Span Programs and General Secure Multi-Party Computation

Ronald Cramer (ETH Zurich [*])
Ivan Damgård (Aarhus University [†] & BRICS
Ueli Maurer (ETH Zurich [‡])

Abstract

We present a new generic approach to verifiable secret sharing and to secure multi-party computation which applies to both the cryptographic and the information-theoretic setting. In contrast to previous results, achieving security when up to a certain fraction ($1/2$ or $1/3$) of the players cheat, security is achieved with respect to a general collection of subsets of the player set (called an adversary structure), any one of which may consist of cheaters.

For the information-theoretic setting we achieve security against any structure that is tolerable as proved by Hirt and Maurer, but our protocols are more efficient. In particular, we exhibit structures for which the efficiency improvement is super-polynomial. For the cryptographic setting, our results are the first achieving security for general adversary structures: secure multi-party computation is possible if and only if no two sets of the adversary structure cover the full player set.

The protocols are based on monotone span programs introduced by Karchmer and Wigderson which naturally define linear secret sharing schemes. We show how to make such schems verifyable and how to design a new type of span program with a special multiplication property, yielding general secure multi-party computation protocols. For tolerating active adversaries we further apply homomorphic commitment schemes and some new generic subprotocols for computing with committed values. We also introduce a new type of shared homomorphic commitment scheme that is both information-theoretically binding and hiding.

1 Introduction

Secure *multi-party computation* (MPC) can be defined as the problem of n players to compute an agreed function of their inputs in a secure way, where security means guaranteeing the correctness of the output as well as the privacy of the players' inputs, even when some players cheat. A key tool for secure MPC, interesting in its own right, is *verifiable secret sharing* (VSS): a dealer distributes a secret value s among the players, where the dealer and/or some of the players may be cheating. It is guaranteed that if the dealer is honest, then the cheaters obtain no information about s, and all honest players are later able to reconstruct s. Even if the dealer cheats, a unique such value s is reconstructable without the cheaters' help.

It is common to model cheating by considering an *adversary A* who may corrupt some of the players. One can distinguish between passive and active corruption. *Passive* corruption means that A obtains the complete information held by the corrupted players, but the players execute the protocol correctly. Hence the correctness of a protocol is not affected. *Active* corruption means that the adversary takes full control of the *corrupted* players. It is (at least initially) unknown to the honest players which subset of players is corrupted, but it is known that the adversary can corrupt only certain subsets (e.g. all subsets with cardinality less than $n/3$). Corrupted players will also be called *cheaters*.

Two basic models have been considered in the literature. In the *cryptographic* model, all players are assumed to have access to messages exchanged between players, and hence privacy and correctness can only be guaranteed in a cryptographic sense (e.g. by encryption), i.e. assuming that the adversary cannot solve some computational problem. Privacy and correctness here means that any adversary could, without carrying out a real protocol, efficiently simulate its entire view of the protocol in a manner indistiguishable from a real execution of the protocol. In the *information-theoretic* (abbreviated i.t., sometimes also called secure channels) model, it is assumed that the players can communicate over pairwise secure channels, and the privacy as well as the correctness can then be guaranteed even when the adversary has unbounded computing power.

The outline of the paper is as folows. In Section 2 we introduce some terminology and state the results. The theorems will be proved in later sections and in the Appendices.[1] Section 3 introduces span programs with multiplication and shows their relation to linear secret sharing schemes. Section 4 introduces a generalized type of commitments and some protocols for computing with committed values. The main secure MPC protocols are described in Section 5. Cryptographic commitments are discussed in Appendix 1

2 Related Work and Results of this Paper

The goals of this paper are two-fold. First, we describe a generic and structured framework for VSS and secure MPC that is based on linear secret sharing schemes and homomorphic commitments as the two basic primitives, and on a number of subprotocols based on these primitives. Second, the

[*] Dept. of Computer Science, ETH Zurich, CH-8092 Zurich, Switzerland. Email: cramer@inf.ethz.ch.
[†] Maths. & Comp. Sc. Dept., Ny Munkegade, Aarhus, Denmark. Email: ivan@daimi.aau.dk
[‡] ETH Zurich, Switzerland. Email: maurer@inf.ethz.ch.

[1] Several general definitions of secure multiparty computation have been proposed (see e.g. [3, 26, 10]), technically different, but all designed to capture the same basic intuition. While either definition could have been used for a formal proof of security of our protocols, any such proof would by far exceed the space limitations. Instead, we have included sketches of proofs, generic enough to fit any of the definitions.

paper proposes a number of concrete technical realizations of the primitives and generic subprotocols which lead to secure MPC protocols that are at least as efficient, but typically more efficient than previously proposed protocols.

The classical results for the information-theoretic model due to Ben-Or, Goldwasser and Wigderson [7] and Chaum, Crépeau and Damgård [12] state that every function can be securely computed if and only if less than $n/2$ passive or less than $n/3$ active cheaters are present.[2] These results were generalized by Hirt and Maurer [24] who considered as the potential adversaries general sets of subsets of the player set, not necessarily specified by their cardinality (see Section 2.1).

The most general previous result for the cryptographic model is by Goldreich, Micali and Wigderson [23] who showed that any minority (i.e. $< n/2$) of active cheaters can be tolerated, assuming that trapdoor one-way permutations exist.

2.1 Players and General Adversaries

Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be the set of players. We call a set of subsets of $\mathcal{P}$ a *structure*. If it is closed under taking supersets (subsets), then the structure is called monotone (anti-monotone). An *access structure* Γ in secret sharing[3] is a monotone structure and, using terminology from [24], an *adversary structure* $\mathcal{A}$ is an anti-monotone structure and is to be understood as meaning that the adversary (for the passive or the active model) can choose to corrupt the players in exactly one of the sets contained in $\mathcal{A}$.

Let Q^2 (and Q^3) be the conditions on a structure that no two (no three) of the sets in the structure cover the full player set $\mathcal{P}$. The result of [24], which apply only to the i.-t. setting, can be stated as follows: for an adversary structure $\mathcal{A}$, secure MPC is possible if and only if the adversary structure satisfies Q^2 in the case of passive and Q^3 in the case of active corruption. Note that the threshold results of [7], [12], [23] and [30] are special cases, where the adversary structure contains all sets of size less than $n/2$ or $n/3$.

This model not only yields more general, but also strictly stronger results. For instance, consider the case of 6 players: $\mathcal{P} = \{P_1, \ldots, P_6\}$ and active corruption. The best threshold-type results states that the corruption of only a single player can be tolerated. In constrast, [24] or this paper yields for example a protocol secure against the structure $\{\{P_1, P_2, P_3\}, \{P_1, P_4\}, \{P_2, P_5\}, \{P_2, P_6\}, \{P_3, P_4\}\}$, where each player is tolerated to cheat even together with one or two other players. As an asymptotic example, consider two groups X and Y of m players each ($n = 2m$) where the players are on friendly terms within each group but tend to distrust players in the other group. This motivates the following assumption: a group of active cheaters can consist of at most $9m/10$ players from only X or only Y, *or* it can consist of less than $m/5$ players coming from both X and Y. It is easily verified that the Q^3 condition is satisfied.

2.2 Preliminaries, Terminology, and Notation

We assume throughout (without loss of generality, and in accordance with the previous literature) that the function to be computed by $\{P_1, \ldots, P_n\}$ is given as an *arithmetic circuit* C of size $|C|$ over some finite field K, consisting of addition and multiplication gates. We further assume that an adversary structure $\mathcal{A}$ to be passively or actively tolerated is specified, and all protocols and subprotocols of this paper will be secure for (at least) this structure. An adversary that corrupts at most a set contained in $\mathcal{A}$ will be called *admissible*. Any player under control of the adversary is called *corrupt*, other players are called *honest*. The adversary is assumed to be static, but for the i.t. setting our results also hold for an adversary who chooses adaptively during the protocol execution which players to corrupt.

We assume synchronous bilateral communication between players, where in the i.t. setting the channels are assumed to be secure like in all the previous literature. The protocols make use of a broadcast channel. Such a channel is generally not available, but it can efficiently be simulated using the protocol of [19] that is secure against any given Q^3 adversary structure. Note that the protocols of [24] also imply a broadcast protocol but generally with complexity exponential in n; in contrast, the complexity of the protocols in [19] are polynomial in n for any given Q^3 adversary structure for which membership can be efficiently decided.

Structures are conveniently characterized by a monotone Boolean function $f : \{0,1\}^n \to \{0,1\}$, by identifying subsets of $\mathcal{P}$ with their characteristic vectors[4]. A set $S \subset \mathcal{P}$ for which $f(S) = 0$ (or $f(S) = 1$) is said to be *rejected* (or *accepted*) by f, and a structure is said to be rejected (accepted) by f if all its sets are. The function f hence defines naturally a monotone access structure (anti-monotone adversary structure), denoted Γ_f ($\mathcal{A}_f$), consisting of those sets accepted (rejected) by f.

A structure $\mathcal{A}$ is *maximal*-Q^2 (maximal-Q^3) if there is no structure $\mathcal{B}$, such that $\mathcal{A}$ is a proper subset of $\mathcal{B}$ and $\mathcal{B}$ is Q^2 (Q^3). A function f is said to be Q^2 (Q^3) or maximal-Q^2 (Q^3) if $\mathcal{A}_f$ is.[5]

2.3 Results on Monotone Span Programs

We will make use of *monotone span programs* (MSP), a linear algebra based model of computation introduced by Karchmer and Wigderson [25]. A precise definition of an MSP is given later, but to state our results it suffices to note that an MSP computes a monotone Boolean function using linear algebra over a finite field K, and that the size of an MSP is defined naturally. As shown in [25], an MSP yields a linear secret sharing scheme whose efficiency is directly related to the size of the MSP.

It is well-known that such an MSP exists for every monotone access structure. One of the goals of this paper is to extend the usefulness of MSPs by constructing homomorphic cryptographically (i.t.) secure VSS schemes from any MSP computing a Q^2 (Q^3) function, where in the cryptographic case we also need a homomorphic commitment scheme.

Going one step further, as a means for applying MSPs to MPC (both settings), we introduce two new types of MSP, called MSP *with multiplication* (with *strong* multiplication).[6]

[2] When a broadcast channel is available, then one can tolerate less than $n/2$ active cheaters [29, 30].

[3] This is the collection of subsets of the players whose joint shares in the shared secret gives full information about the secret.

[4] The characteristic vector of a set S is a vector in $\{0,1\}^n$ whose i-th component is 1 if and only if $P_i \in S$

[5] A maximal-Q^2 function is also called self-dual.

[6] We draw on two ideas appearing in work by Gennaro, Rabin and Rabin [22] on threshold-type secure MPC: the observation that in [7] the product

We first state our results on the existence of such MSPs and upper bounds on their size, which may be of independent interest. Their implications to secure MPC are discussed in the following two subsections.

THEOREM 1 *A monotone Boolean function f is computed by an MSP with (strong) multiplication over some finite field if and only if f is Q^2 (Q^3).*

We define a *majority-accepting formula* to be a Boolean formula with unbounded fan-in and no constants, where all operations are threshold functions T_m^t with threshold $t-1 < m/2$. A *1/3-accepting formula* is the same [7], except we require that $t-1 < m/3$. For any monotone Boolean function f, let $\phi_{ma}(f)$ ($\phi_{1/3}(f)$) be the size of the smallest majority-accepting (1/3-accepting) formula computing f. We have:

THEOREM 2 *The class of Q^2-monotone (Q^3-monotone) functions coincides with the class of functions computable by majority accepting (1/3-accepting) formulæ. Any Q^2-monotone (Q^3-monotone) function f is computable by an MSP with (strong) multiplication of size at most $\phi_{ma}(f)$ (at most $\phi_{1/3}(f)$) and over any field K with $|K| > \phi_{ma}(f)$ (with $|K| > \phi_{1/3}(f)$).*

There are a number of other constructions of such MSPs, in particular also for the field $GF(2)$, which is attractive because in applications functions are often given as binary circuits.

