Cutting-Edge Cryptography Through the Lens of Secret Sharing

Ilan Komargodski[1(✉)] and Mark Zhandry[2]

[1] Weizmann Institute of Science, Rehovot, Israel
ilan.komargodski@weizmann.ac.il
[2] MIT, Cambridge, USA
mzhandry@gmail.com

Abstract. Secret sharing is a mechanism by which a trusted dealer holding a secret "splits" the secret into many "shares" and distributes the shares to a collection of parties. Associated with the sharing is a monotone access structure, that specifies which parties are "qualified" and which are not: any qualified subset of parties can (efficiently) reconstruct the secret, but no unqualified subset can learn anything about the secret. In the most general form of secret sharing, the access structure can be any monotone NP language.

In this work, we consider two very natural extensions of secret sharing. In the first, which we call *distributed* secret sharing, there is no trusted dealer at all, and instead the role of the dealer is distributed amongst the parties themselves. Distributed secret sharing can be thought of as combining the features of multiparty non-interactive key exchange and standard secret sharing, and may be useful in settings where the secret is so sensitive that no one individual dealer can be trusted with the secret. Our second notion is called *functional* secret sharing, which 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.

Our main result is that both of the extensions above are *equivalent* to several recent cutting-edge primitives. In particular, general-purpose distributed secret sharing is equivalent to witness PRFs, and general-purpose functional secret sharing is equivalent to indistinguishability obfuscation. Thus, our work shows that it is possible to view some of the recent developments in cryptography through a secret sharing lens, yielding new insights about both these cutting-edge primitives and secret sharing.

I. Komargodski—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.

M. Zhandry—Supported by NSF.

E. Kushilevitz and T. Malkin (Eds.): TCC 2016-A, Part II, LNCS 9563, pp. 449–479, 2016.
DOI: 10.1007/978-3-662-49099-0_17

1 Introduction

Secret sharing is a mechanism by which a trusted dealer holding a secret "splits" the secret into many "shares" and distributes the shares to a collections of parties. Associated with the sharing is a monotone access structure, that specifies which parties are "qualified" and which are not: any qualified subset of parties can (efficiently) reconstruct the secret, but no unqualified subset can learn anything about the secret.[1] The first secret sharing schemes, due to Shamir [33] and Blakley [7], were for the threshold access structure, where the subsets that can reconstruct the secret are all the sets whose cardinality is at least a certain threshold. Such secret sharing schemes provide a digital analog of the "two-man rule", and are useful for splitting a sensitive key among several individuals so that no single individual knows the key. Secret sharing schemes, even for the simple threshold access structure, have found numerous applications in computer science (see [4] for a thorough survey).[2]

Since their introduction, it has been a major open problem to determine which access structures can secret sharing be realized for. Benaloh and Leichter [6] constructed a secret sharing scheme for any access structure that can be computed by a monotone formula. This result was generalized and improved by Karchmer and Wigderson [23] for access structures that can be computed by a monotone span program. In an unpublished work, Andrew Yao constructed a secret sharing scheme for any access structure that can be computed by a *monotone* circuit (see [4,28]), assuming any one-way function. Recently, Komargodski, Naor and Yogev [25] constructed secret sharing schemes for all of monotone NP (denoted mNP),[3] assuming one-way functions and a recent new primitive called *witness encryption* [18].[4] Monotone NP is essentially the largest class of access structures that we can hope for: if we cannot even efficiently identify a qualified set, we cannot hope to have qualified sets reconstruct the secret.

In this work we take secret sharing even further, by pursuing two very natural directions. First, we ask if the trusted dealer is required, or whether it is possible to *distribute* the role of the dealer amongst the parties themselves. Second, we ask if we can provide more fine-grained access mechanism to the shared secret, whereby qualified sets of parties only learn some function of the secret, each set of parties learning a possibly different function. Surprisingly, in both cases we

[1] In secret sharing, we always restrict our attention to *monotone* access structures, where a superset of a qualified set must be qualified. This is necessary because, if a set of parties contains a qualified subset, they can always "pretend" to be the smaller subset, discard the shares outside that subset, and reconstruct the secret.

[2] Most of the literature on secret sharing treats it as an information-theoretic primitive and insists on perfect security. In this work we consider the computational analog in which we only require security against computationally bounded adversaries. The survey of Beimel [4] discusses extensively both notions.

[3] For access structures in mNP, a qualified set of parties needs to know an NP witness that they are qualified.

[4] We note that the schemes of [6,23] are unconditionally secure, while the schemes of Yao and [25] are only secure against adversaries that run in polynomial-time.

show equivalences between these natural extensions of secret sharing and several cutting-edge cryptographic primitives that have recently been developed.

Distributed Secret Sharing. The usefulness of secret sharing schemes, as defined above, is limited to settings in which there exists a *trusted* dealer who knows the secret. What if we do not want any one individual to know the secret outright? What if our secret is so sensitive that we cannot afford anybody to know it? In this paper, we study the necessity of the trusted dealer in the setting of secret sharing and ask the question:

Is it possible to secret share a secret without anybody knowing it?

To address this question, we introduce the concept of *distributed* secret sharing schemes. Specifically, given an access structure, each party can generate for itself a public share (which is published) and a secret share (which is kept private). Then, there is a string S such that every qualified subset of parties can compute S (using their private shares and all public shares), whereas for every unqualified subset the secret S remains hidden.[5] Similarly to standard secret sharing schemes for mNP, for an access structure M in mNP, a qualified subset X should also provide a witness for the statements $X \in M$. Intuitively, one can view distributed secret sharing schemes as a hybrid of secret sharing schemes and non-interactive key-exchange: Indeed, non-interactive key-exchange is exactly the special case where M is set to be the threshold access structure with threshold $t = 1$.

In this paper we construct and explore distributed secret sharing schemes. Our main result is that distributed secret sharing schemes for access structures in mNP are *equivalent* to witness pseudorandom functions (witness PRFs) for NP. A witness PRF for a language $L \in \mathsf{NP}$ is a function F such that anyone with a valid witness that $x \in L$ can compute $F(x)$ without the secret key, but for all $x \notin L$, $F(x)$ is computationally hidden to anybody that does not know the secret key. Witness PRFs were recently introduced by Zhandry [34] and shown to be very useful in constructing several important cryptographic primitives (including non-interactive multi-party key exchange without setup) that were previously only known to exist assuming seemingly much stronger assumptions.

In addition, we explore the possibility of distributed secret sharing for restricted classes of access structures based on weaker assumptions. To start, we consider the possibility of *information-theoretic* security for distributed secret sharing scheme (that is, security against unbounded adversaries). We show that such information-theoretic security is typically impossible: we prove that a distributed secret sharing scheme for any non-trivial access structure implies the existence of one-way functions.[6]

[5] We note that we do not assume secure point-to-point channels, a standard PKI or additional rounds of interaction (beyond publishing a public key) between the parties. With any of these assumptions the problem can be reduced to standard secret sharing.

[6] We call an access structure M *trivial* if M is empty or if there exists a subset of parties $X \in M$ which is contained in any qualified set. For trivial access structures, we show that there is a simple perfectly-secure distributed secret sharing scheme.

Next, we present a distributed secret sharing scheme for the threshold access structure, and prove its security based on the multilinear decisional Diffie-Hellman (MDDH) assumption. As an interesting application, we show that distributed secret sharing schemes for threshold access structures imply *constrained PRFs* that can be constrained to a Hamming ball around an arbitrary point and are secure for adversaries that obtain a single constrained key. Even though it is known that the MDDH assumption implies constrained PRFs for all circuits which are secure with respect to arbitrary collusions [10], our transformation is generic and applies to any threshold distributed secret sharing scheme, which perhaps can be based on simpler assumptions than multilinear maps.

Functional Secret Sharing. Traditional secret sharing schemes offer an all-or-nothing guarantee when reconstructing a shared secret — a qualified subset of parties can learn the entire secret, while unqualified subsets learn nothing about the secret. For many applications, especially in a distributed setting common to secret sharing, this notion is insufficient. Concretely, standard secret sharing schemes will not help in scenarios in which a dealer wants to share a secret such that every qualified subset of parties will learn a specific function of the secret (and nothing else). For example, a dealer holding a secret S, may want to distribute it such that any qualified subset X will be able to learn only the inner product of X and S, while making sure S remains computationally hidden for unqualified subsets.

A related issue has appeared in the context of encryption schemes, giving rise to the concept of *functional encryption* and a very fruitful line of work (see e.g., [8,30]). We study whether secret sharing schemes can be extended in an analogous way to support such functionalities: Given an efficiently computable two-input function F (that can be thought of as a family of functions indexed by the first input), we ask the question:

Is it possible to secret share a secret S such that any qualified subset of parties X can compute only F(X,S), but for unqualified subsets, S will be computationally hidden?

To study this question, we introduce the concept of *functional* secret sharing schemes. Informally, such a scheme allows to secret share a secret S with respect to a function F and an access structure M, such that any qualified subset of parties X can pool their shares together and compute $F(X, S)$. Security is formalized by requiring that for any function F, any subset of parties X and any two secrets S_0 and S_1, as long as either $M(X) = 0$ or $F(X', S_0) = F(X', S_1)$ for any $X' \subseteq X$, secret shares corresponding to F, X and S_0 cannot be distinguished from secret shares corresponding to F, X and S_1. Notice that the condition that $F(X', S_0) = F(X', S_1)$ for any $X' \subseteq X$ in the case that $M(X) = 1$ is necessary, as otherwise, by evaluating $F(X', S_b)$ an adversary can distinguish between the case that $b = 0$ and $b = 1$.

Our main result is that functional secret sharing schemes for access structures in mNP and functions in P are *equivalent* to indistinguishability obfuscation (iO)

for P.[7] An indistinguishability obfuscator [3,17] guarantees that if two circuits compute the same function, then their obfuscated version are computationally indistinguishable. This primitive was introduced by Barak et al. [3] and later proven to be extremely useful for construction of cryptographic primitives some of which were unknown before (see e.g., [11,17,31]). To complement this, several candidate constructions of indistinguishability obfuscators were recently proposed [1,2,13,17,19,29].

Note that when the function F is defined to be the identity function over its second input parameter (i.e., $F(\cdot, S) = S$) we get the standard definition of secret sharing for mNP of [25]. Moreover, when the access structure is the set of all subsets, the secret S is a description of a function and F is the universal circuit (i.e., $F(X, S) = S(X)$), we obtain a definition of a *function secret sharing scheme*. In such a scheme, the goal is to split a function (and not a secret) into shares that hide the function under some conditions. Our construction gives a way to split a function F into shares such that any subset of parties X can compute $F(X')$ for every $X' \subseteq X$ and "nothing" else. We note that other forms of function secret sharing have been studied in the literature (cf. [5,12,15,32]). However, our notion is quite different from (and incomparable to) these other notions. In particular, our notion is the first to allow for fine-grained access control to the secret by guaranteeing that any qualified set learns a *possibly different* function of the secret. Moreover, previous notions were mostly studied in the context of threshold access structures, only with very specific function classes or insisted on schemes with additional properties.[8]

Conclusions. Recent advances in cryptography, including the first constructions of multilinear maps [16] and obfuscation [17], have lead to the development of many incredible new cryptographic objects. Applications include functional encryption, witness encryption, witness PRFs, deniable encryption, multi-party computation in very few rounds, traitor-tracing schemes with very short messages, and many more. Our work can thus be seen as establishing a close connection between several of these advanced cryptographic capabilities and types of secret sharing, which at first appear totally unrelated. The known relationships, including our work, are depicted and summarized in Fig. 1. Our hope is that the connections we develop can help shed light on the relationships between advanced primitives, or between types of secret sharing: which are equivalent, why do some tasks appear difficult, and so on.

