fig. 3. Secret-sharing Zoo. A *dagger symbol* mark on a *line* denotes the fact that the reduction between the primitives relies also on the existence of one-way functions. *iO* stands for indistinguishability obfuscation. A *double dagger symbol* mark on the line from iO for P to one-way functions means that the reduction assumes a worst-case complexity assumption, namely, that $\mathsf{NP} \not\subseteq \mathsf{io\text{-}BPP}$ (see [23] for more information). Yao '89 and Rudich '90 are unpublished.

Appendix 3: Secret-Sharing Zoo

A summary of the known relations between secret-sharing and other objects (Fig. 3).

References

[1] N. Alon, J. Spencer, *The Probabilistic Method (3rd ed.)* (Wiley, 2008)

[2] A. Beimel, Secret-sharing schemes: a survey, in *3rd International Workshop, IWCC* (2011), pp. 11–46

[3] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. P. Vadhan, K. Yang, On the (im)possibility of obfuscating programs, in *CRYPTO*. Lecture Notes in Computer Science, vol. 2139 (Springer, 2001), pp. 1–18

[4] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. P. Vadhan, K. Yang, On the (im)possibility of obfuscating programs. *J. ACM* **59**(2), 6 (2012). Preliminary version appeared in CRYPTO 2001

[5] B. Barak, S. Garg, Y. T. Kalai, O. Paneth, A. Sahai, Protecting obfuscation against algebraic attacks, in *33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, EYROCRYPTO* (2014), pp. 221–238

[6] A. Beimel, Y. Ishai, On the power of nonlinear secrect-sharing. *SIAM J. Discrete Math.* **19**(1), 258–280 (2005)

[7] J. C. Benaloh, J. Leichter, Generalized secret sharing and monotone functions, in *8th Annual International Cryptology Conference, CRYPTO* (1988), pp. 27–35

[8] G. R. Blakley, Safeguarding cryptographic keys. *Proc. AFIPS Natl. Comput. Conf.* **22**, 313–317 (1979)

[9] M. Bellare, P. Rogaway, Robust computational secret sharing and a unified account of classical secret-sharing goals, in *ACM Conference on Computer and Communications Security* (ACM, 2007), pp. 172–184

[10] Z. Brakerski, G. N. Rothblum, Black-box obfuscation for d-CNFs, in *Innovations in Theoretical Computer Science, ITCS* (2014), pp. 235–250

[11] Z. Brakerski, G. N. Rothblum, Virtual black-box obfuscation for all circuits via generic graded encoding, in *11th Theory of Cryptography Conference, TCC* (2014), pp. 1–25

[12] D. Boneh, M. Zhandry, Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation, in *34th Annual Cryptology Conference, CRYPTO* (2014), pp. 480–499

[13] C. Dwork, M. Naor, O. Reingold, L. J. Stockmeyer, Magic functions. *J. ACM* **50**(6), 852–921 (2003)

[14] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, B. Waters, Candidate indistinguishability obfuscation and functional encryption for all circuits, in *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS* (2013), pp. 40–49

[15] S. Garg, C. Gentry, A. Sahai, B. Waters, Witness encryption and its applications, in *Symposium on Theory of Computing Conference, STOC* (2013), pp. 467–476

[16] C. Gentry, A. B. Lewko, A. Sahai, B. Waters, Indistinguishability obfuscation from the multilinear subgroup elimination assumption. *IACR Cryptol. ePrint Arch.* **2014**, 309 (2014)

[17] C. Gentry, A. B. Lewko, B. Waters, Witness encryption from instance independent assumptions, in *34th Annual Cryptology Conference, CRYPTO* (2014), pp. 426–443

[18] S. Goldwasser, S. Micali, Probabilistic encryption. *J. Comput. Syst. Sci.* **28**(2), 270–299 (1984)

[19] M. Grigni, M. Sipser, Monotone complexity, in *LMS Workshop on Boolean Function Complexity* (1992), pp. 57–75

[20] J. Håstad, R. Impagliazzo, L. A. Levin, M. Luby, A pseudorandom generator from any one-way function. *SIAM J. Comput.* **28**(4), 1364–1396 (1999)

[21] R. Impagliazzo, A personal view of average-case complexity, in *10th Annual Structure in Complexity Theory Conference* (1995), pp. 134–147

[22] M. Ito, A. Saito, T. Nishizeki, Multiple assignment scheme for sharing secret. *J. Cryptol.* **6**(1), 15–20 (1993)

[23] I. Komargodski, T. Moran, M. Naor, R. Pass, A. Rosen, E. Yogev, One-way functions and (im)perfect obfuscation, in *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS* (2014), pp. 374–383

[24] H. Krawczyk, Secret sharing made short, in *13th Annual International Cryptology Conference, CRYPTO* (1993), pp. 136–146

[25] M. Karchmer, A. Wigderson, On span programs, in *8th Annual Structure in Complexity Theory Conference* (1993), pp. 102–111

[26] I. Komargodski, M. Zhandry, Cutting-edge cryptography through the lens of secret sharing. *IACR Cryptol. ePrint Arch.* **2015**, 735 (2015). To appear in TCC 2016-A

[27] M. Naor, Bit commitment using pseudorandomness. *J. Cryptol.* **4**(2), 151–158 (1991)

[28] M. Naor, On cryptographic assumptions and challenges, in *23rd Annual International Cryptology Conference, CRYPTO* (2003), pp. 96–109

[29] M. Naor, Secret sharing for access structures beyond P (2006). Slides: http://www.wisdom.weizmann.ac.il/~naor/PAPERS/minicrypt.html

[30] R. Pass, K. Seth, S. Telang, Indistinguishability obfuscation from semantically-secure multilinear encodings, in *34th Annual Cryptology Conference, CRYPTO* (2014), pp. 500–517

[31] A. 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.* **31**, 354–357 (1985)

[32] A. Shamir, How to share a secret. *Commun. ACM* **22**(11), 612–613 (1979)

[33] A. Sahai, B. Waters, How to use indistinguishability obfuscation: deniable encryption, and more, in *Symposium on Theory of Computing, STOC* (2014), pp. 475–484

[34] V. Vinod, A. Narayanan, K. Srinathan, C. P. Rangan, K. Kim, On the power of computational secret sharing, in *4th International Conference on Cryptology in India, INDOCRYPT* (2003), pp. 162–176