2.4 The Information-Theoretic Model

The adversary structures $\mathcal{A}$ tolerable in i.t. secure MPC (and hence in VSS) were characterized completely in [24][8], but these protocols are generally polynomial in $|\mathcal{A}|$ rather than in n, where $|\mathcal{A}|$ is typically exponential in n. Using MSPs, we explore for which adversary structures there exist efficient protocols.[9] At the same time, this MSP-based approach is not recursive, unlike that of [24], and we believe that it captures naturally the essence of arithmetic circuit-based secure MPC.

THEOREM 3 *For any field K and any Q^3 adversary structure $\mathcal{A}$, there exists an information-theoretically secure VSS protocol secure against $\mathcal{A}$. The protocol can be constructed from any MSP $\mathcal{M}$ over K computing a Q^3 function f rejecting $\mathcal{A}$, and its complexity is polynomial in the size of $\mathcal{M}$ and $\log|K|$.*

THEOREM 4 *For any field K, any arithmetic circuit C over K, any passive Q^2 (active Q^3) adversary structure $\mathcal{A}$, and for any MSP $\mathcal{M}$ over K with the (strong) multiplication property computing a function f rejecting $\mathcal{A}$, there is an MPC protocol computing C, information-theoretically secure against $\mathcal{A}$, with complexity polynomial in $|C|$, $\log|K|$, and the size of $\mathcal{M}$.*

of two shared values is defined as a linear function of the products of the shares, and the idea (independently discovered in the course of our research) of basing cryptographic VSS on homomorphic cryptographic commitments.

[7] Single literals are also considered to be members of both classes.

[8] The protocols proposed in [24] rely on applying a protocol for the threshold case (e.g. [7, 22]) recursively, so that subsets of players together run threshold protocols to simulate virtual players in higher-level protocols.

[9] Like in [7] and [24], our protocols have zero probability of successful cheating by an active adversary.

The complexities of the protocols of [24] match the upper bounds stated in Theorem 2; in fact that construction is based on majority and 1/3-accepting formulæ. Thus our protocols are guaranteed to be as efficient as those of [24] but can be expected to be substantially more efficient in practice.[10] Moreover, there are adversary structures for which our protocol for VSS is more efficient than that of [24] by a super-polynomial factor. This follows from:

THEOREM 5 *There exists families $\{f_n\}$ and $\{g_n\}$ of monotone Boolean functions on n variables such that every f_n is maximal Q^2 and every g_n is maximal Q^3. Each f_n (g_n) can be computed by an MSP over $GF(2)$ of size $O(n)$, but any majority accepting ($1/3$-accepting) formula computing f_n (g_n) has size $n^{\Omega(\log n)}$.*

In fact, we prove the stronger result that this theorem holds even with the formulae replaced by monotone Boolean circuits. It is open whether a similar separation result holds if one requires the MSP is required to have the multiplication property. We conjecture that the answer is yes, even when the strong multiplication property is required. This would imply that also our secure MPC protocol is more efficient than those of [24] by a super-polynomial factor. In support of this conjecture we note [16]:

PROPOSITION 1 *There exists a family of monotone Boolean functions $\{h_n\}$ on n variables such that each h_n is Q^2. Each h_n can be computed by an MSP with multiplication over $GF(2)$ of size n, but a monotone Boolean circuit computing h_n must have size $n^{\Omega(\log n)}$.*

This proposition shows that for the adversary structure rejected by h_n, our protocol is efficient while that of [24] is not. However, because the functions h_n are not maximal-Q^2, we cannot exclude that there could exist other functions rejecting the same structure and leading to efficient protocols using [24], but these would be non-obvious.

2.5 The Cryptographic Model

Generalizing the threshold result of [23], we show that any Q^2 active adversary structure can be tolerated, under the same assumption that trapdoor one-way permutations exist. Moreover, under stronger cryptographic assumptions our protocols are more efficient than all previous protocols, even for threshold structures. Furthermore, the protocol can be based on any secret sharing scheme $\mathcal{S}$, linearity is not needed. The complexity of $\mathcal{S}$ is defined as the complexity of computing the shares.

THEOREM 6 *Let C be an arithmetic circuit over a finite field K, let $\mathcal{A}$ be an active Q^2 adversary structure, and let $\mathcal{S}$ be a secret sharing scheme over K for which all sets in $\mathcal{A}$ are non-qualified and all complements of sets in $\mathcal{A}$ are qualified. If trapdoor one-way permutations exist, then there exists a secure MPC protocol computing C with complexity polynomial in $|C|$, the complexity of $\mathcal{S}$, and the security parameter k.*

[10] Beaver and Wool [4] propose a protocol (only for the i.t. passive setting) with complexity linear in the size of the adversary structure. Our complexity comparison with [24] extends to a comparison with [4].

This result requires general zero-knowledge proof techniques and so is of little practical relevance. However, if $\mathcal{S}$ is linear and in addition one-way group homomorphisms with specific extra properties exist, so called q-one-way group homomorphisms [14], then very efficient protocols can be built. Particular assumptions sufficient for the existence of q-one way group homomorphisms include the RSA assumption, hardness of discrete logarithm in a group of prime order, or polynomial security of Diffie-Hellman encryption. As an example of what one obtains for the most efficient implementation of the primitives, we state the following:

THEOREM 7 *Let C, $\mathcal{A}$ and f be as in Theorem 6 with $K = GF(q)$ for a k-bit prime q. If Diffie-Hellmann based probabilistic encryption in a group of order q is polynomially secure, then there exists a secure MPC protocol computing C with communication complexity $O(k\,z^2\,|C|)$, where z is the size of an MSP with multiplication over $GF(q)$ computing f.*

Even for the threshold case, our protocol yields substantial efficiency improvements over the best known protocols. An MSP for the function rejecting all minorities has only size $z = n$. The best previous protocols (see e.g. [14]) have similar communication complexity, where however $|C|$ would have to be replaced by the number of *bit* instead of $GF(q)$-operations.

3 Span Programs with Multiplication

We introduce and construct monotone span programs (MSP) *with multiplication*, a new type of MSP. MSPs are a linear algebraic model of computation introduced in [25] and provide the most powerful known method for constructing linear secret sharing schemes[11] (LSSS) for arbitrary access structures, a central primitive in the design of many cryptographic protocols. An LSSS has the property that the holders of shares in two secrets can securely compute shares in the *sum* of the secrets by adding their shares locally.

Our new MSPs offer a significant advantage over the MSPs from [25]: due to an additional multiplication property they facilitate the secure MPC of the *product* of two shared secrets. We also define a *strong* multiplication property of an MSP in order to be able to tolerate active cheaters in the i.t. model. These multiplication properties can be seen as an abstraction capturing what is needed for efficient general secure MPC.

3.1 Notation and Definitions

The following definitions are in addition to those of Section 2. Let $f : \{0,1\}^n \longrightarrow \{0,1\}$, be a monotone Boolean function. The dual function of f, denoted $f^* : \{0,1\}^n \longrightarrow \{0,1\}$, is defined by $f^*(x) = f(x \oplus \mathbf{1}) \oplus 1$ for all $x \in \{0,1\}^n$, where $\mathbf{1}$ denotes the all-one bit string and $\oplus$ denotes (bit-wise) exclusive-or. A function f is called *self-dual* if $f = f^*$.

The dual Γ^* of an access structure Γ is defined by $A \in \Gamma^*$ if and only if $A^c \notin \Gamma$. Hence we have $(\Gamma_f)^* = \Gamma_{f^*}$. If $A \in \Gamma$, then A is called *qualified*, otherwise it is called *non-qualified*.

The dimension of a finite-dimensional vector space V over a field K is denoted $\dim_K V$. For a subspace W of V, $W^\perp$ denotes the orthogonal complement of W in V. Relative to a standard basis, $\langle \mathbf{x}, \mathbf{y} \rangle$ denotes the inproduct of $\mathbf{x}, \mathbf{y} \in V$. For $V = K^d$, $\mathbf{x} = (x_1, \ldots, x_d) \in V$ and $\mathbf{y} = (y_1, \ldots, y_d) \in V$, $\mathbf{x} * \mathbf{y}$ denotes $(x_1 y_1, \ldots, x_d y_d) \in V$.

If V is a finite dimensional vector space over a field K, then $\dim_K V$ denotes its dimension. If W is a subspace of V, then $W^\perp$ denotes the orthogonal complement of W in V. Relative to a standard basis, $\langle \mathbf{x}, \mathbf{y} \rangle$ denotes the inproduct of $\mathbf{x}, \mathbf{y} \in V$. For $V = K^d$, $\mathbf{x} = (x_1, \ldots, x_d) \in V$ and $\mathbf{y} = (y_1, \ldots, y_d) \in V$, $\mathbf{x} * \mathbf{y}$ denotes $(x_1 y_1, \ldots, x_d y_d) \in V$.

For a vector $\mathbf{v} = (v_1, \ldots, v_e) \in K^e$, let $\mathbf{v} \otimes \mathbf{v}$ (tensor product) denote the e by e *matrix* whose i-th column is $v_i \cdot \mathbf{v}$ for $i = 1, \ldots, e$. By abuse of notation, $\mathbf{v} \otimes \mathbf{v}$ will sometimes refer to the e^2-*vector* $(v_1 v_1, \ldots, v_1 v_e, \ldots, v_e v_1, \ldots, v_e v_e)$. In that case, we will speak of "the vector $\mathbf{v} \otimes \mathbf{v}$". For $s \in K$ and $\mathbf{w} \in K^{e-1}$, $(s, \mathbf{w})$ denotes the vector in K^e with first coordinate s and remaining ones those of $\mathbf{w}$. The transpose of a matrix M is denoted M^t.

We view M as a K-linear map defined by its columns: $M\mathbf{b}$ means the linear combination of the columns of M whose weights are the coordinates of $\mathbf{b}$. If M has d rows and e columns, then for "M applied to $\mathbf{b} \in K^e$", we write $M\mathbf{b}$. A similar definition applies to $M^t\mathbf{c}$ with $\mathbf{c} \in K^d$. Let $M_\otimes$ denote the matrix obtained by replacing each row $\mathbf{v}$ in M by the vector $\mathbf{v} \otimes \mathbf{v}$. Note that $M_\otimes$ has d rows, and e^2 columns. The image of M is denoted Im M and its kernel (null-space) by Ker M. Finally, a useful fact from elementary linear algebra is that Im $M^t = (\text{Ker } M)^\perp$ if K is a finite field.

3.2 Monotone Span Programs

Let K be a finite field and let M be a matrix with entries in K, having d rows and e columns.[12] Let n be a positive integer and let $\psi : \{1, \ldots, d\} \longrightarrow \{1, \ldots, n\}$ be a surjective function. The matrix M is *labelled* in the sense that ψ assigns the label $\psi(k)$, the index of a player, to the k-th row of M, $1 \le k \le d$. Often, we will treat M just as a matrix, ignoring the labelling.

For $1 \le i \le n$, M_i denotes the matrix consisting of the d_i rows in M labelled with i. Similarly, for $\emptyset \ne A \subset \{1, \ldots, n\}$, M_A denotes the matrix consisting of all rows in M labelled by elements $j \in A$, and d_A denotes the number of rows in M_A. For a vector $\mathbf{s} = (s_1, \ldots, s_d) \in K^d$, we define $\mathbf{s}_i$ and $\mathbf{s}_A$ similarly.

Let $\boldsymbol{\epsilon} = (1, 0, \cdots, 0) \in K^e$ be the so-called target vector.[13] Formally, an MSP is a triple (K, M, ψ), defined as above. An MSP is said to compute the monotone Boolean function $f : \{0,1\}^n \longrightarrow \{0,1\}$ defined by

$$f(x_1, \ldots, x_n) = 1 \iff \boldsymbol{\epsilon} \in \text{Im } M_A^t,$$

where $A = \{1 \le i \le n | x_i = 1\}$. It is well-known that any monotone Boolean function can be computed by an MSP.