For example, our results indicate why witness PRFs, which are closely related to witness encryption, may be the "right" primitive for building non-interactive

[7] To show that iO implies functional secret sharing schemes, we also assume the existence of one-way functions. By a result of [25] we can actually only assume iO and NP $\not\subseteq$ io-BPP. Moreover, we note that in this paper we assume functions are represented as circuits, so we actually work with functions in P/poly (and not P).

[8] For example, the functional secret sharing notion of [12] is similar to ours but requires an additional homomorphic property for the reconstruction procedure. Our scheme does not have this extra property, however, our construction relies on iO while their construction relies on subexponentially-secure iO.

key exchange, and why witness encryption may be insufficient. Indeed, distributed secret sharing essentially combines the features of secret sharing for mNP (which is equivalent to witness encryption [25]) with non-interactive key exchange. If these non-interactive key exchange features could be obtained from witness encryption, then perhaps witness encryption could also imply witness PRFs. In addition, at first it may not be obvious what is the relationship between functional secret sharing and distributed secret sharing. Our results and the simple observation that indistinguishability obfuscation implies witness PRFs, show that functional secret sharing *implies* distributed secret sharing (assuming one-way functions).

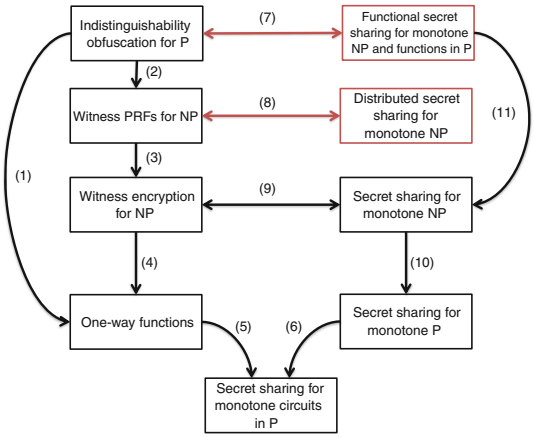

Fig. 1. The secret sharing zoo. (1) Holds assuming that NP $\not\subseteq$ io-BPP [24]. (2) Holds assuming one-way functions. (3) [34]. (4) Holds assuming the existence of a hard-on-average NP-problem [24]. (5) Yao's unpublished work. (6) By definition. (7) This work; the left-to-right arrow assumes one-way functions. (8) This work. (9) [25]; the left-to-right arrow assumes one-way functions. (10) By definition. (11) By definition.

1.1 Overview of Our Techniques

Distributed Secret Sharing and Witness PRFs. Here, we provide a high-level overview of our technique for transforming distributed secret sharing schemes into witness PRFs. At first, this seems like a difficult task. Indeed, distributed secret sharing only specifies a single secret: the shared secret for the groups of qualified parties. In contrast, in a witness PRF each instance corresponds to a secret, namely the output of the PRF on this instance. How can we obtain many secrets out of one?

Our main observation is that distributed secret sharing schemes are *reusable*. Suppose a set $\mathcal{P}_1$ of n parties runs the distributed secret sharing protocol, each

party in $\mathcal{P}_1$ generating a secret/public share pair, and publishing the public share. Now, suppose a second set of n parties $\mathcal{P}_2$ wishes to run the distributed secret sharing protocol, and that there is some party i that is in both $\mathcal{P}_1$ and $\mathcal{P}_2$. Distributed secret sharing is reusable in the sense that party i does not need to generate a fresh secret/public share pair for the second invocation of the protocol, but can instead *reuse* the shares he already has. Thus, party i does not need to publish any additional material to take part in the second sharing. Taking this a step further, $N \gg n$ parties can each generate secret/public shares and publish the public shares. Then, various sets of n of them can engage in the distributed secret sharing protocol *without any additional setup or interaction*. This observation can be seen as a generalization of the fact that non-interactive key exchange (both in the two-party and multi-party setting) is reusable.

Since distributed secret sharing schemes are reusable, there are really many implicit secrets, one for every possible subset of the N parties of size n. This will be the source of our many secrets for our witness PRFs. To show how we use this idea of reusability, we sketch our approach for a simpler task: using *threshold* distributed secret sharing to build Hamming ball constrained PRFs.

Threshold Distributed Sharing Schemes to Hamming Ball Constrained PRFs. Recall that a constrained PRF (as defined by Boneh and Waters [10]) is a normal PRF with some additional requirements: First, given the secret key k, and a subset $T \subset \mathcal{X}$ where $\mathcal{X}$ is the domain of the PRF, it is possible to *constrain* the key k to the set T, producing a constrained key k_T. Next, given k_T and a point $x \in T$, it is possible to compute $\mathsf{PRF}_k(x)$. For security, we require that, even given k_T, for all $x \notin T$, $\mathsf{PRF}_k(x)$ is pseudorandom. For this exposition, we will consider Hamming ball constraints, where $\mathcal{X} = \{0,1\}^n$, and the possible sets T consist of all points withing Hamming distance r of some center point c.

Suppose that r is fixed a priori (this is assumed here for simplicity – our actual scheme handles the case in which r is not fixed a priori). Our Hamming ball constrained PRF is defined as follows. Let $N = 2n$ be the total number of parties, and label each party by a pair $(i,b) \in [n] \times \{0,1\}$. Generate secret/public shares $(\Pi_{i,b}, P_{i,b})$ for each of the N parties for the threshold distributed secret sharing scheme on n parties and threshold $n - r$. The secret key consists of all the public and secret shares. For every input $x \in \{0,1\}^n$, let $\mathcal{P}_x$ be the subset of n parties labeled by (i, x_i) for $i \in [n]$. $\mathsf{PRF}(x)$ is defined to be the shared secret S for the set of parties $\mathcal{P}_x$ defined by x. Since the secret key consists of $n \geq n - r$ of the secret shares for $\mathcal{P}_x$, the secret key allows for computing $\mathsf{PRF}(x)$.

The constrained key k_T for the Hamming ball T of radius r around center c consists of all of the public shares, as well as the secret shares for the set $\mathcal{P}_c$. For any input x with Hamming distance at most r from c, k_T contains at least $n - r$ of the secret shares for $\mathcal{P}_x$, and so $\mathsf{PRF}(x)$ can be computed. For x at distance more than r away, k_T contains fewer than $n - r$ secret shares for $\mathcal{P}_x$, so the security of the threshold distributed secret sharing scheme implies that $\mathsf{PRF}(x)$ is hidden.

For the general distributed secret sharing to witness PRF construction, we will make use of a similar strategy, defining the output of the PRF to be the

shared secret S corresponding to a subset of parties. However, the construction becomes somewhat more complicated. For starters, the class of Hamming balls is very simple, and moreover has a lot of symmetry. In contrast, the general NP languages are much more complex and have no simple structural properties we can use. Additionally, we will need to allow the parties to be able to input a witness. We refer to Sect. 3.4 for the full details.

Functional Secret Sharing and iO. The fact that general-purpose functional secret sharing implies iO is rather straight-forward. Indeed, as we mentioned, function secret sharing is a special case of functional secret sharing, and thus, an obfuscation of a circuit is just the shares generated by the function secret sharing. Security of the obfuscator follows directly from the security of the function secret sharing scheme.

The other direction (namely, from iO to functional secret sharing) is more complicated. To this end, we rely on ideas developed by [25] in order to show that witness encryption implies (standard) secret sharing for mNP. Specifically, when sharing the secret S with respect to a function F and an access structure M, the share of party i will be an opening of a commitment and the iO of a circuit that given as input the secret openings of a subset of parties X verifies the openings, verifies the validity of the instance (together with a witness) with respect to M, and if all tests pass, it outputs the value $F(X, S)$. The security of this scheme relies on the perfect binding of the commitments and the indistinguishability guarantee of the obfuscator.

We note that multi-input functional encryption (MIFE) [20] provides another natural path to functional secret sharing. In an MIFE scheme, a secret key SK_G corresponds to an k-input function G, and message can be encrypted to any one of the k inputs to G. Denote the encryption of a message m to the i^{th} input as $\mathsf{Enc}_i(m)$. With the secret key and ciphertexts $\mathsf{Enc}_i(m_i)$ for $i = 1, \ldots, k$, it is possible to compute $f(m_1, \ldots, m_k)$, but impossible to learn anything else the plaintexts. For simplicity, we will sketch the construction of functional secret sharing where both access structure M and function F are in P, the case of more general access structures being a straightforward extension. Let $G(x_1, \ldots, x_n, S) = M(x_1, \ldots, x_n) \wedge F(x_1, \ldots, x_n, S)$. The secret share for party $i \in [n]$ consists of $\mathsf{SK}_G, \mathsf{Enc}_1(0), \cdots, \mathsf{Enc}_n(0), \mathsf{Enc}_{n+1}(S), \mathsf{Enc}_i(1)$. Then, any subset X of parties can use SK_G together with ciphertexts $\{\mathsf{Enc}_i(X_i)\}_{i\in[n]}, \mathsf{Enc}_{n+1}(S)$ to compute $M(X) \wedge F(X, S)$. If X is qualified, this will give $F(X, S)$, whereas if X is unqualified, this will give 0. Since iO and MIFE are equivalent for general-purpose functionalities (assuming one-way functions), this construction gives an alternative way to build functional secret sharing from iO.[9]

2 Preliminaries

In this section we present the notation and basic definitions that are used in this work. For a distribution X we denote by $x \leftarrow X$ the process of sampling a value

[9] We thank a reviewer for pointing out this alternative solution.

x from the distribution X. Similarly, for a set $\mathcal{X}$ we denote by $x \leftarrow \mathcal{X}$ the process of sampling a value x from the uniform distribution over $\mathcal{X}$. For a randomized function f and an input $x \in \mathcal{X}$, we denote by $y \leftarrow f(x)$ the process of sampling a value y from the distribution $f(x)$. For an integer $n \in \mathbb{N}$ we denote by $[n]$ the set $\{1, \ldots, n\}$. 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 $\lambda > N_c$. Throughout this paper we denote by λ the security parameter.