3.3 Secret Sharing based on MSPs

Let $f(x_1, \ldots, x_n)$ be a monotone Boolean function and let (K, M, ψ) be a (publicly known) MSP computing it. A perfect LSSS with access structure Γ_f is constructed as follows [25].

[11] Recall that in an LSSS the dealer chooses a set of random coefficients in K, and each share is obtained as a certain linear function of the secret and the random coefficients.

[12] We may assume without loss of generality that $e \le d$. Keeping only a collection of columns (including the first) that span the column space does not affect the multiplication property defined later.

[13] We can take any non-zero vector as the target vector by applying a suitable injective linear map on the rows.

Share Distribution: To share a secret value $s \in K$, the dealer chooses $\boldsymbol{\rho} \in K^{e-1}$ at random and writes $\mathbf{b} := (s, \boldsymbol{\rho})$. For $i = 1, \ldots, n$, he sends the share $\mathbf{s_i} := M_i \mathbf{b}$ privately to player P_i. Note that $\mathbf{s}_i \in K^{d_i}$.

Reconstruction: Let $A \subset \{1, \ldots, n\}$ satisfy $f(A) = 1$, and let $\boldsymbol{\lambda}_A$ satisfy $M_A^t \boldsymbol{\lambda}_A = \boldsymbol{\epsilon}$. Then we have $s = \langle \boldsymbol{\lambda}_A, \mathbf{s}_A \rangle$. Note that $\boldsymbol{\lambda}_A, \mathbf{s}_A \in K^{d_A}$.

For completeness, we prove the scheme correct and private. Correctness: For A with $f(A) = 1$ we have $s = \langle \boldsymbol{\epsilon}, \mathbf{b} \rangle = \langle M_A^t \boldsymbol{\lambda}_A, \mathbf{b} \rangle = \langle \boldsymbol{\lambda}_A, M_A \mathbf{b} \rangle = \langle \boldsymbol{\lambda}_A, \mathbf{s}_A \rangle$. Privacy: If $f(A) = 0$, then $\boldsymbol{\epsilon} \notin \text{Im } M_A^t$. By the fact that $\text{Im } M_A^t = (\text{Ker } M_A)^\perp$, this implies that Ker M_A contains a vector with non-zero first coordinate. Hence, the equation $M_A \mathbf{b}' = \mathbf{s}_A$, which has solution space equal to $\mathbf{b} + \text{Ker } M_A$, has the same number of solutions for each possible choice of a secret s'. The argument is completed by noting that $\boldsymbol{\rho}$ is chosen at random by the dealer.

We will always assume that an LSSS is *non-degenerate* in the sense that the full player set $\mathcal{P}$ is qualified. It is known [5] that the above LSSS construction is completely general. More precisely, given an LSSS where the total size of the shares is $d \cdot \log |K|$, there exists an MSP of size d whose associated LSSS is the given one. Hence we will speak of the MSP *underlying* an LSSS and of the LSSS *induced* by an MSP.

3.4 Adding the Multiplication Property

We introduce a new class of MSPs and induced LSSS in which the players holding shares in two values can (interactively, but easily) compute shares in the product of the two values, without leaking any information. This may well be the most general definition of an LSSS suitable for this task. We first consider an LSSS and later establish the connection with MSPs.

Let $(\mathcal{S}, \mathcal{S}_\mu)$ be a pair of LSSSs with access structures Γ and Γ_μ. Write (K, M, ψ) and (K, M_μ, ψ) for their underlying MSPs, and e and e_μ for the respective number of rows. Note that the MSPs have the same labelling fuction, and that in particular this implies that their respective sizes are identical.

DEFINITION 1 *$(\mathcal{S}, \mathcal{S}_\mu)$ is called an LSSS* with multiplication *if for all $s, s' \in K$ and for all $\boldsymbol{\rho}, \boldsymbol{\rho}' \in K^{e-1}$, there exists $\boldsymbol{\rho}_\mu \in K^{e_\mu - 1}$ such that $M(s, \boldsymbol{\rho}) * M(s', \boldsymbol{\rho}') = M_\mu(ss', \boldsymbol{\rho}_\mu)$. An LSSS with* strong multiplication *is one for which additionally $\Gamma^* \subset \Gamma_\mu$ holds.*

It will be shown in Section 5.1 how this multiplication property facilitates the secure multi-party computation of the product of two shared values s and s'. Intuitively, due to the LSSS $\mathcal{S}_\mu$ the product ss' is a linear function of the products of corresponding shares s_k and s'_k $(1 \leq k \leq d)$ of s and s'. Hence an $\mathcal{S}$-sharing of ss' can be computed as a linear combination of $\mathcal{S}$-sharings of the values $s_k s'_k$. In order to tolerate active adversaries in the i.t. scenario, the strong multiplication property will be crucial.

We now further explore the connection with MSPs.

DEFINITION 2 *An MSP (K, M, ψ) computing the function f is* with multiplication *if there exists a vector $\mathbf{r}$, called the* recombination vector*, such that for all $\mathbf{b}, \mathbf{b}'$ we have*

$$\langle \mathbf{r}, M\mathbf{b} * M\mathbf{b}' \rangle \;=\; \langle \boldsymbol{\epsilon}, \mathbf{b} \rangle \cdot \langle \boldsymbol{\epsilon}, \mathbf{b}' \rangle.$$

If in addition for all A with $f(A) = 0$ there exists such a recombination vector $\mathbf{r}$ with $\mathbf{r}_A = \mathbf{0}$, then the MSP is with strong multiplication.

The following two propositions show that without loss of generality or efficiency we can focus on MSPs with multiplication.[14]

PROPOSITION 2 *An MSP (K, M, ψ) is with multiplication if and only if the vector $\boldsymbol{\epsilon} \otimes \boldsymbol{\epsilon}$ is in Im $M_\otimes^t$. Furthermore, any vector $\mathbf{r}$ satisfying $M_\otimes^t \mathbf{r} = \boldsymbol{\epsilon} \otimes \boldsymbol{\epsilon}$ is a recombination vector.*[15] *Let f be the function computed by the MSP. The MSP is with strong multiplication if and only if for any set A with $f(A) = 0$ the condition holds with respect to the subset of the rows of M obtained by removing those labelled with elements from A.*

The proof appears in Appendix 2 and is by using uniqueness of algebraic normal form. In Appendix 2 we also state a corollary establishing a connection with linear codes.

Note that Proposition 2 also provides an efficient way of testing whether a given MSP has multiplication, and if it does, to compute efficiently a recombination vector. For testing strong multiplication, it seems that we have to run through all possible non-qualified sets. But if we are promised that a given MSP has strong multiplication, then for any A with $f(A) = 0$ we can efficiently a find recombination vector $\mathbf{r}$ with $\mathbf{r}_A = \mathbf{0}$.

PROPOSITION 3 *Let $(\mathcal{S}, \mathcal{S}_\mu)$ be an LSSS with* (strong) multiplication *over K. Then the MSP $\mathcal{M}$ underlying $\mathcal{S}$ has (strong) multiplication. Conversely, if (K, M, ψ) is an MSP with (strong) multiplication, then $(\mathcal{S}, \mathcal{S}_\mu)$ is an LSSS with* (strong) multiplication *over K, where $\mathcal{S}$ and $\mathcal{S}_\mu$ are the LSSS induced by (K, M, ψ) and $(K, M_\otimes, \psi)$, respectively.*

PROOF. First claim: Let $\mathbf{r}$ be any fixed linear combination over any complete set of shares resulting from $\mathcal{S}_\mu$ that yields the secret. Then, by the assumptions on $(\mathcal{S}, \mathcal{S}_\mu)$ it follows that $\mathbf{r}$ is a recombination vector for (K, M, ψ). Second claim: we have $M\mathbf{b} * M\mathbf{b}' = M_\otimes(\mathbf{b} \otimes \mathbf{b}')$, where $\mathbf{b} \otimes \mathbf{b}'$ is seen as a vector. $\mathcal{S}_\mu$ is non-denegerate by the fact that $\boldsymbol{\epsilon} \otimes \boldsymbol{\epsilon} \in$ Im $M_\otimes^t$, following from Proposition 2. The claim about strong multiplication is immediate from the above.

$\triangle$

3.5 Existence and Bounds

We prove Theorems 1 and 2 from Section 2. Theorem 5 and Proposition 1 are proved in Appendix 2. We refer to Section 2 for definitions.

Proofs of Theorems 1 and 2.

We first prove the forward direction of Theorem 1 and begin with the Q^3-case. Let (K, M, ψ) be an MSP with strong multiplication computing $f(x_1, \ldots, x_n)$. Suppose f is not Q^3. Then there exist disjoint sets A, B, C with $f(A) = f(B) = f(C) = 0$ and $A \cup B \cup C = \{1, \ldots, n\}$. Let $\boldsymbol{\kappa}$ be such that its first coordinate is equal to 1 and $M_B \boldsymbol{\kappa} = \mathbf{0}$. Such $\boldsymbol{\kappa}$ exists by

[14] Furthermore, it is easy to show by elementary linear algebra that the (strong) multiplication property is preserved under linear transformations on the rows of the MSP.

[15] When passing from M to $M_\otimes$ we leave the labelling unchanged.

the last remark in Section 3.1 and the fact that $f(B) = 0$. Let $\mathbf{r}$ be a recombination vector with $\mathbf{r}_C = \mathbf{0}$. By definition it follows that for all $\mathbf{b}$ we have $\langle M\boldsymbol{\kappa} * M\mathbf{b}, \mathbf{r}\rangle = s$, where s is the first coordinate of $\mathbf{b}$. But by the choice of $\boldsymbol{\kappa}$ and $\mathbf{r}$, it follows that$\langle (M\boldsymbol{\kappa} * M\mathbf{b})_A, \mathbf{r}_A\rangle = s$ also holds. This means that in the secret sharing scheme from Section 3.2 the shares given to A uniquely determine the secret. This is a contradiction with $f(A) = 0$, and hence f is Q^3. The Q^2-case is proved similarly (just leave out C from the above proof). This completes the proof of the forward direction of Theorem 1.

We now address how to construct new MSPs with (strong) multiplication) from old. Let f, $g_1, \ldots, g_n$ be any monotone Boolean functions such that each of them can be computed by MSPs with (strong) multiplication. Write $\mathcal{F} = (K, F, \psi)$, $\mathcal{G}^{(i)} = (K, G^{(i)}, \psi_i)$, $i = 1, \ldots, n$, for their respective MSPs. As before write F_i for the rows of F labelled with i. The following propositions[16] are proved in Appendix 2.

PROPOSITION 4 *$f(g_1, \ldots, g_n)$ is computable by an MSP $\mathcal{M} = (K, M, \chi)$ with (strong) multiplication. Furthermore, size($\mathcal{M}$) is equal to $\sum_{i=1}^{n}$ rows(F_i) $\cdot$ size($\mathcal{G}^{(i)}$).*

PROPOSITION 5 *Every Q^2-monotone (Q^3-monotone) function f can be computed by an unbounded fan-in Boolean formula consisting of majority accepting (1/3-accepting) gates.*

The first part of following Lemma follows from [31], and the second part is due to [22] (except for the strong multiplication case), and it is cast here in our terminology. The proof is given in the Appendix and uses observations about polynomial interpolation.

LEMMA 1 *Let T_n^t be an arbitrary threshold function. Then T_n^t can be computed by an MSP (K, M, ψ) of size n, provided that $|K| > n$ if $t > 1$. If additionally T_n^t is majority accepting (1/3-accepting), then the MSP is with (strong) multiplication.*

We now prove Theorem 2 and we finalize the proof of Theorem 1. Let f be Q^2-monotone (Q^3-monotone). Let Φ be *any* unbounded fan-in formula with majority accepting (1/3-accepting) gates computing f. The existence of such a formula Φ is guaranteed by Proposition 5. By applying Proposition 4 recursively (applying Lemma 1 to the gates of Φ and choosing the field K large enough), we obtain an MSP with (strong) multiplication computing f, with size equal to the number of input wires of Φ.

Finally, it is now clear that any unbounded fan-in formula with majority accepting (1/3-accepting) gates computes a Q^2-monotone (Q^3-monotone) function since it can be computed by an MSP with (strong) multiplication.