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 any probabilistic polynomial-time algorithm $\mathcal{A}$ there exists a negligible function $\mathsf{neg}(\cdot)$ such that for all $\lambda \in \mathbb{N}$ it holds that $\left|\Pr[\mathcal{A}(1^\lambda, X_\lambda) = 1] - \Pr[\mathcal{A}(1^\lambda, Y_\lambda) = 1]\right| \leq \mathsf{neg}(\lambda)$.

2.1 Monotone-NP and Access Structures

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 $f(Y) = 1$. Given a potentially non-monotone function $f : 2^{[n]} \to \{0,1\}$, we define the *monotone closure* of f, denoted $\overline{f}$, such that $\overline{f}(Y) = 1$ if and only if there is some $X \subset Y$ such that $f(X) = 1$.

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 1 ([21]). *A function L is in* mNP *if there exists a uniform family of polynomial-size monotone non-deterministic circuit that computes L.*

Lemma 1 ([21, Theorem 2.2]). $\mathsf{mNP} = \mathsf{NP} \cap \mathsf{mono}$, *where* mono *is the set of all monotone functions.*

A 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 computational secret-sharing scheme for A is a method by which the dealer efficiently distributes shares to the parties such that (1) any subset in A can efficiently reconstruct the secret from its shares, and (2) any subset not in A cannot efficiently reveal any partial information on the secret. For more information on secret-sharing schemes we refer to [4] 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 (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 $X' \in M$.*

2.2 Commitment Schemes

In some of our constructions we need a non-interactive commitment scheme such that commitments of different strings has disjoint support. Jumping ahead, 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 (for ease of notation, we usually ignore the CRS).

Definition 3 (Commitment scheme in the CRS model). *Let $\lambda \geq 0$ be a parameter. Let $\mathsf{Com}\colon \{0,1\} \times \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^{q(\lambda)}$ be polynomial-time computable function. We say that Com is a (non-interactive perfectly binding) commitment scheme in the CRS model if the following two conditions hold:*

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

As usual, the above definition can be generalized to commitments of strings of polynomial size (rather than bits) by commiting to each bit separately.

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 [22]). For simplicity, throghout the paper we ignore the CRS and simply write $\mathsf{Com}(\cdot,\cdot)$. We say that $\mathsf{Com}(x,r)$ is the commitment to the value x with the opening r.

2.3 Multilinear Maps

Definition 4 (Multilinear maps). *We say that a map $e\colon \mathbb{G}_1^n \to \mathbb{G}_2$ is an n-multilinear map if it is satisfies the following:*

1. *$\mathbb{G}_1$ and $\mathbb{G}_2$ are groups of the same prime order.*
2. *If $a_1, \ldots, \in \mathbb{Z}$ and $x_1, \ldots, x_n \in \mathbb{G}_1$, then*

$$e(x_1^{a_1}, \ldots, x_n^{a_n}) = e(x_1, \ldots, x_n)^{\prod_{i=1}^n a_i}.$$

3. *The map e is non-degenerate in the following sense: if $g \in \mathbb{G}_1$ is a generator of $\mathbb{G}_1$, then $e(g, \ldots, g)$ is a generator of $\mathbb{G}_2$.*

We say that e is an efficient n-multilinear map if it is effiently computable, namely, there exists a polynomial-time algorithm that computes $e(x_1^{a_1}, \ldots, x_n^{a_n})$ for any $a_1, \ldots, a_n \in \mathbb{Z}$ and $x_1, \ldots, x_n \in \mathbb{G}_1$.

An efficient mulilinear map generator $\mathsf{MMap.Gen}(1^\lambda, n)$ is a probabilistic polynomial-time algorithms that gets as input two inputs 1^λ and n, and outputs a tuple (Γ, g, ℓ), where Γ is the description of an efficient n-multlilinear map $e\colon \mathbb{G}_1^n \to \mathbb{G}_2$, g is a generator of G_1, and ℓ is the order of the groups $\mathbb{G}_1$ and $\mathbb{G}_2$.

Next, we define the multilinear Diffie-Hellman assumption. Roughly, the assumption is that given $g, g^{a_1}, \ldots, g^{a_n}$, it is hard to compute $e(g, \ldots, g)^{\prod_{i=1}^n a_i}$, or even distinguish it from a random value.

Definition 5 (Multilinear decisional Diffie-Hellman assumption [9]). *We say that an efficient n-multilinear map generator* $\mathsf{MMap.Gen}$ *satisfies the* multilinear decisional Diffie-Hellman *(MDDH) assumption if for every polynomial time algorithm $\mathcal{A}$ there exists a negligible function* $\mathsf{neg}(\cdot)$ *such that for $\lambda \in \mathbb{N}$ it holds that*

$$\mathsf{Adv}^{\mathsf{mDH}}_{\mathsf{MMap.Gen},\mathcal{A},n,\lambda} = \Big| \Pr\left[\mathcal{A}\left(g, g^{a_0}, \ldots, g^{a_n}, e(g, \ldots, g)^{\prod_{i=0}^n a_i}\right) = 1\right] - \Pr\left[\mathcal{A}\left(g, g^{a_0}, \ldots, g^{a_n}, K\right)\right] = 1\Big] \Big| \leq \mathsf{neg}(\lambda),$$

where the probability is over the execution of $(\Gamma, g, \ell) \leftarrow \mathsf{MMap.Gen}(1^\lambda, n)$, the choice of $a_0, \ldots, a_n \leftarrow (\mathbb{Z}/\ell\mathbb{Z})^{n+1}$, $K \leftarrow \mathbb{G}_2$, and the internal randomness of $\mathcal{A}$.

We note that we do not know of any "ideal" multilinear maps as described above that plausibly support the MDDH assumption. Instead, current candidates are "noisy" [14,16]. In particular, the group elements have some noise, and only a certain number of group operations are allowed before the multilinear identity fails. Moreover, each group element actually has many representations, and a special extraction procedure is required to obtain a unique "canonical" representation for a particular element. The extraction is only allowed in $\mathbb{G}_2$. Despite this departure from the ideal notion described above, it is usually straightforward (though often tedious) to use current candidate maps in place of the ideal maps. Therefore, for ease of exposition, we will describe our applications of multilinear maps in terms of the ideal abstraction, noting that the applications can be adapted to use the noisy candidate multilinear maps from the literature.

2.4 Witness Pseudorandom Functions

Witness pseudorandom functions (witness-PRFs) were recently introduced by Zhandry [34]. He showed that several important primitives, that were previously only known from iO (see Definition 7), follow from this seemingly weaker assumption. We note that witness-PRFs are related to witness encryption [18], but seem to be stronger.

Definition 6 (Witness-PRFs [34]). *A witness pseudorandom function is a tuple* $(\mathsf{Gen}, \mathsf{PRF}, \mathsf{Eval})$ *where:*

1. $\mathsf{Gen}(1^\lambda, R)$ *is a polynomial-time randomized procedure that takes as input a security parameter and a relation* $R : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ *represented as a circuit, and outputs a private function key* fk *and a public evaluation key* ek. *The relation* R *defines an* NP *language* L.
2. $\mathsf{PRF}(\mathsf{fk}, x)$ *is a polynomial-time deterministic procedure that takes as input the function key* fk *and an instance* $x \in \{0,1\}^n$.
3. $\mathsf{Eval}(\mathsf{ek}, x, w)$ *is a polynomial-time deterministic procedure that takes as input the evaluation key* ek, *an instance* $x \in \{0,1\}^n$, *and a witness* $w \in \{0,1\}^m$.
4. ***Correctness:*** *If* $x \in L$, *and moreover* w *is a valid witness for* x *(that is,* $R(x,w) = 1$*), then*

$$\Pr[\mathsf{Eval}(\mathsf{ek}, x, w) = \mathsf{PRF}(\mathsf{fk}, x)] = 1,$$

   where $(\mathsf{fk}, \mathsf{ek}) \leftarrow \mathsf{Gen}(1^\lambda, R)$ *and the probability is taken over the randomness* Gen.
5. ***Security:*** *For any relation* R *and any probabilistic polynomial-time algorithm* D, *there exists a negligible function* $\mathsf{neg}(\cdot)$ *such that for any* $\lambda \in \mathbb{N}$ *and any* $x \notin L$, *it holds that*

$$|\Pr[D(\mathsf{ek}, \mathsf{PRF}(\mathsf{fk}, x)) = 1] - \Pr[D(\mathsf{ek}, y) = 1]| < \mathsf{neg}(\lambda),$$

   where $(\mathsf{fk}, \mathsf{ek}) \leftarrow \mathsf{Gen}(1^\lambda, R)$, y *is chosen uniformly over the codomain of* PRF, *and the probabilities are taken over the randomness of* Gen, D, *and the choice of* y.

2.5 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 7 (Indistinguishability obfuscation [3]). *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 polynomial-time algorithm* iO *is called an indistinguishability obfuscator for the class* $\mathcal{C}$ *if it takes as input a security parameter and a circuit in* $\mathcal{C}$ *and outputs a new circuit so that following properties are satisfied:*

1. ***Preserving functionality:*** *There exists a negligible function* α *such that for any input length* $n \in \mathbb{N}$, *any* λ *and any* $C \in \mathcal{C}_n$ *it holds that*

$$\Pr_{\mathsf{iO}}\left[C \equiv \mathsf{iO}(1^\lambda, C)\right] = 1,$$

   where the probability is over the internal randomness of iO.
2. ***Polynomial slowdown:*** *There exists a polynomial* $p(\cdot)$ *such that: For any input length* $n \in \mathbb{N}$, *any* λ *and any circuit* $C \in \mathcal{C}_n$ *it holds that* $\left|\mathsf{iO}(1^\lambda, C)\right| \leq p(|C|)$.

3. ***Indistinguishable obfuscation:*** *For any probabilistic polynomial-time algorithm D and any polynomial $p(\cdot)$, there exists a negligible function* $\mathsf{neg}(\cdot)$*, such that for any $\lambda, n \in \mathbb{N}$, any two* equivalent *circuits $C_1, C_2 \in \mathcal{C}_n$ of size $p(\lambda)$, it holds that*

$$\left|\Pr\left[D\left(\mathsf{iO}\left(1^\lambda, C_1\right)\right) = 1\right] - \Pr\left[D\left(\mathsf{iO}\left(1^\lambda, C_2\right)\right) = 1\right]\right| \leq \mathsf{neg}(\lambda),$$

where the probabilities are over the internal randomness of iO *and D.*

3 Distributed Secret Sharing

In this section we define the notion of distributed secret sharing schemes.

Definition 8 (Distributed secret sharing). *A distributed secret sharing (DSS) scheme consists of a probabilistic setup procedure* SETUP*, a probabilistic sharing procedure* SHARE *and a deterministic reconstruction procedure* RECON *that satisfy the following requirements:*

- $\mathsf{SETUP}(1^\lambda, 1^n, V_M)$ *takes as input a security parameter λ (in unary representation) the number n of parties (also in unary), the verification procedure V_M for an* mNP *access structure M on n parties.* SETUP *outputs a common reference string* CRS*.*
- $\mathsf{SHARE}(1^\lambda, 1^n, \mathsf{CRS}, V_M, i)$ *takes as input λ, n, the common reference string* CRS*, the verification procedure V_M for an* mNP *language M, and a party index $i \in [n]$. It outputs a public share $P(i)$ and a secret share $\Pi(i)$. For $X \subseteq \mathcal{P}_n$ we denote by $\Pi(X)$ the random variable that corresponds to the set of secret shares of parties in X. We denote by P the random variable that corresponds to the set of public shares of parties in $\mathcal{P}_n$.*
- $\mathsf{RECON}(1^\lambda, 1^n, \mathsf{CRS}, V_M, P, \Pi(X), w)$ *gets as input $\lambda, n, \mathsf{CRS}, V_M$, the public shares P of all n parties, the secret shares $\Pi(X)$ of a subset of parties $X \subseteq \mathcal{P}_n$, and a witness w, and outputs a shared secret. We will sometimes abuse notation, and also write $X \subseteq [n]$ to refer to the subset of the party indices appearing in X.*
- **Correctness:** *For every set of parties $\mathcal{P}_n$ with corresponding public shares P, there is a string S such that any set of qualified parties $X \subseteq \mathcal{P}_n$ with valid witness w (i.e., $V_M(X, w) = 1$) can recover S. That is,*

$$\Pr[\mathsf{RECON}(1^\lambda, 1^n, \mathsf{CRS}, V_M, P, \Pi(X), w) = S] = 1,$$

where the probability is taken over the generation of the shares — namely, over $(P(i), \Pi(i)) \leftarrow \mathsf{SHARE}(1^\lambda, 1^n, \mathsf{CRS}, V_M, i)$ for $i \in [n]$ — and the choice of S (which will typically be information-theoretically determined by P). We will sometimes refer to S as the shared secret*.*
- **Pseudorandomness of the secret:** *For any language $M \in \mathsf{mNP}$ and any probabilistic polynomial-time algorithm D, there exists a negligible function*

$\mathsf{neg}(\cdot)$ *such that for any* $\lambda \in \mathbb{N}$ *and any unqualified set* $X \subseteq \mathcal{P}_n$ *(that is,* $X \notin M$*), it holds that*

$$|\Pr[D(P, \Pi(X), S) = 1] - \Pr[D(P, \Pi(X), K) = 1]| \leq \mathsf{neg}(\lambda),$$

where the probability is taken over the generation of the shares, namely, over $(P(i), \Pi(i)) \leftarrow \mathsf{SHARE}(1^\lambda, 1^n, \mathsf{CRS}, V_M, i)$ *for* $i \in [n]$*,* K *is sampled uniformly at random, and* S *is the shared secret defined above.*

The Shared Secret *S*. Suppose M is non-empty, which is true for any interesting access structure M. In this case, by the monotonicity of M, $\mathcal{P}_n \in M$ and there exists a witness w attesting to this fact. Then, the shared secret S is well defined and information-theoretically determined, as we can use the correctness requirement for the set $\mathcal{P}_n$ as the definition of S: $S = \mathsf{RECON}(1^\lambda, 1^n, \mathsf{CRS}, V_M, P, \Pi, w)$.[10]

In the case where M is empty, correctness is trivially satisfied for *any* definition of S. We can therefore take S to be a uniformly random variable that is completely independent of the scheme, and *unconditional* security will be trivially satisfied as well. Interestingly, this means that, when analyzing schemes, it is only necessary to analyze correctness and security for non-empty access structures M, as *any* scheme will automatically be correct and secure for empty M.

In Sect. 3.2, we show how to obtain unconditional security for a slightly wider class of access structures, which we call *trivial* access structures.

Reusability. In this work, it will be useful to distinguish between *party* and *index*. A *party* is an entity that has run SHARE, and obtained a secret and public share. That party's *index* is the input i that was fed into SHARE. Multiple parties may share the same index. We will say a set X of parties is *complete* if, for every index i, there is exactly one party. Complete sets of parties are those for which RECON can be run, and therefore there is a shared secret S_X associated with every complete set of parties. In this sense, a DSS scheme is *reuseable*: an individual party with index i can take part in multiple sharings as part of different complete sets of parties, while only running SHARE once and publishing a single public share. This observation generalizes the fact that non-interactive key exchange (in the 2-party or multi-party setting) is reusable. This reusability property will be crucial for building witness PRFs from DSS.

Restricted Access Structures. The above definition requires that the DSS algorithms work for *any* access structure M recognized by a polynomial-sized verification circuit V_M. It is also possible to consider weaker versions where M is required to have a specific structure. For example, it is possible to consider M that are recognized by polynomial-size circuits (that is, $M \in \mathsf{P}$). In Sect. 3.3, we consider an even more restricted setting where M is just a threshold function: $X \in M$ if and only if $|X| \geq t$ for some threshold t. We call these restrictions

[10] We note that to compute S we need to know w which may be computationally hard for some languages.

DSS for P or *DSS for threshold*, respectively. When distinguishing DSS for these limited classes from the standard definition above, we call the standard definition *DSS for* mNP. Finally, one can consider DSS for a specific, fixed access structure M, which we call *DSS for* M. For example, if M consists of all non-empty subsets (a special case of threshold where $t = 1$), then DSS for M is exactly multiparty non-interactive key exchange with trusted setup [9].

3.1 Alternative Definitions

We introduce several alternative definitions for distributed secret sharing. We first give a *strong* variant in which the sharing procedure is independent of the access structure V_M and of the party index i. Our second alternative is a *witnessless* version in which qualified sets are defined by an arbitrary circuit (possibly a non-monotone one). Our last variant is a definition of distributed secret sharing that has no setup (also known as no common reference string).

Definition 9 (Strong distributed secret sharing). *A strong distributed secret sharing scheme is a special case of a regular distributed secret sharing scheme (as in Definition 8) with the following differences:*

- $\mathsf{SETUP}(1^\lambda, 1^n, V_M) = \mathsf{SETUP}(1^\lambda, 1^n, 1^{|V_M|})$. *That is,* SETUP *does not depend on V_M, except through the size of the circuit for V_M, but is otherwise independent of V_M or the language M.*
- $\mathsf{SHARE}(1^\lambda, 1^n, \mathsf{CRS}, V_M, i) = \mathsf{SHARE}(1^\lambda, 1^n, \mathsf{CRS}, 1^{|V_M|})$. *That is,* SHARE *does not depend on V_M except for its size, and also does not depend on the party index i.*
- $\mathsf{RECON}(1^\lambda, 1^n, \mathsf{CRS}, V_M, P, \Pi(X), w)$ *now interprets P as a being ordered, and uses the order to determine the party index corresponding to each public share. From this information and $\Pi(X)$,* RECON *can determine the subset $X \subseteq [n]$ of indices for which secret shares are provided.*
- *For each verification circuit V_M and set of n parties $\mathcal{P}_n$, correctness is defined using an associated secret $S_{V_M,\mathcal{P}_n}$ that potentially varies for different V_M and $\mathcal{P}_n$ pairs. Notice that since a party is not assigned an index at sharing time, the only restriction we place on $\mathcal{P}_n$ is its size (i.e., $|\mathcal{P}_n| = n$), but we do not need $\mathcal{P}_n$ to be a complete set.*

The advantage of a strong DSS scheme is that the access structure does not need to be specified at sharing time. This allows parties to play multiple roles in different sharing executions without having to generate new shares, and allows a single sharing to be used for many different access structures. This will result in significant communication savings if many sharings with different access structures are being executed. When differentiating between the strong and regular variants, we will call the regular distributed secret sharing variant a *weak* scheme.

Definition 10 (Witnessless distributed secret sharing). *A* witnessless *distributed secret sharing is the following modification to (weak) distributed secret*

sharing, where the access structure M is set to be the monotone closure $\overline{C}$ of some (potentially non-monotone) function C.[11] *In addition, we make the following modifications to the algorithms of the scheme:*

- $\mathsf{SETUP}(1^\lambda, 1^n, C)$*, instead of taking as input the verification circuit V_M, now takes as input a circuit for the function C, which is potentially non-monotone. For the strong variant,* SETUP *takes as input $|C|$ instead of $|V_M|$.*
- $\mathsf{SHARE}(1^\lambda, 1^n, \mathsf{CRS}, C, i)$ *also takes as input C instead of V_M. For the strong variant,* SHARE *takes as input $|C|$ instead of $|V_M|$, and does not take i as input.*
- $\mathsf{RECON}(1^\lambda, 1^n, \mathsf{CRS}, C, P, \Pi(X))$ *similarly takes as input C instead of V_M. Also,* RECON *does* not *take as input a witnesses, hence the term* witnessless.
- *Correctness is modified so that* $\Pr[\mathsf{RECON}(1^\lambda, 1^n, \mathsf{CRS}, C, P, \Pi(X)) = S] = 1$ *for any $X \subseteq \mathcal{P}_n$ such that $C(X) = 1$.*

A set X of qualified parties in $M = \overline{C}$ cannot simply feed in all of the secret shares $\Pi(X)$ into RECON to obtain the secret, as $C(X)$ may not be 1. Instead, if they know a subset $X' \subseteq X$ such that $C(X') = 1$ (which must exist since X is qualified), they may simply feed the subset of their secret shares corresponding to X', namely $\Pi(X')$, into RECON, and correctness guarantees that they will learn the secret. Thus, even though the algorithms in a witnessless distributed secret sharing scheme do not take a witness as input, reconstructing the secret still requires knowing a witness, namely the subset X'.

We note that the access structure M is monotone, and is clearly in NP. Therefore, Lemma 1 shows that M is recognized by a monotone nondeterministic verification procedure V_M. Thus, the above formulation of distributed secret sharing is equivalent to regular DSS (with witnesses) where we restrict to access structures of this form. Therefore, this notion is no stronger than regular DSS.

We note that many NP languages naturally are represented using a circuit C, such as Hamiltonian Cycle (where C checks that the set of edges forms a Hamiltonian cycle) and Subset Sum (where C checks that the subset of integers sums to 0).

Definition 11 (Distributed secret sharing without setup). *In a distributed secret sharing scheme* without setup*, there is no* SETUP *algorithm, and* SHARE *and* RECON *do not take* CRS *as input. When distinguishing between schemes with and without setup, we call the standard notion (Definition 8)* distributed secret sharing with trusted setup*.*

Immediate Relations Between Definitions. All of the above variations are orthogonal, giving us 8 variants of distributed secret sharing. We make the following observation:

- Any of the 4 variants of *strong* distributed secret sharing imply the corresponding variant of *weak* distributed secret sharing. This is because being a strong scheme just imposes constraints on the form of the algorithms.

[11] Recall that the monotone closure $\overline{C}$ of a function C includes all sets X such that some subset $X' \subseteq X$ satisfies $C(X') = 1$ (see Sect. 2.1).