4 Generalized Commitments and VSS

In Section 4.1 we define homomorphic commitment schemes in a way that allows us to treat the i.t. and the cryptographic setting within the same framework. Commitments will be our basic tool for converting an MPC protocol secure against passive adversaries into one secure against active adversaries (see Section 5.2): each player is committed to his shares of intermediate resuts of the computation. However, each player will have to prove that he is performing all local computations correctly, and hence the players must be able to perform these computations on a given player's commitments. The homomorphic property assures that addition and multiplication with constants can be performed directly on the commitments. For performing the other operations on committed values we construct three subprotocols: CTP for transfering a committed value from one player to another, CSP for sharing a committed value in a verifiable way (which in particular implies VSS), and CMP for multiplying committed values, assuming in addition that the underlying LSSS has strong multipication.

We will define security of the protocols relative to an adversary structure $\mathcal{A}$, meaning that the security properties are required to hold against any adversary corrupting at most a set in $\mathcal{A}$. Such an adversary is called *admissible*.

In Section 4.2 we describe a new homomorphic commitment scheme that is both information-theoretically binding and hiding with respect to $\mathcal{A}$ if $\mathcal{A}$ is Q^3. In Appendix 1 we discuss cryptographic homomorphic commitments.

4.1 Commitments and Related Protocols

In its most general form, a commitment scheme is a pair of protocols, COMMIT and OPEN, among a possibly corrupted dealer (or committer) D and a set $P_1, \ldots, P_n$ of players. In our context, D is always one of the players in $P_1, \ldots, P_n$, but the protocols apply to, and are therefore described in, the more general context. Most known commitment protocols do not involve any interaction between $P_1, \ldots, P_n$ and are hence described as taking place between the committer and just a single verifier. Our cryptographic commitments will also be of this special form, where each of the players independently plays the role of the verifier, but for the i.t. setting the interaction between the players $P_1, \ldots, P_n$ of the MPC will be crucial.

The protocol COMMIT allows D to commit to a value $a \in K$, and the protocol OPEN allows D to later open the commitment. Both protocols can end in all (honest) players rejecting the outcome, in which case D is deemed corrupt. By the *commitment* by D to a value a, denoted $[a]_D$, we will mean the total information distributed to the correct players during a (successful) execution of COMMIT.[17]

Let $\mathcal{A}$ be an adversary structure. The commitment scheme is *$\mathcal{A}$-secure* if the following security requirements hold for any admissible adversary:

(i) For COMMIT and OPEN, if one honest player rejects, then all honest players reject. If D is honest, then no honest player rejects.

(ii) *Binding*: COMMIT is either rejected, or D is committed to a unique value.[18] OPEN is either rejected, or the committed value is correctly opened.

[16]The claim also holds if we disregard the multiplication property. Viewed in that way, our result is a slight improvement for the MSP complexity upper bound given in [28], Theorem 3.4, 4-th claim. Moreover, we incorporate multiplication.

[17]Commitments can be probabilistic. Then $[a]_D$ corresponds to one of many possible sets of values committing D to a.

[18]For both settings we only consider schemes that are unconditionally binding, but cryptographically binding schemes could also be considered.

(iii) *Hiding*: If D is honest and commits to value a, any admissible adversary learns nothing about a from his view of the COMMIT protocol.[19]

It is well-known that commitments in the conventional two-party setting can be unconditionally (i.t.) hiding or binding, but not both. In contrast, our "interactive" commitments for the i.t. setting are i.t. hiding *and* binding.

All our commitment schemes are *homomorphic* in the following sense: When D is committed to values a and b by commitments $[a]_D$ and $[b]_D$, respectively, then the players can, without interaction, compute commitments to $a + b$, $-a$, or ca for any given constant $c \in K$ such that an honest D is able to open them without giving any additional information to any admissible adversary except the opened value.[20] We will denote these commitments by $[a + b]_D$, by $[-a]_D$, and by $[ca]_D$, respectively.[21] Similarly, the homomorphic property allows the non-interactive computation of $[g(a_1, \ldots, a_k)]_D$ for any given linear function g and commitments $[a_1]_D, \ldots, [a_k]_D$.

It is important to point out some relations and subtle differences between this and other protocols for similar purposes described in the literature. First of all, note that when the commitment scheme is i.t. binding, then the information held by $P_1, \ldots, P_n$ (actually only the honest players are needed) uniquely determines the committed value. If the players $P_1, \ldots, P_n$ can *efficiently*[22] reconstruct the committed value, then such a scheme is identical to a VSS. For small n, reconstruction is always efficient. More generally, the i.t. commitment scheme (of Section 4.2) will for many choices of $\mathcal{A}$ (but not always) be a VSS.

If the scheme is not a VSS, then the committer D can always refuse to open a value, and the players cannot efficiently determine the value without D's help. In Weak Secret Sharing (WSS) introduced in [29, 30], the dealer may even escape the commitment stage without a value being uniquely defined, but then he is always caught in the OPEN protocol. In both these cases, one could define a default value to be taken as the result of OPEN, i.e., playing incorrectly is simply defined as one way for the dealer to commit to the default value. This is similar to the standard definition of a VSS (see [22]), where a default value is defined when the dealer handed out inconsistent shares. However, schemes of the type discussed in this paragraph can never be homomorphic because inconsistencies must be prevented in the first place.

4.1.1 Commitment Transfer Protocol (CTP)

The CTP is a generic protocol that exists for every commitment scheme, and its security is with respect to the same adversary structure. The CTPs for both settings are described in the corresponding sections below.

The CTP allows a player D committed to a value a by a commitment $[a]_D$ to transfer the commitment to another player R who will afterwards be committed to the same value by a commitment $[a]_R$. The players $P_1, \ldots, P_n$ may be involved in this protocol. In the context of secure MPC, both D and R will also be players in $\{P_1, \ldots, P_n\}$. A CTP (for adversary structure $\mathcal{A}$) is $\mathcal{A}$*-secure* if:

(i) *Correctness*: After the protocol, either R is committed to a or D is globally (i.e., by all honest players) detected as being corrupt. An honest D is never deemed corrupt.

(ii) *Privacy*: If both D and R are honest, then any admissible adversary obtains no information about a from his view of the CTP protocol.

Note that if D is honest, then a corrupted R cannot achieve anything more by cheating in CTP than what he could achieve by playing honestly in CTP but cheating afterwards.

4.1.2 Commitment Sharing Protocol (CSP) and VSS

Let $\mathcal{A}$ be an adversary structure, and let $\mathcal{S}$ be an LSSS over a finite field K, with access structure Γ satisfying $\Gamma \cap \mathcal{A} = \emptyset$. Let ψ be the labelling function of the MSP underlying $\mathcal{S}$. Let an $\mathcal{A}$*-secure* homomorphic commitment scheme be given over K with a CTP.

The CSP protocol allows a player D already committed to a value a by $[a]_D$ to share a according $\mathcal{S}$, resulting in shares $a_1, \ldots, a_d$, and to let every player be committed to his $\mathcal{S}$-share of a: for $1 \leq k \leq d$, player $P_{\psi(k)}$ is committed to a_k by $[a_k]_{P_{\psi(k)}}$. A CSP is $\mathcal{A}$*-secure* if:

(i) If D is honest, then (a) every player is committed to his $\mathcal{S}$-shares of a, and (b) any admissible adversary learns no information about a.

(ii) If D is corrupt, then either condition (a) holds or D is globally detected as being corrupt.

Our $\mathcal{A}$-secure CSP works as follows:

1. D chooses random coefficients needed in an $\mathcal{S}$-sharing of a, commits to them, and computes $a_1, \ldots, a_d$.
2. Using the homorphic property the players (non-interactively) compute the following commitments (by D): $[a_1]_D, \ldots, [a_d]_D$.
3. For $1 \leq k \leq d$, D transfers $[a_k]_D$ to player $P_{\psi(k)}$ using CTP, resulting in $[a_k]_{P_{\psi(k)}}$.
4. Cheaters are detected in CSP if they are in CTP or COMMIT.

The CSP directly yields a homomorphic VSS for the same $\mathcal{A}$:

Sharing: To VSS a value a, D commits to a and uses the CSP to commit all players to $\mathcal{S}$-shares of a.

Reconstruction: Every player opens the commitment to his $\mathcal{S}$-share(s) of a and the standard reconstruction procedure for $\mathcal{S}$ is applied.[23]

[19] Depending on the setting, the hiding property is i.t. or cryptographic.

[20] In other words, the hiding properties of all commitments are independent.

[21] In the i.t. setting, two commitments are added by adding the corresponding information held by the players. The known cryptographic homomorphic commitments consist of an element in a suitable (multiplicatively written) finite group, and $[a + b]_D$ is equal to $[a]_D \cdot [b]_D$ (see Appendix 1).

[22] even in the presence of an admissible adversary

[23] Note that no false shares can be contributed. Because of the Q^2 condition on $\mathcal{A}$, the set of correct players is qualified in $\mathcal{S}$.

4.1.3 Commitment Multiplication Protocol (CMP)

This type of protocol takes place in the same setting as described for CSP except that we assume we have an LSSS $(\mathcal{S}, \mathcal{S}_\mu)$ with *strong* multiplication. Let Γ be the access structure of $\mathcal{S}$. We require that $\Gamma \cap \mathcal{A} = \emptyset$. It allows a player D who is committed to values a and b by commitments $[a]_D$ and $[b]_D$ to commit to a value c using $[c]_D$. A CMP is $\mathcal{A}$-secure if:

(i) For any admissible adversary, either $c = ab$ or all correct players reject the outcome of the protocol.[24]

(ii) If D is honest, the protocol always results in $c = ab$, and any admissible adversary learns nothing about c (except that $c = ab$). Moreover, opening $[c]_D$ reveals no information about a and b other than ab.

Our $\mathcal{A}$-secure CMP works as follows:

1. For commitments $[a]_D$ and $[b]_D$, D performs steps 1 and 2 of CSP for LSSS $\mathcal{S}$ and sends the shares privately to the players.
2. D commits to $c := ab$ using commitment $[c]_D$, and performs the steps 1 and 2 of CSP for LSSS $\mathcal{S}_\mu$, setting the random coefficients appropriately (in step 1 of CSP) such that the resulting shares satisfy $c_k = a_k b_k$ for $1 \leq k \leq d$.[25] He sends the shares privately to the players.
3. For $1 \leq k \leq d$, D executes the OPEN protocol for the commitments $[a_k]_D$, $[b_k]_D$ and $[c_k]_D$, such that all information is sent to $P_{\psi(k)}$ instead of being broadcast. Each player checks for his shares whether the opening succeeded and $c_k = a_k b_k$ holds and, if the check fails, accuses D who must then open (in public) the corresponding commitments. If OPEN fails or if for some successfully opened commitments $[a_k]_D$, $[b_k]_D$ and $[c_k]_D$ it holds that $c_k \neq a_k b_k$, then D is deemed corrupt, otherwise P_i is deemed corrupt.

This protocol applies both to the cryptographic and the i.t. setting, but in the cryptographic setting we will for efficiency reasons and because usually $\mathcal{A}$ is not Q^3 use a different CMP exploiting the concrete realization of the commitments.

Regarding the security of CMP we note the following. *Strong* multiplication implies that the honest players are qualified in $\mathcal{S}_\mu$. Hence, if the outcome of the protocol is accepted, then the $\{a_k b_k\}$ and $\{c_k\}$ held by the honest players are $\mathcal{S}_\mu$-shares determining the respective secrets ab and c uniquely. But since these sets of shares are identical, we must have $ab = c$.

The honest dealer case is clear since, by inspection, what an admissible adversary receives contains no information about a and b and $[ab]_D$ is independently generated, and a dealer is only deemed corrupt if he really is.

4.2 Homomorphic Information-Theoretic Commitments

We first present a generic transformation that converts an LSSS into another LSSS with the same access structure, suitable to be used as the basis of the commitment scheme.