- Any of the 4 variants of distributed secret sharing *with witnesses* imply the corresponding *witnessless* variant, since the witnessless condition imposes a restriction on the languages allowed.
- Any of the 4 variants of distributed secret sharing *without trusted setup* imply the corresponding variant *with trusted setup*, where SETUP outputs an empty string.

In Sect. 3.4, we will show that all of the above notions are equivalent, and moreover that they are equivalent to witness PRFs.

3.2 Distributed Secret Sharing Implies One-Way Functions

Witness PRFs trivially imply one-way functions, and therefore by our equivalence in Sect. 3.4, information-theoretic distributed secret sharing is impossible for general access structures.

In this section, we consider DSS for specific access structures, and ask: for what access structures M is information-theoretic DSS possible? To answer this question we first define trivial access structures, and then in Theorem 1 we show that a DSS scheme for any non-trivial access structures implies one-way functions. DSS for trivial access structures, on the other hand, are shown to have a very simple information-theoretically secure construction.

Definition 12 (Trivial access structures). *We say that an access structure M for a set of parties $\mathcal{P}$ is* trivial *if either M is empty, or there exists a subset $X \subseteq \mathcal{P}$ such that $Y \in M$ if and only if $X \subseteq Y$.*

We call such access structures trivial due to the following reasons:

- Parties outside of X are irrelevant to the access structure, as they can be added or removed from a set of parties without changing the set's qualified status. Therefore, such a protocol is morally equivalent to the case where $X = \mathcal{P}$.
- When $X = \mathcal{P}$, all parties must get together to reconstruct the shared secret. In this case, there appears to be no reason to engage in the protocol in the first place, as the parties can just choose the group secret when they all coordinate at reconstruction time.

Note that trivial access structures are in P, so there is no distinction between standard DSS and witnessless DSS.

Theorem 1. *For an access structure M, the following hold:*

- *If M is trivial, then there exists a perfectly-secure DSS for M in the strongest possible sense (that is, strong DSS without setup)*
- *If M is non-trivial, then the existence of any DSS for M in the weakest possible sense (that is, weak DSS with trusted setup) implies the existence of one-way functions.*

Proof. Let M be trivial, with subset X such that $Y \in M$ if and only if $X \subseteq Y$. We then get the following strong DSS scheme for M without setup that has single-bit shared secrets:

- $\mathsf{SHARE}()$: sample a random $\Pi(i) \leftarrow \{0,1\}$, and publish an empty string as the public share $P(i) = \emptyset$.
- $\mathsf{RECON}(P, \Pi(Y))$: if $X \subset Y$, simply XOR and output the shares for parties in X, namely $S \leftarrow \mathtt{XOR}_{i \in X} \Pi(i)$. If X is not a subset of Y, abort.

The correctness of the protocol is trivial. For security, note that for any set Y that does not contain X, there is some party $i \in X \setminus Y$ such that the set of shares for Y does not contain the secret share $\Pi(i)$. Therefore, $\Pi(i)$ is independent of the shares $\Pi(Y)$. Thus, S is independent of $\Pi(Y)$. Perfect security follows.

The proof of the other case (in which M is a non-trivial access structure) can be found in the full version [26]. ■

3.3 Distributed Secret Sharing for Threshold

In this section we present a distributed secret sharing scheme for the threshold access structure. The proof of security relies on the multilinear decisional Diffie-Hellman assumption (see Definition 5). This construction works in the trusted setup model (which is used for the setup of the multilinear map). Assume there are n parties and the threshold condition says that any t of them should be able to reconstruct the secret.

Lemma 2. *Assuming an $(n-t)$-multilinear map that satisfied the MDDH assumption, there is a t-out-of-n (weak) distributed secret sharing scheme (with trusted setup).*[12]

Proof. We start with the description of the scheme. The trusted setup will consists of an $n-t$ multilinear map. For the sharing, party p_i generates a random s_i and published $h_i = g^{s_i}$. The shared secret key is $S = e(g, \ldots, g)^{\prod_{i=1}^{n} s_i}$. With t of the s_i's one can easily compute S by pairing the other h_i's, and then raising the result by each of the s_i's. Security in the case of fewer than t shares follows from the security of the multilinear DH assumption.

More precisely, in the trusted setup we run $\mathsf{MMap.Gen}(1^\lambda, n)$ to get (Γ, g, ℓ) which we set as the public parameters. The sharing procedure of party p_i samples a random $s_i \leftarrow \mathbb{Z}$ (which is kept secret) and outputs $h_i = g^{s_i}$. The shared secret key is $S = e(g, \ldots, g)^{\prod_{i=1}^{n} s_i}$. For correctness, we observe that given the secret shares of any subset of the t parties one can compute S. Indeed, given $h_{i_1}, \ldots, h_{i_{n-t}}$ one can compute

$$e(h_{i_1}, \ldots, h_{i_{n-t}}) = e(g, \ldots, g)^{\prod_{j=1}^{n-t} s_{i_j}}$$

[12] Since threshold is in P, there are no witnessness, so there is no distinction between the standard and witnessless notions of DSS.

and then, by raising the right-hand side to the powers $s_{i_{n-t+1}}, \ldots, s_{i_n}$, compute

$$\left(e(g,\ldots,g)^{\prod_{j=1}^{n-t} s_{i_j}}\right)^{\prod_{j=n-t+1}^{n} s_{i_j}} = S$$

The proof of security can be found in the full version [26]. ■

Hamming Ball Constrained PRFs. We show that any distributed secret sharing scheme for threshold implies constrained PRFs that can be constrained to a Hamming ball around an arbitrary point. One limitation of our construction is that the PRF only allows a single collusion: an adversary that sees the PRF constrained to two Hamming balls can potentially recover the entire secret key.

Of course, our construction of DSS for threshold relies on the multilinear Diffie-Hellman assumption, which already implies constrained PRFs for all circuits with arbitrary collusions [10]. However, our conversion here is generic and applies to any threshold DSS scheme, which perhaps can be based on simpler assumptions than multilinear maps. Perhaps more importantly, the ideas presented here will be used in Sect. 3.4 to show the equivalence of general DSS and witness PRFs. Thus, this construction can be viewed as a warm-up to Theorem 3.

Definition 13 (One-time constrained PRFs for Hamming balls). *A constrained PRFs for Hamming balls is a tuple of algorithms* $(\mathsf{Gen}, \mathsf{PRF}, \mathsf{Constrain}, \mathsf{Eval})$ *where:*

- $\mathsf{Gen}(1^\lambda, 1^n)$ *is a polynomial-time randomized procedure that takes as input a security parameter* λ *and a bit length* n*, and outputs a function key* fk*.*
- $\mathsf{PRF}(\mathsf{fk}, x)$ *is a polynomial-time deterministic procedure that takes as input the function key* fk *and a bit string* $x \in \{0,1\}^n$*.*
- $\mathsf{Constrain}(\mathsf{fk}, c, r)$ *is a polynomial-time (potentially randomized) procedure that takes as input the function key* fk*, a point* $c \in \{0,1\}^n$*, and a radius* $r \in [0, n]$*, and outputs the constrained evaluation key* ek *corresponding to the Hamming ball of radius* r *centered at* c*.*
- $\mathsf{Eval}(\mathsf{ek}, x)$ *is a polynomial-time deterministic procedure that takes as input the evaluation key* ek *and a bit string* $x \in \{0,1\}^n$*.*
- ***Correctness:*** *If* x *and* c *differ on at most* r *bits, then*

$$\Pr[\mathsf{Eval}(\mathsf{ek}, x) = \mathsf{PRF}(\mathsf{fk}, x)] = 1,$$

  where $\mathsf{fk} \leftarrow \mathsf{Gen}(1^\lambda, 1^n), \mathsf{ek} \leftarrow \mathsf{Constrain}(\mathsf{fk}, c, r)$ *and the probability is taken over the randomness of* $\mathsf{Gen}, \mathsf{Constrain}$*.*
- ***One-time security:*** *For any probabilistic polynomial time algorithm* D*, there exists a negligible function* $\mathsf{neg}(\cdot)$ *such that for any* $\lambda \in \mathbb{N}$ *and any* $r \in [0, n]$*,* $x \in \{0,1\}^n$ *and* $c \in \{0,1\}^n$ *such that* x *and* c *differ in* strictly more than r *points, it holds that*

$$|\Pr[D(\mathsf{ek}, \mathsf{PRF}(\mathsf{fk}, x)) = 1] - \Pr[D(\mathsf{ek}, y) = 1]| < \mathsf{neg}(\lambda),$$

  where the probabilities are taken over the choice of $\mathsf{fk} \leftarrow \mathsf{Gen}(1^\lambda, 1^n)$*,* $\mathsf{ek} \leftarrow \mathsf{Constrain}(\mathsf{fk}, c, r)$*, and* y *which is chosen uniformly at random over the co-domain of* PRF*.*

Theorem 2. *If secure distributed secret sharing for threshold access structures exists, then secure one-time constrained PRFs for Hamming balls exists.*

At first glance, building a Hamming ball constrained PRFs from threshold DSS appears to be a difficult task. Indeed, the natural approach to constructing witness PRFs would be have the public evaluation key be the set of public shares P, and perhaps some subset of secret shares $\Pi(X)$ for $X \subseteq \mathcal{P}$; the secret function key would naturally be the complete set of secret shares $\Pi(\mathcal{P})$. However, it is unclear how to define the PRF $\mathsf{PRF}(\cdot)$. One possibility is to try to set the outputs of the PRF to be the shared secret S. However, our threshold DSS only explicitly has a single S. Yet, we need many secret outputs, one for each possible input.

To get around these limitations, we make use of the fact that distributed secret sharing is *reusable*, as discussed in the beginning of Sect. 3. For example, suppose two distinct sets of parties $\mathcal{P}_0 \neq \mathcal{P}_1$ wish to carry out the protocol, and there is some party i that is a member of both sets. Then, party i could *reuse* his public share for both runs of the protocol. More generally, for a large collection $\mathcal{C}$ of parties with $|\mathcal{C}| \gg n$, all parties can run SHARE exactly once, and then any subset $\mathcal{P} \subseteq \mathcal{C}$ of n parties can then run the distributed secret sharing protocol without any interaction (assuming that $\mathcal{P}$ is complete, meaning every party index is present exactly once).

Our idea, then, is to have the PRF value be the shared secret for a subset of $\mathcal{C}$, and the input to the PRF selects which subset to use. We need to be careful, though, as we need to ensure that the subset is complete and contains every party index exactly once. We show that such valid subsets can still be used to construct witness PRFs.

Proof of Theorem 2. Let $(\mathsf{SETUP}, \mathsf{SHARE}, \mathsf{RECON})$ be a distributed secret sharing scheme for threshold. We start with the construction of the constrained PRF.

- **$\mathsf{Gen}(1^\lambda, 1^n)$**: First, run $\mathsf{CRS} \leftarrow \mathsf{SETUP}(1^\lambda, 1^{2n}, \mathsf{thr} = n)$. That is, initialize the setup procedure for the threshold DSS scheme with $2n$ parties and threshold n. Next, we will define a set $\mathcal{P} = \{(i,b)\}_{i\in[n]} \cup [n+1, 2n]$ of parties, where party (i,b) for $i \in [n]$ has index i, and party i for $i \in [n+1, 2n]$ has index i. Now run SHARE for each party. That is, run

$$(P_{i,b}, \Pi_{i,b}) \leftarrow \mathsf{SHARE}(1^\lambda, 1^{2n}, \mathsf{CRS}, \mathsf{thr} = n, i) \text{ for } i \in [n],$$
$$(P_i, \Pi_i) \leftarrow \mathsf{SHARE}(1^\lambda, 1^{2n}, \mathsf{CRS}, \mathsf{thr} = n, i) \text{ for } i \in [n+1, 2n].$$

  Let $\Pi = \{\Pi_{i,b}\}_{i\in[n]} \cup \{\Pi_i\}_{i\in[n+1,2n]}$ be the set of secret shares, and P the corresponding set of public shares. Output the function key $\mathsf{fk} = (\mathsf{CRS}, \Pi, P)$.
- **$\mathsf{PRF}(\mathsf{fk}, x)$**: Define $\mathcal{P}_x$ to be the collection of parties (i, x_i) for $i \in [n]$, together with parties i for $i \in [n+1, 2n]$. Define

$$P(\mathcal{P}_x) = \{P_{i,x_i}\}_{i\in[n]} \cup \{P_i\}_{i\in[n+1,2n]} \text{ and } \Pi(\mathcal{P}_x) = \{\Pi_{i,x_i}\}_{i\in[n]} \cup \{\Pi_i\}_{i\in[n+1,2n]}.$$

  Notice that $\mathcal{P}_x$ is complete, in that each party index is present. Now, use the secret shares to reconstruct the shared secret for $\mathcal{P}_x$:

$$S \leftarrow \mathsf{RECON}(1^\lambda, 1^{2n}, \mathsf{CRS}, \mathsf{thr} = n, P(\mathcal{P}_x), \Pi(\mathcal{P}_x))$$

  and output S.

– **Constrain(fk, c, r)**: Let

$$\mathsf{ek} = (c, r, P, \{\Pi_{i,c_i}\}_{i\in[n]} \cup \{\Pi_i\}_{i\in[n+1,n+r]})$$

be the set of secret shares Π_{i,c_i} for parties $(i, c_i), i \in [n]$, as well as r of the secret shares Π_i for for parties $i \in [n+1, 2n]$. Output ek.

– **Eval(ek, x)**: Check that x and c differ in at most r points, and otherwise abort. Let $T \subseteq [n]$ be the set of indices where x and c agree. Then, the set of parties $X = \{(i, x_i)\}_{i\in T} \cup [n+1, n+r]$ forms a subset of $\mathcal{P}_x$. Moreover, X consists of $|T|+r \geq n = t$ parties (since x and c agree on at least $n-r$ points), and ek contains the secret shares $\Pi(X)$ for all of these parties. Therefore, run

$$K \leftarrow \mathsf{RECON}(1^\lambda, 1^{2n}, \mathsf{CRS}, \mathsf{thr} = n, P(\mathcal{P}_x), \Pi(X))$$

and output K.

An example of our construction for the case $n = 5$ is given in Fig. 2.

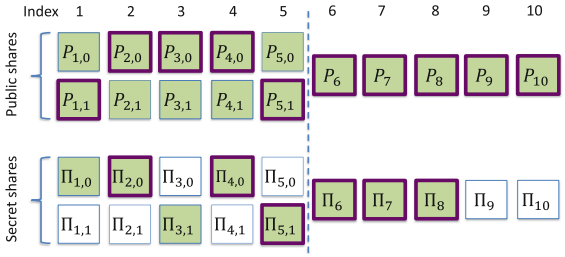

Fig. 2. Example instantiation for $n = 5$. The underlying threshold DSS scheme is instantiated with 10 indices and threshold $t = 5$. For indices 1 through 5, SHARE is run twice, returning two sets of secret/public pairs for each index 1 through 5. For indices 6 through 10, SHARE is run once. The secret key fk consists of *all* public shares and secret shares. The shares highlighted in green correspond to the evaluation key ek for the Hamming ball centered at $c = 00101$ with radius $r = 3$. The public shares outlined in bold purple indicate the public shares whose shared secret S is $\mathsf{PRF}(\mathsf{fk}, x = 10001)$. Notice that x and c have a Hamming distance $2 \leq r$, so S should be computable from ek. Indeed ek contains $6 \geq t$ of the corresponding secret shares (also outlined in bold purple), meaning that it is possible to construct $S = \mathsf{PRF}(\mathsf{fk}, x)$ from ek.

Correctness follows immediately from the observations above. Indeed, given x and r that differ on at most r coordinates, one can generate the secret shares for the set of parties X defined above. Now, the correctness of the distributed secret sharing scheme implies that K must be equal to S, where K and S are as defined in the scheme above. For security, we have the following claim whose proof can be found in the full version [26]:

Claim. If $(\mathsf{SETUP}, \mathsf{SHARE}, \mathsf{RECON})$ is a secure distributed secret sharing scheme for threshold, then $(\mathsf{Gen}, \mathsf{PRF}, \mathsf{Constrain}, \mathsf{Eval})$ is a one-time secure constrained PRF for Hamming balls.

This completes the proof of the theorem.

3.4 Distributed Secret Sharing Is Equivalent to Witness PRFs

In this section, we prove that all variants of distributed secret sharing are actually equivalent to witness PRFs. Together with Zhandry's construction of witness PRFs [34], this gives a construction of distributed secret sharing from simple assumptions on multilinear maps.

Theorem 3. *The existence of the following are equivalent:*

- *Witness PRFs for* NP.
- *Any of the 8 variants of distributed secret sharing for* mNP.

Proof. To prove the theorem, it suffices to prove the following:

1. Weak distributed secret sharing without witnesses and with trusted setup implies witness PRFs.
2. Witness PRFs imply strong distributed secret sharing with witnesses and without trusted setup.

Distributed Secret Sharing Implies Witness PRFs. We first give the construction of witness PRFs from weak witnessless DSS with a trusted setup. Our construction and proof leverage the reusability of distributed secret sharing, and is based on the threshold DSS to Hamming ball PRF conversion presented in Sect. 3.3.

Let $(\mathsf{SETUP}, \mathsf{SHARE}, \mathsf{RECON})$ be a witnessless weak distributed secret sharing scheme with trusted setup. We build the following witness PRF $(\mathsf{Gen}, \mathsf{PRF}, \mathsf{Eval})$:

- $\mathbf{Gen}(\boldsymbol{R})$: Let n be the instance size and m the witness size. We will use a DSS scheme over a set of parties $\mathcal{P}$ with $2n + m$ party indices. We will generally think of the index set as containing $2n$ pairs $(i, b) \in [n] \times \{0, 1\}$, as well as m integers $j \in [m]$. The set of pairs $[n] \times \{0, 1\}$ we will call the "instance set", and the set of integers $[m]$ we will call the "witness set".
  Define a circuit $C\colon 2^{\mathcal{P}} \to \{0, 1\}$ that operates, given an input $S \subseteq \mathcal{P}$, as follows. If $S = \mathcal{P}$, output 1. For any i, if either both $(i, 0), (i, 1)$ from the instance set are in S or neither are in S, then C outputs 0. Otherwise if $(i, b) \in S$ (and therefore $(i, 1 - b) \notin S$), set $x_i = b$. Let x be the bit string $x_1 x_2 \ldots x_n$. Let w_j be 1 if $j \in S$ and let w be the bit string $w_1 w_2 \ldots w_n$. Then, C outputs $R(x, w)$. Recall that the monotone closure of C, $M = \overline{C}$, satisfies $X \in M$ if some subset $X' \subseteq X$ causes C to accept.
  First, we generate the CRS by running

$$\mathsf{CRS} \leftarrow \mathsf{SETUP}(1^\lambda, 1^{2n+m}, C).$$

  Now, we define the set $\mathcal{P}$ to consist of the following parties: for each index (i, b) in the instance set, we will associate two parties $\{(i, b, c)\}_{c \in \{0,1\}}$, and for each index $j \in [m]$ in the witness set, we will associate a party j. Next, we

run SHARE for each party. That is, for each $i \in [n], b \in \{0,1\}$ and $c \in \{0,1\}$, run

$$(P_{i,b,c}, \Pi_{i,b,c}) \leftarrow \mathsf{SHARE}(1^\lambda, 1^{2n+m}, \mathsf{CRS}, C, (i,b))$$

and for each $j \in [m]$, run

$$(P_j, \Pi_j) \leftarrow \mathsf{SHARE}(1^\lambda, 1^{2n+m}, \mathsf{CRS}, C, j).$$

Let $P = \{P_{i,b,c}\}_{i\in[n],b,c\in\{0,1\}} \cup \{P_j\}_{j\in[m]}$ and $\Pi = \{\Pi_{i,b,c}\}_{i\in[n],b,c\in\{0,1\}} \cup \{\Pi_j\}_{j\in[m]}$ be the set of public and secret shares, respectively. Output the function key

$$\mathsf{fk} = (\mathsf{CRS}, P, \Pi)$$

and the evaluation key

$$\mathsf{ek} = (\mathsf{CRS}, P, \{\Pi_{i,b,b}\}_{i\in[n],b\in\{0,1\}}, \{\Pi_j\}_{j\in[m]}).$$

That is, the evaluation key consists of all of the public shares, all of the secret shares for indices in the witness set, and one of the secret shares for each index (i,b) in the instance set (recall that for each index in the instance set, we have two parties).

- **$\mathsf{PRF}(\mathsf{fk}, x)$**: Let

$$\mathcal{P}_x = \{(i, b, x_i)\}_{i\in[n],b\in\{0,1\}} \cup [m]$$

so that $P(\mathcal{P}_x) = \{P_{i,b,x_i}\}_{i\in[n],b\in\{0,1\}} \cup \{P_j\}_{j\in[m]}$ and $\Pi(\mathcal{P}_x) = \{\Pi_{i,b,x_i}\}_{i\in[n],b\in\{0,1\}} \cup \{\Pi_j\}_{j\in[m]}$.
Notice that $\mathcal{P}_x$ is complete, in the sense that each index is represented exactly once. Therefore, run

$$K \leftarrow \mathsf{RECON}(1^\lambda, 1^{2n+m}, \mathsf{CRS}, C, P(\mathcal{P}_x), \Pi(\mathcal{P}_x))$$

and output K.
That is, out of the entire collection of $4n+m$ parties, use the input x to select the appropriate set of parties $\mathcal{P}_x$ of size $2n+m$. Then, compute the shared key for that set of parties.