[24]For the cryptographic scenario, a super-polynomially small error probability, taken over the coin flips of correct players, is allowed.

[25]Such coefficients exist by the multiplication property, and can be computed by solving a system of linear equations.

Let $\mathcal{S}$ be an LSSS with access structure Γ on n players. Let (K, M, ψ) be the underlying MSP computing $f = f_\Gamma$. Say that M has d rows and e columns. Note that if f rejects a given adversary structure $\mathcal{A}$, then $\mathcal{A} \subset \mathcal{A}_f$ and that we can efficiently recognize the sets in $\mathcal{A}_f$ using the MSP. We require that f is Q^3.

Taking $\mathcal{A}_f$ as the adversary structure, the new LSSS derived from the above can be shown to be robust against any admissible set of cheaters when the dealer is honest, but this property will not be needed here. It can be viewed as a two-dimensional (instead of one-dimensional) redundant sharing, where the redundancy will be used for checking in the commitment scheme.

Share Distribution: Let $s \in K$ be the secret to be distributed. The dealer chooses at random a symmetric e by e matrix R over K, subject to the constraint that the upper-left corner of R contains s. For $i = 1, \ldots, n$, the dealer puts $U_i := M_i R$, and sends the matrix U_i privately to player i.

Reconstruction: Each player P_i broadcasts U_i. Let $\tilde{U}_i$ be the value actually broadcast by P_i. Let $\tilde{\mathbf{s}}_i$ denote the first column of $\tilde{U}_i$. Let G be the n by n (0,1)-matrix whose (i,j)-entry is 1 if and only if $M_j \tilde{U}_i{}^t = (M_i \tilde{U}_j^t)^t$. Let $B_i = \{j : G(i,j) = 0\}$ and $B = \{i : f(B_i) = 0\}$. Then $f(B) = 1$ and $\tilde{\mathbf{s}}_i = \mathbf{s}_i$ for all $i \in B$. Hence, s can be recovered from the shares of the players in B.

The following proposition is proved in the Appendix.

PROPOSITION 6 *The transformed scheme is a secret sharing scheme for* Γ *as well and is robust against cheaters if the dealer is honest.*

We now extend the above LSSS to a commitment scheme by ensuring that even when the dealer is corrupt, the correct players are guaranteed to receive consistent shares of a secret. Note that $M_i U_j^t = (M_j U_i^t)^t$ can be computed by players P_i and P_j who can hence check the consistency on these values.

The protocol COMMIT consists of five steps:

1. Let $a \in K$ be the value to be committed to by D. D runs the above Share Distribution stage with $s = a$, thus he sends privately to each player P_i the matrix U_i.
2. Each player P_i computes for $1 \leq j \leq n$ the d_i by d_j check matrix $C_{ij} := U_i M_j^t$ and, for all $j > i$, sends it privately to P_j.
3. Each player P_i checks for all $j < i$ whether $C_{ij} = C_{ji}$. If this fails, P_i broadcasts a complaint and D must broadcast C_{ij}. If the check fails for a set of indices j corresponding to a set $A \notin \mathcal{A}_f$, then P_i broadcasts an accusation [26] against D who must then broadcast U_i.
4. Each player performs all relevant verifications on the values broadcast by D (due to the complaints and accusations) and those known to him and accuses D if there is an inconsistency. Now, D must make public U_j for any new accusing players P_j. This process continues until no new accusations are made.

[26]After having made an accusation, the player halts and awaits Step 5.

5. If D refuses to perform the protocol correctly, publishes inconsistent information, or if the set of players who accused D is not in $\mathcal{A}_f$, then D is deemed corrupt and all players take a default value for D's value. Otherwise, the accusing players take the published information as their shares.

The players only need to store the actual shares of the underlying LSSS, i.e. player P_i stores only the first columns $\mathbf{a}_i$ of U_i. We thus have $[a]_D = (\mathbf{a}_1, \ldots, \mathbf{a}_n)$.

To OPEN a commitment, D broadcasts the first column of R (including a), allowing all players to compute all of $[a]_D$. Each player P_i accuses D if his stored value $\mathbf{a}_i$ does not match. If the set of accusations corresponds to a set in $\mathcal{A}_f$, then a is taken as the opened value, otherwise D is deemed corrupt. The following theorem is proved in Appendix 3.

THEOREM 8 *The described scheme is an $\mathcal{A}$-secure homomorphic commitment scheme that is information-theoretically binding and hiding.*

The CTP for a commitment $[a]_D$ of this type is as follows, where the receiving player is here called Rec instead of R:

1. D sends the first column of R to Rec, from which Rec computes the value of $[a]_D$ claimed by D. Each player P_i sends $\mathbf{a}_i$ to Rec who compares with the corresponding value claimed by D.

2. If the set of inconsistencies does not correspond to a set in $\mathcal{A}_f$, then Rec accuses D and we go to step 3. Otherwise, for each inconsistency, Rec asks D to broadcast the value of $\mathbf{a}_i$. If this does not match Rec's value for $[a]_D$, Rec accuses D and we go to step 3. Else, P_i changes his value for $\mathbf{a}_i$ to the one broadcast.

3. If at this point, D has been accused, he must open $[a]_D$. If this fails, he is deemed corrupt, and otherwise the opened commitment is taken to be $[a]_{Rec}$.

   If D has not been accused, let $[a]_D$ be the values after the modifications in step 2. Then Rec makes a commitment $[a]_{Rec}$, and he opens $[a]_D - [a]_{Rec}$. If this fails or does not reveal 0, D opens $[a]_D$, and $[a]_{Rec}$ is set to be $[a]_D$.

The last part of Step 3 ensures that even if both D and Rec are corrupt, the protocol will not produce an $[a]_{Rec}$ that is not well formed. On the other hand, step 2 ensures that an honest Rec can always complete it successfully.

Proof of Theorem 3. This theorem follows directly from the above commitent scheme realized from any MSP $\mathcal{M}$ computing a Q^3 function f rejecting $\mathcal{A}$, and from the CSP, based on the LSSS induced by $\mathcal{M}$, described in Section 4.1.2. Note that by reverting to default shares if the dealer is deemed corrupt, we can guarantee that the VSS always terminates successfully.

5 General Secure Multi-Party Computation

Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be the set of players, let C be the arithmetic circuit over some finite field K consisting of addition and multiplication gates to be securely computed, where each input is associated with a player.[27] Without loss of generality we assume that all gates have two inputs.

Like many other MPC protocols (e.g. [7, 12]), our protocols consist of the following three stages: input distribution, computation stage, and output reconstruction. The computation stage consists of a gate-by-gate evaluation of C, keeping as an invariant that each intermediate value is shared among the players according to a certain LSSS which, in order enable the computation of multiplication gates, must have the multiplication property.

Let $\mathcal{A}$ be any adversary structure for $\mathcal{P}$, where $\mathcal{A}$ must be Q^2, except for the i.t. active setting where $\mathcal{A}$ must be Q^3. Let further f be any monotone function that rejects $\mathcal{A}$ and that is computable by be an MSP with multiplication $\mathcal{M} = (K, M, \psi)$, and let $(\mathcal{S}, \mathcal{S}_\mu)$ be the LSSS with multiplication induced by $\mathcal{M}$ via Proposition 3. In the i.t. *active* setting, the strong multiplication property is assumed,[28] which implies that f is Q^3. We refer to Sections 2 and 3 for definitions.

5.1 Passive Adversaries

Most parts of the protocol are described as an iteration over the row index of M, where for $1 \leq k \leq d$, row k is associated with player $P_{\psi(k)}$. The same player may be associated with several rows, and a typical implementation would probably iterate over the players rather than the rows of M.

Input Distribution: For each input variable of C, the associated player (say P_i) distributes shares of the input value among all the players, using the LSSS $\mathcal{S}$.

Computation Stage: Let G be a gate to be computed, let s and s' be the inputs to G, and for $1 \leq k \leq d$, let s_k and s'_k be player $P_{\psi(k)}$'s shares[29] in s and s', respectively. After the computation stage for gate G, $P_{\psi(k)}$ holds a share v_k of the output of G.

If G is an addition gate, then all players locally add their shares of s and s': for $1 \leq k \leq d$, player $P_{\psi(k)}$ computes $v_k := s_k + s'_k$. Multiplication with a constant is done analogously. If G is a multiplication gate, then the following subprotocol is used.

1. *Re-Sharing*. For $1 \leq k \leq d$, player $P_{\psi(k)}$ computes $t_k := s_k s'_k$ and shares t_k (according to $\mathcal{S}$) among the players. Let u_{kj} (for $1 \leq j \leq d$) be $P_{\psi(j)}$'s resulting share in t_k.
2. *Recombination*. For $1 \leq k \leq d$, player $P_{\psi(k)}$ computes $v_k \leftarrow \sum_{l=1}^{d} r_l u_{lk}$, where $\mathbf{r} = (r_1, \ldots, r_d)$ is the recombination vector[30] of $\mathcal{M}$.

Output Reconstruction Stage: After the output gate(s) are computed, each player sends his share(s) of the result(s) to all other players. The players locally reconstruct the result(s) using the normal reconstruction algorithm for $\mathcal{S}$.[31]

[27] This could also be a player not in $\mathcal{P}$. Each player P_k can be associated with zero, one, or several inputs.

[28] This is needed in the CMP.

[29] $P_{\psi(k)}$ may have several shares in s and s' if $\psi(k) = \psi(l)$ for $k \neq l$.

[30] If there are several such $\mathbf{r}$, then any fixed one can be used.

[31] If the specification of the function is such that every result value should be obtained only by a certain subset of the players, then the corresponding shares are sent only to those players.

Proof of Theorem 4, passive case. We must show that the protocol computes the correct result and that the adversary learns no information about inputs of honest players, except what is implied by the result. Let $M(a, \boldsymbol{\rho})$ be a sharing of value a computed in the protocol, where a is an input value or an output value from some gate. It is enough to prove that a is correct (given the input values), and if a is or depends on an input from an honest player, then $M(a, \boldsymbol{\rho})$ is a random set of shares with the only restriction that it determines the value a.

This can be shown by induction over the gates, where the basis follows from the fact that input values are correctly shared. The induction step is trivial for addition gates and multiplication by a constant. For a multiplication gate, we can prove correctness as follows, assuming that we have as inputs correct sharings of values s, s':

Let $\boldsymbol{\rho}_l \in K^{e-1}$ be the random coefficients used by P_l for sharing t_l, and let $\boldsymbol{m}_k$ be the k-th row of M. We have $v_k = \sum_{l=1}^{d} r_l \langle \boldsymbol{m}_k, (t_l, \boldsymbol{\rho}_l)\rangle$ and hence $(v_1, \ldots, v_d) = \sum_{l=1}^{d} r_l M(t_l, \boldsymbol{\rho}_l) = M(ss', \sum_{l=1}^{d} r_l \boldsymbol{\rho}_l)$, which is a correct sharing of ss'.

For the privacy part, note that for at least one honest $P_{\psi(l)}$, $r_l \neq 0$, since otherwise the adversary could compute ss' from his shares of s and s'. But this implies that $\sum_{l=1}^{d} r_l \boldsymbol{\rho}_l$ is uniform and chosen independently of anything else. This concludes the proof[32].

5.2 Active Adversaries

The approach taken in [7] for converting the protocol secure against passive adversaries into one secure against active adversaries consists of two key elements: replacing normal secret sharing by VSS and using error-correcting codes for correcting false values contributed by corrupt players. In order for error correction to be efficient, the cardinality of the multiplicative group of the field K must be chosen to be a multiple of n, and the field elements assigned to players must be n-th roots of unity. This implies that the decoding problem corresponds to the well-known decoding problem for Reed-Solomon codes.

In this paper we take an entirely different approach, not relying on error correction and hence without need for a restriction on K. Our VSS protocol described in Section 4.1.2 has the additional property that the players are *committed* to their shares. Thus the invariant holding for the output of each gate is stronger than in [7]: each value is shared according to $\mathcal{S}$, where each player is committed to all his share(s). As a consequence, instead of performing error correction during reconstruction, false shares can immediately be recognized.