- **$\mathsf{Eval}(\mathsf{ek}, x, w)$**: Let $\mathcal{P}_x$ and $P(\mathcal{P}_x)$ be as above. Let $S_{x,w} = \{(i, x_i, x_i)\}_{i\in[n]} \cup \{j\}_{j:w_j=1}$ and $\Pi(S_{x,w}) = \{\Pi_{i,x_i,x_i}\}_{i\in[n]} \cup \{\Pi_j\}_{j:w_j=1}$. Run

$$K \leftarrow \mathsf{RECON}(1^\lambda, 1^{2n+m}, \mathsf{CRS}, C, P(\mathcal{P}_x), \Pi(S_{x,w}))$$

and output K.

To show correctness, we need to argue that $\mathsf{Eval}(\mathsf{ek}, x, w) = \mathsf{PRF}(\mathsf{fk}, x)$ for all w such that $R(x,w) = 1$. Indeed, $\mathsf{Eval}(\mathsf{ek}, x, w)$ attempts to compute the shared secret for the set of parties $\mathcal{P}_x$. Notice that the set $S_{x,w}$ is a subset of the set $\mathcal{P}_x$, and consists of the parties in $\mathcal{P}_x$ with indices in $T_{x,w} = \{(i, x_i)\}_{i\in[n]} \cup \{j\}_{j:w_j=1}$.

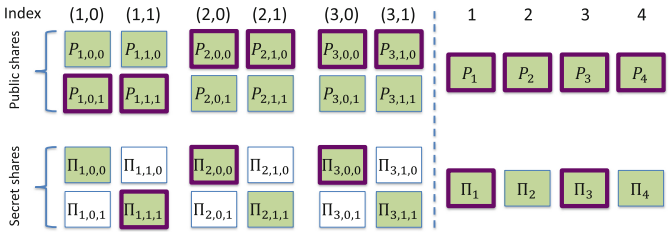

Fig. 3. Example instantiation for instance size $n = 3$ and witness size $m = 4$. The underlying threshold DSS scheme is instantiated with 10 indices, 6 for the instance set having the form (i, b), and 4 for the witness set having the form j. For each instance set index (i, b), SHARE is run twice, returning two sets of secret/public pairs for parties $(i, b, 0), (i, b, 1)$. For witness set indices, SHARE is run once. The secret key fk consists of *all* public keys and secret shares, and the evaluation key consists of the green highlighted shares. An example evaluation on $x = 100$ is given. The instance x selects the subset $\mathcal{P}_x$, whose public shares are bolded in purple. For these shares, there is a shared secret S, and the value of PRF on x is defined to be S. Suppose $w = 1010$ is a valid witness for x. Then, the secret shares for parties in $S_{x,w}$ are boxed in bold purple and represent the set of secret shares inside ek that can be fed into RECON to yield S. Notice that, among the instance set of indices, ek only contains secret shares for the parties in $\mathcal{P}_x$ that have indices (i, x_i).

Now notice that $C(T_{x,w})$ computes exactly $R(x, w) = 1$. Thus, the set of secret shares $\Pi(S_{x,w})$ is sufficient to reconstruct the shares secret S for $\mathcal{P}_x$. Notice that S is also the value outputted by $\mathsf{PRF}(\mathsf{fk}, x)$. Therefore, $\mathsf{Eval}(\mathsf{ek}, x, w) = \mathsf{PRF}(\mathsf{fk}, x)$ as desired. An example instantiation is given in Fig. 3.

It remains to prove that the scheme is secure. The proof of the following claim can be found in the full version [26].

Claim. If $(\mathsf{SETUP}, \mathsf{SHARE}, \mathsf{RECON})$ is a secure weak distributed secret sharing scheme without witnesses and with trusted setup, then $(\mathsf{Gen}, \mathsf{PRF}, \mathsf{Eval})$ is a secure witness PRF.

Witness PRFs Imply Distributed Secret Sharing. Given a Witness PRF $(\mathsf{Gen}, \mathsf{PRF}, \mathsf{Eval})$, we can easily obtain a one-way function, and from this we can obtain a pseudorandom generator f [22]. We construct the following *strong* distributed secret sharing scheme $(\mathsf{SHARE}, \mathsf{RECON})$ without trusted setup.

- **$\mathsf{SHARE}(1^\lambda, 1^n, 1^k)$:** Run $(\mathsf{fk}, \mathsf{ek}) \leftarrow \mathsf{Gen}(R)$ where R is the following NP circuit. R takes as input an instance $(V_M, \{y_i\}_{i \in \mathcal{P}_n})$, where V_M is the description of an mNP circuit of size at most k, and witness $w' = (w, \{s_i\}_{i \in X})$ for some subset $X \subseteq \mathcal{P}_n$. It outputs 1 if (1) $V_M(X, w) = 1$ and (2) $y_i = f(s_i)$ for each $i \in X$. Otherwise, R outputs 0.
  Let $s \leftarrow \mathcal{S}$ where $\mathcal{S}$ is the domain of f, and $y = f(s)$. Output public share $P(i) = (\mathsf{ek}, y)$ and secret share $\Pi(i) = s$.

– **$\mathsf{RECON}(1^\lambda, 1^n, V_M, P, \Pi(X), w)$:** Write $\Pi(X) = \{s_i\}_{i\in X}$ and $P = \{(\mathsf{ek}_i, y_i)\}_{i\in\mathcal{P}}$. Let x be the instance $(V_M, \{y_i\}_{i\in\mathcal{P}})$, and let $w' = (w, \{s_i\}_{i\in X})$ be a witness. For each i, compute

$$S_i = \mathsf{Eval}(\mathsf{ek}_i, x, w'),$$

and then compute $S = S_1 \oplus S_2 \oplus \cdots \oplus S_n$. Output S.

The correctness of the scheme follows immediately from the correctness of the underlying witness PRF. The security of the scheme follows from the following claim whose proof can be found in the full version [26].

Claim. If $(\mathsf{Gen}, \mathsf{PRF}, \mathsf{Eval})$ is a secure witness PRF and f is a secure PRG, then $(\mathsf{SHARE}, \mathsf{RECON})$ is a secure *strong* distributed secret sharing scheme with witnesses and without trusted setup.

We have shown that the weakest variant of distributed secret sharing implies witness PRFs, which in turn imply the strongest variant of distributed secret sharing. Thus, all variants of DSS and witness PRFs are equivalent, completing the proof.

4 Functional Secret Sharing

We start this section with a definition of functional secret sharing. Later, in Theorem 4, we show that general-purpose functional secret sharing is equivalent to indistinguishability obfuscation for polynomial-size circuits.

Definition 14 (Functional secret sharing). *Let $\mathcal{F} = \{F\colon 2^{\mathcal{P}_n} \to \{0,1\}^*\}$ be a class of functions. 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* functional secret sharing *scheme for M and $\mathcal{F}$ consists of a setup procedure* SETUP *and a reconstruction procedure* RECON *that satisfy the following requirements:*

1. $\mathsf{SETUP}(1^\lambda, F, S)$ *gets as input an efficiently computable function $F\colon 2^{\mathcal{P}_n} \times \{0,1\}^* \to \{0,1\}^*$ and a secret $S \in \{0,1\}^*$, and distributes a share for each party. For $i \in [n]$ denote by $\Pi(F, S, i)$ the random variable that corresponds to the share of party p_i. Furthermore, for $X \subseteq \mathcal{P}_n$ we denote by $\Pi(F, S, X)$ the random variable that corresponds to the set of shares of parties in X.*
2. ***Completeness:*** *If $\mathsf{RECON}(1^\lambda, \Pi(F, S, X), w)$ gets as input the shares of a "qualified" subset of parties and a valid witness, and outputs the value of F on X and the shared secret. Namely, for $X \subseteq \mathcal{P}_n$ such that $M(X) = 1$ and any valid witness w such that $V_M(X, w) = 1$, it holds that:*

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

where the probability is over the internal randomness of the scheme and of RECON.

3. ***Indistinguishability of the Secret:*** *For every probabilistic polynomial-time algorithm D, every function $F \in \mathcal{F}$, every subset of parties $X \subseteq \mathcal{P}_n$ and every pair of secrets S_0, S_1, as long as either $M(X) = 0$ or $F(X', S_0) = F(X', S_1)$ for every $X' \subseteq X$, there exists a negligible function $\mathsf{neg}(\cdot)$ such that for $\lambda \in \mathbb{N}$ it holds that*

$$\left| \Pr\left[D(1^\lambda, \Pi(F, S_b, X)) = b\right] - \frac{1}{2} \right| \leq \mathsf{neg}(\lambda),$$

where the probability is over the internal randomness of the scheme, the internal randomness of D and $b \leftarrow \{0,1\}$ chosen uniformly at random.

A Remark on the Condition in the Security Definition. We note that in Definition 14, given a set of shares $\Pi(F, S, X)$, it is possible to derive for any $X' \subseteq X$ the set of shares $\Pi(F, S, X')$ simply by removing the shares for parties not in X'. Feeding $\Pi(F, S, X')$ into RECON then gives $F(X', S)$ for any $X' \subseteq X$. Thus, in the security definition above, the condition that $F(X', S_0) = F(X', S_1)$ for all $X' \subseteq X$ is required to have a satisfiable assumption. Our definition states that this is the only requirement.

Two Relaxations of Definition 14. We remark that when the function F is defined to be the identity function over its second input parameter (i.e., $F(\cdot, S) = S$) we get the definition of Rudich secret sharing for NP of [25].[13] Moreover, when $M = 2^{\mathcal{P}_n}$ (i.e., the access structure includes all subsets of parties), the secret S is a description of a function and F is the universal circuit (i.e., $F(X, S) = S(X)$), then Definition 14 boils down to the definition of *function secret sharing* which we formalize next.

Definition 15 (Function secret sharing). *Let $\mathcal{F} = \{F \colon 2^{\mathcal{P}_n} \to \{0,1\}^*\}$ be a class of functions. A functional secret sharing scheme for $\mathcal{F}$ consists of a setup procedure SETUP and a reconstruction procedure RECON that satisfy the following requirements:*

1. $\mathsf{SETUP}(1^\lambda, F)$ *gets as input a function $F \in \mathcal{F}$, and distributes a share for each party. For $i \in [n]$ denote by $\Pi(F, i)$ the random variable that corresponds to the share of party p_i. Furthermore, for $X \subseteq \mathcal{P}_n$, we denote by $\Pi(F, X)$ the random variable that corresponds to the set of shares of parties in X.*
2. ***Completeness:*** $\mathsf{RECON}(1^\lambda, \Pi(F, X))$ *gets as input the shares of some subset X of parties, and outputs $F(X)$. More precisely,*

$$\Pr[\mathsf{RECON}(1^\lambda, \Pi(F, X)) = F(X)] = 1,$$

where the probability is over the internal randomness of the scheme and of RECON.

[13] [25] considered a uniform version of the above definition. We remark that our definitions from above can also be given in a uniform version and our results also apply to them (using ideas from [25]). For simplicity, we focus on the non-uniform versions.

3. ***Indistinguishability of the function:*** *For every probabilistic polynomial-time algorithm D, every equal size $F_0, F_1 \in \mathcal{F}$ and $X \subseteq 2^{\mathcal{P}_n}$ such that $F_0(X') = F_1(X')$ for all $X' \subseteq X$, there exists a negligible function $\mathsf{neg}(\cdot)$ such that for $\lambda \in \mathbb{N}$ it holds that*

$$\left|\Pr[D\left(1^\lambda, \Pi(F_b, X)\right) = b] - \frac{1}{2}\right| \leq \mathsf{neg}(\lambda),$$

where the probability is over the internal randomness of the scheme, the internal randomness of D and $b \leftarrow \{0,1\}$ chosen uniformly at random.

4.1 Functional Secret Sharing Is Equivalent to iO

In this section we state and prove our main result.

Theorem 4. *The following holds:*

1. *Function secret sharing (Definition 15) for polynomial-size circuits implies* iO *for polynomial-size circuits.*
2. iO *for polynomial-size circuits and one-way functions imply functional secret sharing (Definition 14) for access structures in* mNP *and functions computed by polynomial-size circuits.*

Recall that Definition 14 is a generalization of Definition 15. Thus, Theorem 4 implies that functional secret sharing is equivalent to function secret sharing and is equivalent to iO.[14]

Next, we provide a proof for each of the items in Theorem 4 separately.

Proof of Item 1 in Theorem 4. Given a circuit C with n inputs the indistinguishability obfuscator works as follows. We first run the $\mathsf{SETUP}(1^\lambda, C)$ procedure with the circuit C as input and get back a list of n shares $\Pi(C, 1), \ldots, \Pi(C, n)$. The obfuscation consists of these n shares.

To evaluate an obfuscated circuit at a point $x \in \{0,1\}^n$, we run $\mathsf{RECON}(1^\lambda, \Pi(C, x))$ and get a value y that we output. By the correctness of the functional secret sharing scheme, we have that $y = C(x)$, as required.

To prove security consider two equal size *functionally equivalent* circuits C_1 and C_2 and an adversary $\mathcal{A}$ that can distinguish their obfuscations with noticeable probability. Hence, $\mathcal{A}$ can distinguish secret shares corresponding to $\mathsf{SETUP}(1^\lambda, C_1)$ from secret shares corresponding to $\mathsf{SETUP}(1^\lambda, C_2)$. Since the circuits are equal size and functionally equivalent, this is a contradiction to the security guarantee of the function secret sharing scheme. ■

Proof of Item 2 in Theorem 4. We start with the description of the functional secret sharing scheme. For every $i \in [n]$, the *share* of party p_i is composed of 2 components: (1) $r_i \in \{0,1\}^\lambda$, an opening of a commitment to the value i,

[14] One of the directions requires one-way functions which can be relaxed to require a worst-case hardness assumption by [24].

and (2) an *obfuscated* circuit $\mathsf{iO}(C)$. The circuit C to be obfuscated has the following hardwired: the function F, 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 $r_1, \ldots, r_n$ of the commitments are *not* hardwired into the circuit. The input to the circuit C consists of alleged k openings $r'_{i_1}, \ldots, r'_{i_k}$ corresponding to a set of parties $X \in 2^{\mathcal{P}_n}$ denoted $\mathsf{p}_{i_1}, \ldots, \mathsf{p}_{i_k}$ 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 $F(X, S)$; otherwise, if any of the tests fail, the circuit C outputs NUL. The secret sharing scheme is formally described next.

Let iO be an efficient indistinguishability obfuscator (see Definition 7). Let $\mathsf{Com} \colon [2n] \times \{0,1\}^\lambda \to \{0,1\}^{q(\lambda)}$ be a string commitment scheme where $q(\cdot)$ is a polynomial (see Definition 3). Let $M \in \mathsf{NP}$ be an access structure.

The $\mathsf{SETUP}(1^\lambda, F, S)$ procedure. Gets as input a function F represented as a polynomial-size circuits, a secret S and does the following:

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 the circuit C from Fig. 4, where $C = C^{F, S, \mathsf{c}_1, \ldots, \mathsf{c}_n}$ has the function F, the secret S and the list of commitments $\mathsf{c}_1, \ldots, \mathsf{c}_n$ hardwired.
3. Set the share of party p_i to be $\Pi(S, i) = \langle r_i, \mathsf{iO}(C) \rangle$.

The $\mathsf{RECON}(X, w)$ procedure. Gets as input a non-empty subset of parties $X \subseteq \mathcal{P}_n$ together with their shares and a witness w of X for M.

The Circuit $C^{F, S, \mathsf{c}_1, \ldots, \mathsf{c}_n}(r'_1, \ldots, r'_n, w)$

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

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

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 one. That is, verify that $V_M(X, w) = 1$.
2. If any of the above tests fails, output NUL; otherwise, output $F(X, S)$.

Fig. 4. The circuit to be obfuscated as part of the secret shares.

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

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

Security. Fix two secrets S_0, S_1, a subset of parties X, and a function F such that $F(X', S_0) = F(X', S_1)$ for every $X' \subseteq X$. The proof of security follows by a sequence of hybrid experiments that can be found in the full version [26]. ■

Acknowledgments. We thank Moni Naor and the anonymous reviewers of TCC 2016A for helpful remarks.

References

1. Applebaum, B., Brakerski, Z.: Obfuscating circuits via composite-order graded encoding. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part II. LNCS, vol. 9015, pp. 528–556. Springer, Heidelberg (2015)
2. Barak, B., Garg, S., Kalai, Y.T., Paneth, O., Sahai, A.: Protecting obfuscation against algebraic attacks. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 221–238. Springer, Heidelberg (2014)
3. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S.P., Yang, K.: On the (im)possibility of obfuscating programs. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 1–18. Springer, Heidelberg (2001)
4. Beimel, A.: Secret-sharing schemes: a survey. In: Chee, Y.M., Guo, Z., Ling, S., Shao, F., Tang, Y., Wang, H., Xing, C. (eds.) IWCC 2011. LNCS, vol. 6639, pp. 11–46. Springer, Heidelberg (2011)
5. Beimel, A., Burmester, M., Desmedt, Y., Kushilevitz, E.: Computing functions of a shared secret. SIAM J. Discrete Math. **13**(3), 324–345 (2000)
6. Benaloh, J.C., Leichter, J.: Generalized secret sharing and monotone functions. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 27–35. Springer, Heidelberg (1990)
7. Blakley, G.R.: Safeguarding cryptographic keys. Proc. AFIPS Natl. Comput. Conf. **48**, 313–317 (1979)
8. Boneh, D., Sahai, A., Waters, B.: Functional encryption: definitions and challenges. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 253–273. Springer, Heidelberg (2011)
9. Boneh, D., Silverberg, A.: Applications of multilinear forms to cryptography. IACR Cryptology ePrint Archive, p. 80 (2002)
10. Boneh, D., Waters, B.: Constrained pseudorandom functions and their applications. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part II. LNCS, vol. 8270, pp. 280–300. Springer, Heidelberg (2013)
11. Boneh, D., Zhandry, M.: Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 480–499. Springer, Heidelberg (2014)

12. Boyle, E., Gilboa, N., Ishai, Y.: Function secret sharing. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 337–367. Springer, Heidelberg (2015)
13. Brakerski, Z., Rothblum, G.N.: Virtual black-box obfuscation for all circuits via generic graded encoding. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 1–25. Springer, Heidelberg (2014)
14. Coron, J., Lepoint, T., Tibouchi, M.: New multilinear maps over the integers. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9215, pp. 267–286. Springer, Heidelberg (2015)
15. Desmedt, Y.G., Frankel, Y.: Threshold cryptosystems. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 307–315. Springer, Heidelberg (1990)
16. Garg, S., Gentry, C., Halevi, S.: Candidate multilinear maps from ideal lattices. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 1–17. Springer, Heidelberg (2013)
17. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: 54th Annual IEEE Symposium on Foundations of Computer Science, FOCS, pp. 40–49 (2013)
18. Garg, S., Gentry, C., Sahai, A., Waters, B.: Witness encryption and its applications. In: Symposium on Theory of Computing Conference, STOC, pp. 467–476 (2013)
19. Gentry, C., Lewko, A.B., Sahai, A., Waters, B.: Indistinguishability obfuscation from the multilinear subgroup elimination assumption. IACR Cryptology ePrint Archive p. 309 (2014), to appear in FOCS 2015
20. Goldwasser, S., Gordon, S.D., Goyal, V., Jain, A., Katz, J., Liu, F.-H., Sahai, A., Shi, E., Zhou, H.-S.: Multi-input functional encryption. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 578–602. Springer, Heidelberg (2014)
21. Grigni, M., Sipser, M.: Monotone complexity. In: Proceedings of LMS Workshop on Boolean Function Complexity. pp. 57–75 (1992)
22. Håstad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A pseudorandom generator from any one-way function. SIAM J. Comput. **28**(4), 1364–1396 (1999)
23. Karchmer, M., Wigderson, A.: On span programs. In: 8th Annual Structure in Complexity Theory Conference, pp. 102–111 (1993)
24. Komargodski, I., Moran, T., Naor, M., Pass, R., Rosen, A., Yogev, E.: One-way functions and (im)perfect obfuscation. In: 55th Annual IEEE Symposium on Foundations of Computer Science, FOCS, pp. 374–383 (2014)
25. Komargodski, I., Naor, M., Yogev, E.: Secret-sharing for NP. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014, Part II. LNCS, vol. 8874, pp. 254–273. Springer, Heidelberg (2014)
26. Komargodski, I., Zhandry, M.: Cutting-edge cryptography through the lens of secret sharing. IACR Cryptology ePrint Archive p. 735 (2015)
27. Naor, M.: J. Cryptol. Bit commitment using pseudorandomness **4**(2), 151–158 (1991)
28. Naor, M.: Secret sharing for access structures beyond P (2006), slides: http://www.wisdom.weizmann.ac.il/~naor/PAPERS/minicrypt.html
29. Pass, R., Seth, K., Telang, S.: Indistinguishability obfuscation from semantically-secure multilinear encodings. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 500–517. Springer, Heidelberg (2014)
30. Sahai, A., Waters, B.: Slides on functional encryption (2008). http://www.cs.utexas.edu/~bwaters/presentations/files/functional.ppt

31. Sahai, A., Waters, B.: How to use indistinguishability obfuscation: deniable encryption, and more. In: Symposium on Theory of Computing, STOC, pp. 475–484 (2014)
32. Santis, A.D., Desmedt, Y., Frankel, Y., Yung, M.: How to share a function securely. In: Symposium on Theory of Computing, STOC, pp. 522–533 (1994)
33. Shamir, A.: How to share a secret. Commun. ACM **22**(11), 612–613 (1979)
34. Zhandry, M.: How to avoid obfuscation using witness PRFs. IACR Cryptology ePrint Archive p. 301 (2014). To appear. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016-A, Part II, LNCS 9563, pp. 421–448 (2016)