This implies that the protocol can be modeled according to that of Section 5.1, but every player will have to prove to the others that he is performing all local operations correctly. Then the only possible type of active cheating is to stop executing the protocol by performing an illegal operation. The protocol is generic in that it applies to both the i.t. and the cryptographic setting because these two cases differ only in the corresponding type of commitments used.

For the cryptographic (i.t.) setting, let $\mathcal{A}$ be an arbitrary Q^2 (Q^3) adversary structure to be tolerated, and let a homomorphic cryptographic (i.t.) commitment scheme be given. Let $\mathcal{M}$ and $(\mathcal{S}, \mathcal{S}_\mu)$ be as described above. In particular this means that $\mathcal{M}$ can be used as the basis for the commitment scheme in the i.t. setting, and that $\mathcal{S}$ can be used as the basis of CSP and of the CMP in the i.t. setting (which additionally uses $\mathcal{S}_\mu$). For simplicity, we first describe the protocol assuming that no players are detected as cheaters, and then describe how to deal with cheating.

Input Distribution Stage: For each input variable of C, the associated player (say P_i) commits to the input value and applies to it the CSP of Section 4.1.2 with respect to the LSSS $\mathcal{S}$ (i.e. uses VSS from Section 4.1.2). In the cryptographic setting, P_i also proves that he knows the committed value.[33]

Computation Stage: Let G be a gate to be computed, let s and s' be the inputs to G, and for $1 \leq k \leq d$, let s_k and s'_k be player $P_{\psi(k)}$'s shares in s and s', respectively, and let $[s_k]_{P_{\psi(k)}}$ and $[s'_k]_{P_{\psi(k)}}$ be the corresponding commitments. After the computation stage for gate G, $P_{\psi(k)}$ holds a share v_k of the output of G and is committed to it by $[v_k]_{P_{\psi(k)}}$.

If G is an addition gate, then all players locally add their shares of s and s'. More precisely, for $1 \leq k \leq d$, player $P_{\psi(k)}$ computes $v_k = s_k + s'_k$ and, using the homomorphic property of the commitment scheme, is committed to v_k by commitment $[s_k + s'_k]_{P_{\psi(k)}}$. This step involves no interaction between the players, regardless of whether cryptographic or i.t. commitments are used. Multiplication with a constant c is done analogously.

If G is a multiplication gate, then the following subprotocol is used.

1. For $1 \leq k \leq d$, player $P_{\psi(k)}$ computes $t_k := s_k s'_k$ and uses CMP with inputs $[s_k]_{P_{\psi(k)}}$ and $[s'_k]_{P_{\psi(k)}}$ to be committed to t_k by commitment $[t_k]_{P_{\psi(k)}}$.
2. For $1 \leq k \leq d$, player $P_{\psi(k)}$ performs CSP with respect to $\mathcal{S}$ for commitment $[t_k]_{P_{\psi(k)}}$. Let u_{kj} (for $1 \leq j \leq d$) be $P_{\psi(j)}$'s resulting share in t_k and let $[u_{kj}]_{P_{\psi(j)}}$ be the corresponding commitment.
3. For $1 \leq k \leq d$, player $P_{\psi(k)}$ computes $v_k := \sum_{l=1}^{d} r_l u_{lk}$, where $\mathbf{r} = (r_1, \ldots, r_d)$ is the recombination vector of $\mathcal{S}$. Then, using the homomorphic property of commitments, $P_{\psi(k)}$ is committed to v_k by the commitment $[v_k]_{P_{\psi(k)}}$. This step involves no communication.

Output Reconstruction Stage: For $1 \leq k \leq d$, $P_{\psi(k)}$ knows the $\mathcal{S}$-share w_k of the result and is committed to it by $[w_k]_{P_{\psi(k)}}$. This commitment is opened. Finally, all players locally reconstruct the result w from $w_1, \ldots, w_d$ using the normal reconstruction algorithm for $\mathcal{S}$ (Some players may refuse to open their shares, but this can be ignored since a qualified set will always be available).

[32] It is useful to note that the privacy of the protocol only depends on the honest behaviour of the non-corrupted players and the corrupted players performing consistent but not necessarily random sharing. Also a crash-failure of a non-qualifed set of players can be tolerated.

[33] This is to ensure input independence and affects the over-all complexity only by a constant factor. An alternative would be the use of a non-malleable cryptosystem [17].

Dealing with cheaters. When one or several players are detected as cheaters during one of the subprotocols (for instance if they refuse to continue the protocol execution), they are excluded from the further computation. The only problem with such an event occurs in Step 3 of the Computation Stage, since not all u_{lk} will be available. If $\mathcal{S}$ has strong multiplication (which is always the case in the i.t. setting), we can use instead a recombination vector involving only entries from those players that complete Steps 1 and 2 successfully, and so cheaters can simply be ignored. Otherwise, the remaining players need to reset the protocol to right after the input distribution stage, open the cheater(s) VSSed input(s), and redo the computation of C without the cheaters. In the worst case, an adversary can hence prolong the computation by at most a factor equal to the maximal size of a set in $\mathcal{A}$, which is upper bounded by n.

Acknowledgements

We would like thank Mathias Fitzi, Anna Gál, Rosario Gennaro, Martin Hirt, Peter Bro Miltersen, Tal Rabin, and Markus Stadler for interesting discussions and comments. Michael Ben-Or inspired us by showing us an unpublished threshold-type multiplication subprotocol for the information theoretic scenario.

References

[1] S. B. Akers: *Logical Design with Three-Input Majority Gates*, Computer Design, March/April/May/June 1963.

[2] L. Babai, A. Gál, J. Kollár, L. Rónyai, T. Szabó, A. Wigderson: *Extremal Bipartite Graphs and Superpolynomial Lowerbounds for Monotone Span Programs*, Proc. ACM STOC '96, pp. 603–611.

[3] D. Beaver: *Foundations of Secure Interactive Computing*, Proc. of Crypto 91.

[4] D. Beaver and A. Wool: *Quorum-based secure multi-party computation*, to appear in Proc. of Eurocrypt 98.

[5] A. Beimel: *Secure Schemes for Secret Sharing and Key Distribution*, Ph.D.-thesis, Technion, Haifa, June 1996.

[6] J. Benaloh, J. Leichter: *Generalized Secret Sharing and Monotone Functions*, Proc. of Crypto '88, Springer Verlag LNCS series, pp. 25–35.

[7] M. Ben-Or, S. Goldwasser, A. Wigderson: *Completeness theorems for Non-Cryptographic Fault-Tolerant Distributed Computation*, Proc. ACM STOC '88, pp. 1–10.

[8] M. Bertilsson, I. Ingemarsson: *A Construction of Practical Secret Sharing Schemes using Linear Block Codes*, Proc. AUSCRYPT '92, LNCS 718 (1993), 67-79.

[9] E. F. Brickell: *Some Ideal Secret Sharing Schemes*, J. Combin. Maths. & Combin. Comp. 9 (1989), pp. 105–113.

[10] R. Canetti: *Studies in Secure Multiparty Computation and Applications*, Ph. D. thesis, Weizmann Institute of Science, 1995.

[11] R. Canetti, U. Feige, O. Goldreich, M. Naor: *Adaptively Secure Multi-Party Computation*, Proc. ACM STOC '96, pp. 639–648.

[12] D. Chaum, C. Crépeau, I. Damgård: *Multi-Party Unconditionally Secure Protocols*, Proc. of ACM STOC '88, pp. 11–19.

[13] B. Chor, S. Goldwasser, S. Micali, B. Awerbuch: *Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults*, Proc. IEEE FOCS '85, pp. 383–395.

[14] R. Cramer, I. Damgård: *Zero Knowledge for Finite Field Arithmetic or: Can Zero Knowledge be for Free?*, to appear in Proc. of CRYPTO'98.

[15] C. Crepeau, J.vd.Graaf and A. Tapp: *Committed Oblivious Transfer and Private Multiparty Computation*, proc. of Crypto 95, Springer Verlag LNCS series.

[16] Stefan Dziembowski: private communication, 1998.

[17] D. Dolev, C. Dwork, and M. Naor, *Non-malleable cryptography*, Proc. ACM STOC '91, pp. 542–552.

[18] P. Feldman, S. Micali: *An Optimal Probabilistic Protocol for Synchronous Byzantine Agreement*, SIAM J. Comp. Vol. 26, No. 4, pp. 873–933, August 1997.

[19] M. Fitzi, U. Maurer: *Efficient Byzantine agreement secure against general adversaries*, preprint.

[20] A. Gál: *Combinatorial Methods in Boolean Function Complexity*, Ph.D.-thesis, University of Chicago, 1995.

[21] R. Gennaro: *Theory and Practice of Verifiable Secret Sharing*, Ph.D. thesis, MIT, 1996.

[22] R. Gennaro, M. Rabin, T. Rabin, *Simplified VSS and Fast-Track Multiparty Computations with Applications to Threshold Cryptography*, to appear in Proc of ACM PODC'98.

[23] O. Goldreich, S. Micali and A. Wigderson: *How to Play Any Mental Game or a Completeness Theorem for Protocols with Honest Majority*, Proc. of ACM STOC '87, pp. 218–229.

[24] M. Hirt, U. Maurer: *Complete Characterization of Adversaries Tolerable in General Multiparty Computations*, Proc. ACM PODC'97, pp. 25–34.

[25] M. Karchmer, A. Wigderson: *On Span Programs*, Proc. of Structure in Complexity, 1993.

[26] S. Micali and P. Rogaway:*Secure Computation*, Manuscript, Preliminary version in Proceedings of Crypto 91.

[27] T. P. Pedersen: *Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing*, Proc. CRYPTO '91, Springer Verlag LNCS, vol. 576, pp. 129–140.

[28] P. Pudlák, J. Sgall: *Algebraic Models of Computation and Interpolation for Algebraic Proof Systems* Proc. Feasible Arithmetic and Proof Complexity, Springer Verlag LNCS series.

[29] T. Rabin: *Robust Sharing of Secrets when the Dealer is Honest or Cheating*, J. ACM, 41(6):1089-1109, November 1994.

[30] T. Rabin, M. Ben-Or: *Verifiable Secret Sharing and Multiparty Protocols with Honest majority*, Proc. ACM STOC '89, pp. 73–85.

[31] A. Shamir: *How to Share a Secret*, Communications of the ACM 22 (1979) 612–613.

[32] M. van Dijk: *Secret Key Sharing and Secret Key Generation*, Ph.D. Thesis, Eindhoven University of Technology, 1997.

Appendix 1: Homomorphic Cryptographic Commitments

In [14], Cramer and Damgård introduced the concept of q-one-way group homomorphisms and showed how to base a homomorphic cryptographic commitment scheme on any realization of it. Here, we only show one concrete realization based on Diffie-Hellman type encryption, and refer to [14] for more examples.

Let $K = Z_q$ for a (large) prime q and let G be a finite group of order q in which computing discrete logarithms is infeasible.[34] For example, if p is a prime such that q divides $p - 1$ and g is an element of order q in Z_p^*, then $G = \langle g \rangle$ is a possible choice. Let g be a generator of G.

Every player (say D) can choose at random a secret key $x \in \{0, \dots, q-1\}$ and a random generator y of G. The public key of D will be $h = g^x, y$. A commitment $[a]_D$ by D to a value a is computed by choosing at random $r \in \{0, \dots, q-1\}$ and letting $[a]_D = (g^r, y^a h^r)$. It can be opened by making a and r public. The players can then (individually) check whether this matches the commitment. This scheme is clearly

[34] Depending on the implementation, the group G can be different for each player, and chosen by him, or it can be universal.

homomorphic and unconditionally binding, and it is computationally hiding assuming that this type of Diffie-Hellman probabilistic encryption is polynomially secure.

In [14], a zero-knowledge CMP for any commitment scheme of this type is presented. From this protocol, it is easy to derive a zero-knowledge protocol for proving that one knows how to open a commitment, and for showing that two commitments contain the same value; this latter protocol works, even if the two commitments are with respect to different public keys or even different groups (but with the same order q). All these protocols are efficient in the sense that they achieve error probability $1/q$ and have communication complexity corresponding to only a constant number of commitments.

Note that with the same keys and assumption, one can securely send a message m to D by sending the encryption (g^r, mh^r). D can recover m using x. The sender can prove in zero knowledge that he knows what he is encrypting by proving knowledge of r. Also this is possible efficiently in the same sense as above.

The CTP for a commitment $[a]_D = (g^r, y^a h^r)$ is as follows:

1. D sends a and the random choices for the commitment (in this case r) to R, encrypted under the public key of R. Also, D proves that he knows the encrypted value [35].
2. R tests if this information, after decryption, opens $[a]_D$ correctly. If this test fails, he asks D to open $[a]_D$ (in public). If this opening fails, D is deemed corrupt, and otherwise R is assigned a default commitment to a. If the test succeeds, R commits to a under his own public key and proves to the other players that it contains the same value as $[a]_D$. If this fails, D opens $[a]_D$ and R is assigned a default commitment to a.

As a final remark, we note that [14] also give examples of unconditionally hiding and computationally binding commitments based on q-one way homomorphisms. Even though our definition of commitments requires unconditional binding, these commitments can still be used in our protocols, but this requires a more complex procedure for setting up keys, and is omitted here for brevity.

Appendix 2: Completion of Proofs for Theorems on Span Programs

Proof of Proposition 2.

Recall that $\epsilon = 10\ldots0$. "If"-part: Say Im $M^t_{\otimes}\mathbf{r}$ equals the vector $(10\ldots0) \otimes (10\ldots0)$. Set $M = (m_{ij})$. Then $\sum_i r_i m_{i1}^2 = 1$ and for all $(j,k) \neq (1,1)$, $\sum_i r_i m_{ij} m_{ik} = 0$. The i-th coordinate of $M\mathbf{b} * M\mathbf{b}'$ is equal to $\sum_{j,k} m_{ij}m_{ik}b_j b'_k$, where the b_i and the b'_i are the coordinates of $\mathbf{b}$ and $\mathbf{b}'$, respectively. Hence $< \mathbf{r}, M\mathbf{b} * M\mathbf{b}' >= \sum_i r_i \sum_{j,k} m_{ij}m_{ik}b_j b'_k = \sum_{j,k} b_j b'_k \sum_i r_i m_{ij} m_{ik} = b_1 b'_1$. "Only if"-part: we use the uniqueness of algebraic normal form. If (K, M, ψ) is an MSP with multiplication (write $\mathbf{r}$ for a recombination vector), define the polynomial χ as $\chi(b_1, \ldots, b_e, b'_1, \ldots, b'_e) =< \mathbf{r}, M\mathbf{b} * M\mathbf{b}' >$. This is a polynomial in $2e$ variables (each of which occurs with degree at most 1). By assumption, χ is equal to $b_1 b'_1$ as a function, and thus $\chi = b_1 b'_1$ as polynomials. Hence, $\sum_i r_i m_{ij} m_{ik} = 0$ for $(i,j) \neq (1,1)$, $\sum_i r_i m_{i1}^2 = 1$, and $M^t_{\otimes}\mathbf{r} = (10\ldots0) \otimes (10\ldots0)$. The claim about strong multiplication is immediate from the above.

Corollary to Proposition 2. Let C be a binary code of length d such that

- C contains its orthogonal complement $C^{\perp}$.
- There is $\mathbf{c} \in C$ with $w_H(\mathbf{c}) = 1 \bmod 2$.

Then C defines a matrix M with d rows with $\epsilon \otimes \epsilon \in M^t_{\otimes}$. If we label the i-th row of M with the literal x_i, then the monotone function $f(x_1, \ldots, x_d)$ computed by the monotone span program with multiplication $(GF(2), M, \psi)$, where $\epsilon = (10\ldots0)$, is as follows: $f(\mathbf{x}) = 1$ if and only if there is $\mathbf{y}$ such that $\mathbf{y} \leq \mathbf{x}$ and $\mathbf{y} \in C \cap H_c$, where H_c is the hyperplane $\{\mathbf{y} :< \mathbf{y}, \mathbf{c} >= 1\}$.

Conversely, any monotone binary span program with multiplication of size d, defines such a code C of length at most d.

Proof of Proposition 4.

Write $h = f(g_1, \ldots, g_n)$. From the MSP's for f and the g_i's we first construct an LSSS whose access structure is Γ_h and use that the fact that the underlying distribution matrix provides an MSP computing h. Finally, we show that it has (strong) multiplication if the constituent MSP's do.

Say that h operates on N binary variables. For $i = 1\ldots n$, consider the subset of those N variables that g_i operates on, and collect their indices in the set U_i. The LSSS is on N players.

Share Distribution:

Let $s \in K$ be the secret to be distributed. The dealer chooses the vector $\boldsymbol{\rho}$ at random (its length will be clear). For $i = 1\ldots n$, the dealer does the following.

Compute $\mathbf{u}_i \leftarrow F_i(s, \boldsymbol{\rho})$. This vector has length $m_i \equiv$ rows (F_i). Write $\mathbf{u}_i = (u_{i1}, \ldots, u_{im_i})$.

For $j = 1\ldots m_i$ do the following. Choose $\boldsymbol{\rho}_{ij}$ at random (its length will be clear) and for all $k \in U_i$ compute $\mathbf{s}_{ijk} \leftarrow G_k^{(i)}(u_{ij}, \boldsymbol{\rho}_{ij})$ and send $\mathbf{s}_{ijk}$ privately to player k.

For later convenience, write $\mathbf{s}_{ij}$ for $G^{(i)}(u_{ij}, \boldsymbol{\rho}_{ij})$. This is a complete set of shares with respect to $G^{(i)}$ in the j-th coordinate of $\mathbf{u}_i$.

Reconstruction:

For a set $B \subset \{1, \ldots, N\}$ define $W_B = \{1 \leq i \leq n \mid g_i(B) = 1$. Note that $h(B) = f(W_B)$. For non-empty B, the players in B can jointly reconstruct all $\mathbf{u}_i$ with $i \in W_B$. Clearly, if $h(B) = f(W_B) = 1$, then those $\mathbf{u}_i$'s are sufficient to reconstruct s, and otherwise not.

The scheme above is clearly an LSSS whose access structure is Γ_h. By inspection, we see that a vector describing a complete set of shares has length $\sum_{i=1}^n$ rows $(F_i) \cdot$ size $(\mathcal{G}^{(i)})$.

[35] This is necessary for technical reasons when proving security of the overall multi-party computation protocol.

Thus the MSP (K, M, χ) computing h (definition of χ is obvious) that can be derived from the dealer's algorithm has exactly that size.

We proceed by proving (strong) multiplication. Let two complete sets of shares $\mathbf{s} = (\mathbf{s}_1, \ldots, \mathbf{s}_n)$ and $\mathbf{s}' = (\mathbf{s}'_1, \ldots, \mathbf{s}'_n)$, with for $i = 1 \ldots n$, $\mathbf{s}_i = (\mathbf{s}_{i1}, \ldots, \mathbf{s}_{im_i})$ and $\mathbf{s}'_i = (\mathbf{s}'_{i1}, \ldots, \mathbf{s}'_{im_i})$ be given, and let s and s' denote the secrets that were shared out.

Assume that the MSPs for f and the g_i's have multiplication. Consider $\mathbf{s}_{ij} * \mathbf{s}'_{ij}$. Since $\mathcal{G}^{(i)}$ has multiplication, there exists a recombination vector $\mathbf{r}^{(i)}$ for it such that $(< \mathbf{r}^{(i)}, \mathbf{s}_{i1} * \mathbf{s}'_{i1} >, \ldots, < \mathbf{r}^{(i)}, \mathbf{s}_{im_i} * \mathbf{s}'_{im_i} >)$ is equal to $\mathbf{u}_i * \mathbf{u}'_i$. Write $\mathbf{u} = (\mathbf{u}_1 * \mathbf{u}'_1, \ldots, \mathbf{u}_n * \mathbf{u}'_n)$. Since $\mathcal{F}$ has multiplication, $< \mathbf{r}, \mathbf{u} > = s \cdot s'$. Thus (K, M, ψ) has multiplication.

We now prove the strong multiplication case. Assume that the MSPs for f and the g_i's have strong multiplication. By the first observations in Section 3.5 and what we proved above h, f and the g_i's are Q^2. Let $A \subset \{1, \ldots, N\}$ satisfy $h(A) = 0$. Write B for its complement. Hence, $h(B) = f(g_1(B), \ldots, g_n(B)) = 1$. It is sufficient to find a set $V \subset \{1, \ldots, n\}$ with the following properties:

- For all $i \in V$, $g_i(B) = 1$ and $g_i(A) = 0$.
- $f(V) = 1$ and $f(V^c) = 0$.

Indeed, we first discard all $\mathbf{s}_i$ and $\mathbf{s}'_i$ with $i \notin V$. Next, for all $i \in V$ we discard from $\mathbf{s}_i$ and $\mathbf{s}'_i$ all shares held by players in A, call what is left $\mathbf{z}_i$ and $\mathbf{z}'_i$, and write $\mathbf{z}_i = (\mathbf{z}_{i1}, \ldots, \mathbf{z}_{im_i})$ and $\mathbf{z}'_i = (\mathbf{z}'_{i1}, \ldots, \mathbf{z}'_{im_i})$.

But by the choice of V and the fact that $(K, G^{(i)}, \psi_i)$ has strong multiplication, there exists a recombination vector $\mathbf{r}^{(i)}$ for it that has coordinates zero at those entries labelled with elements from $A \cap U_i$. For convenience, cut off those coordinates from the recombination vector.

Hence, $(< \mathbf{r}^{(i)}, \mathbf{z}_{i1} * \mathbf{z}'_{i1} >, \ldots, < \mathbf{r}^{(i)}, \mathbf{z}_{im_i} * \mathbf{z}'_{im_i} >)$ is equal to $\mathbf{u}_i * \mathbf{u}'_i$.

Again by the choice of V, there exists a recombination vector $\mathbf{r}$ for (K, F, ψ) that has coordinates zero at those entries labelled with elements from V^c. Hence, $\mathbf{r}$ determines a fixed linear combination over the $\mathbf{u}_i * \mathbf{u}'_i$ with $i \in V$ that yields $s \cdot s'$.

Now define $V = \{1 \le i \le n \mid g_i(A) = 0\}$, then $f(V^c) = h(A) = 0$ and $f(V) = 1$ (f is Q^2). Also, $g_i(A) = 0$ implies $g_i(B) = 1$ (g_i is Q^2). Thus (K, M, χ) has strong multiplication.

Proof of Proposition 5.

In the Q^2-case, it is sufficient to prove that T_3^2- and T_2^1-gates suffice.

Let $f = f(x_1, \ldots, x_n)$ be a monotone function, and let Min (f) denote its minimal satisfying assignments and let Max (f) denote its maximal non-satisfying assignment (minimality/maximality w.r.t. usual Boolean ordering).

Let $V(f)$ denote an arbitrary collection of f's satisfying assignments that contains Min (f) (to make sure that $V(f)$ uniquely determines f). We first describe an induction step that works for any monotone f.

Suppose that $|V(f)| > 2$. Define three proper subsets $V(f_i)$ of $V(f)$ such that each element of $V(f)$ is a member of two of the subsets, and define, for $i = 1, 2, 3$, $f_i = f_i(x_1, \ldots, x_n)$ as the monotone function that evaluates to 1 if and only if the assignment is equal to or greater than some element of $V(f_i)$. By construction we have $f = T_3^2(f_1, f_2, f_3)$.

If some $V(f_i)$ has more than two elements, we can apply the same construction on f_i. Hence by induction we can construct a tree whose internal nodes are T_3^2-gates and whose leaves are monotone functions α with $1 \le |V(\alpha)| \le 2$. Note that the union over all $V(\alpha)$'s is equal to $V(f)$. This tree can be seen as a formula computing f.

For convenience, say that there are m leaves α_i and write $f = h(\alpha_1, \ldots, \alpha_m)$ where h represents the part of the tree consisting of the T_3^2-gates. It is easy to see that h is a self-dual monotone function.

We have the following important property by construction: Let $\mathbf{z} \in V(f)$. Call a leaf α $\mathbf{z}$-good if $\mathbf{z} \in V(\alpha)$. Let $\mathbf{w}$ denote the m-bit string defined by $w_j = 1$ if and only if α_j is $\mathbf{z}$-good. Then $f(\mathbf{z}) = h(\mathbf{w}) = 1$. In other words, the values $\alpha(\mathbf{z})$ where α is not $\mathbf{z}$-good are immaterial.

Now assume in addition that f is Q^2. First note that the functions inductively constructed above need not be Q^2 themselves, so there is no direct way to apply the claim we are proving as an induction hypothesis to them.

By making a specific choice for $V(f)$, we can replace the α_i's by monotone functions β_i of the form $\beta_i = T_2^1(x_k, x_l)$ or of the form $\beta_i = x_k$, without changing the function computed. This then settles the proof (if $|V(f)| < 2$ then f is clearly of the form $f = x_k$ or of the form $f = T_2^1(x_k, x_l)$).

First we define $\overline{\text{Max}}(f) = \{\mathbf{y} \oplus \mathbf{1} \; : \; \mathbf{y} \in \text{Max}\,(f)\}$. Note that all members of this set are satisfying assignments of f, since f is Q^2. An additional implication is that if $\mathbf{z} \in \overline{\text{Max}}(f)$ and $\mathbf{z}'$ is an arbitrary satisfying assignment of f, then there is an index k with $z_k = z'_k = 1$ (call this a shared index). Now set $V(f)$ to the union of $\overline{\text{Max}}(f)$ and Min (f).

We divide the α_i's into two types. *Type I* is the collection of α_i's with $V(\alpha_i) \cap \overline{\text{Max}}(f) \neq \emptyset$, and for *Type II* we require the latter intersection to be empty.

Now we substitute the α_i's for β_i's. If α_i is of Type I, say $V(\alpha_i) = \{\mathbf{z}\}$, define $\beta_i = x_k$ for an arbitrary k with $z_k = 1$. If $|V(\alpha_i)| = 2$, we also define $\beta_i = x_k$ but now we require that k is a shared index. Finally, if α_i is of Type II and $|V(\alpha_i)| = 1$ we do the same as above. Else, if $V(\alpha_i) = \{\mathbf{z}, \mathbf{z}'\}$ we select k and l such that $z_k = z'_l = 1$ and put $\beta_i = T_2^1(x_k, x_l)$. Observe that for $i = 1 \ldots m$, $\alpha_i \le \beta_i$, i.e. if β_i evaluates to 0 then so does α_i.

Define $g = h(\beta_1, \ldots, \beta_m)$. We claim that $f \equiv g$. By the choice of the β_i's, $f \le g$.

We show that $f(\mathbf{z}) = 0$ implies $g(\mathbf{z}) = 0$, hence $g \le f$. We may assume that $\mathbf{z} \in$ Max (f). Then $\mathbf{z} \oplus \mathbf{1} \in V(f)$ and $f(\mathbf{z} \oplus \mathbf{1}) = 1$. Say wlog that $\alpha_1, \ldots, \alpha_r$ are the $\mathbf{z} \oplus \mathbf{1}$-good leaves. Note that these are of Type I. Then

$$f(\mathbf{z} \oplus \mathbf{1}) = h(\alpha_1(\mathbf{z} \oplus \mathbf{1}), \ldots, \alpha_r(\mathbf{z} \oplus \mathbf{1}), 0, \ldots, 0) = 1.$$

Using the monotonicity of h, the fact that $\beta_i \ge \alpha_i$ for $1 \le i \le m$, and the facts that h and the β_i's with $1 \le i \le r$ are self-dual, we have

$$g^*(\mathbf{z} \oplus \mathbf{1}) = h^*(\beta_1^*(\mathbf{z} \oplus \mathbf{1}), \ldots, \beta_m^*(\mathbf{z} \oplus \mathbf{1})) = 1.$$

Hence $g(\mathbf{z}) = 0$. Observe that if f is self-dual, then there are no Type II leaves. In that case f can be computed by a formula consisting of T^2_3-gates only[36].

The Q^3-case is handled similarly except that we use T^1_2-, T^1_3-, and T^2_4-gates (in the recursion, use T^3_4-gates and replace them by T^2_4-gates afterwards).

Proof of Lemma 1.

The first part of the claim (not dealing with multiplication) is well-known. If $t = 1$, we have a trivial MSP. Else, let $M_{t,n}$ denote a Vandermonde matrix (over a field K with $|K| > n$) with n rows and t columns, the i-th row being labelled i, and of the form $1, \alpha_i, \ldots, \alpha_i^{t-1}$ with $\alpha_i \neq \alpha_j$ if $i \neq j$. It is easy to see that $M_{t,n}$ can be viewed as an MSP computing $f_{t,n}$, with root $\boldsymbol{\epsilon} = (10\cdots0)$. In this case the associated secret sharing scheme (as defined in Section 3.2) is identical to Shamir's [31].

To see that $(K, M_{t,n}, \psi)$ has multiplication, observe that the product $h = f \cdot g$ of any two polynomials $f, g \in K[X]$, both of degree at most $t-1$, can be interpolated given n points if $2t \leq n+1$. In particular, if we are given distinct values $P_1, \ldots, P_n \in K \setminus \mathbf{0}$, and the evaluations $h(P_1), \ldots, h(P_n)$, we can reconstruct the coefficients of h by linear operations (coefficients only depending on the P_i's) on the $h(P_i)$'s. This holds in particular for the lowest order coefficient of h, which is the product of the secrets distributed by f and g in an execution of Shamir's scheme with threshold (t, n) and the $f(P_i)$'s and $g(P_i)$'s are the shares in the respective secrets. Thus a fixed linear combination over the $f(P_i) \cdot g(P_i)$'s yields the product of the secrets. For strong multiplication, note that if $3t \leq n+2$, then actually $n-t+1$ of the $f(P_i) \cdot g(P_i)$ suffice.

Proof sketches for Theorem 5 and Proposition 1.

The key to the proof will be the following lemma:

LEMMA 2 *Let $h : \{0,1\}^n \to \{0,1\}$ be any monotone function, and suppose that an MSP of size m computes h. Then there exist monotone functions $f : \{0,1\}^{n+1} \to \{0,1\}$ and $g : \{0,1\}^{n+2} \to \{0,1\}$ such that: f is maximal Q^2, g is maximal Q^3, both f and g can be computed by MSPs of size $O(m)$; and finally any monotone Boolean circuit computing f (or g) yields immediately a monotone Boolean circuit computing h on any input of Hamming weight less than $n/2$.*

This can be shown by defining the function $\tilde{h}$ on n variables as $\tilde{h}(\mathbf{x}) = h(\mathbf{x}) \vee T_n^{n/2}(\mathbf{x})$. Then we can construct the claimed functions f, g by introducing two new Boolean variables y, z and defining $f(\mathbf{x}, y) = T^2_3(y, \tilde{h}(\mathbf{x}), \tilde{h}^*(\mathbf{x}))$ and $g(\mathbf{x}, y, z) = f(\mathbf{x}, y) \vee (z \wedge (\vee_{i=1}^n x_i))$. The claims on the properties of f, g follow from Proposition 4, the well known fact that threshold functions have linear MSP complexity and that a function and its dual have the same MSP complexity (see e.g. [20]).

In [2], the monotone function ODDFACTOR is studied. This function is defined on bipartite graphs on $t = \sqrt{n}$ pairs of vertices, where an input graph can be specified by n Boolean variables, one for each of the potential edges. An input graph is accepted if and only it contains an odd factor, i.e. a subgraph in which every vertex has odd degree (in particular degree > 0). The following result is proved in [2]:

THEOREM 9 *The ODDFACTOR function on n variables can be computed by an MSP of size n, but a monotone Boolean circuit computing the function must have size $n^{\Omega(\log n)}$.*

Now suppose we are given a monotone Boolean circuit family that for any n computes ODDFACTOR on input graphs with less than $n/2$ edges. A simple reduction will show that this family can in fact compute ODDFACTOR in general, at polynomial extra cost. This, together with Theorem 9 and Lemma 2, immediately implies Theorem 5, and in fact even a stronger result where the formulas in the statement are replaced by monotone circuits.

For Proposition 1, we construct the function family $\{h_n\}$ claimed in the proposition by extending the ODDFACTOR function from bipartite graphs to general graphs in the natural way: as before, a graph is accepted if and only if it contains an odd factor. The proposition now follows since both the lower bound from Theorem 9 and the construction of an MSP extend easily to h_n. One can directly check that the resulting MSP has the multiplication property whenever the total number of vertices is even.

Appendix 3: Completion of Proofs Related to Homomorphic Information-Theoretic Commitments

Proof of Proposition 6.

First observe that $M_i U_j^t = M_i (M_j R)^t = M_i R M_j^t = U_i M_j^t$. Next, we argue that a non-qualified set gets no information about s before the reconstruction. Let the set A satisfy $f(A) = 0$. We show that $M_A R$, which is the information players P_i with $i \in A$ get from the dealer, contains no information about s. Let X be any symmetric matrix satisfying the equation $M_A X = M_A R$, and say that X has $\tilde{s} \in K$ in its upper-left corner. From our assumption on A, there exists a vector $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_e) \in \text{Ker}M_A$ with $\mu_1 = 1$ (see Section 3.2). Note that $\boldsymbol{\mu} \otimes \boldsymbol{\mu}$ is a symmetric matrix whose i-th column is $\mu_i \boldsymbol{\mu}$ and that it has 1 in its upper left corner. Then $X + (s - \tilde{s})\boldsymbol{\mu} \otimes \boldsymbol{\mu}$ satisfies the equation as well, and has s in its upper left corner and is symmetric. Hence, for each possible $\tilde{s}$, the number of different solutions X with $\tilde{s}$ in the upper left corner is the same.

Now we prove the claims from the reconstruction. First note that if $\mathbf{c}$ and $\mathbf{c}'$ (both from K^e) have different first coordinates c and c', then $M_i \mathbf{c} = M_i \mathbf{c}'$ for at most a non-qualified set of players i (call this set D). Indeed, we have $M_D(\mathbf{c} - \mathbf{c}') = \mathbf{0}$. But $c - c' \neq 0$ implies that $\boldsymbol{\epsilon} \notin \text{Im } M_D^t$, and hence $f(D) = 0$. Next, it is important to note that for all i, j, $M_j U_i^t = U_j M_i^t$ can be viewed as a collection of shares for player j in the coordinates of $\mathbf{s}_i$ ("shares of shares"). Thus, if a corrupted player i broadcasts any $\tilde{U}_i$ such that $\tilde{\mathbf{s}}_i \neq \mathbf{s}_i$, the set of players j for which $M_j \tilde{U}_i^t = U_j M_i^t$ (the *original* U_j) is non-qualified. But since the corrupted players A control the $\tilde{U}_j$ with $j \in A$, the i-th row contains 1-entries corresponding to at most the union of *two* non- qualified sets. Then, since f is Q^3, we must have $f(B_i) = 1$. Note that if player i is uncorrupted, we trivially have that $f(B_i) = 0$ and that $\tilde{\mathbf{s}}_i = \mathbf{s}_i$. Hence, bad shares $\tilde{\mathbf{s}}_i$ are detected, and there are always sufficient ones to reconstruct [37]

[36] If f is self-dual, then by choosing $V(f) = \text{Min}(f)$ our proof is similar to a method outlined in [1]

[37] It is easy to show that the scheme from Section 3.2 already has the right

the secret s (at least those corresponding to the players in A^c). It is possible to define the transformation also for an arbitrary secret sharing scheme (not necessarily linear) at the cost of some loss of efficiency.

Proof of Theorem 8.

We first prove for the COMMIT protocol that if the dealer is correct, then any admissible adversary learns nothing about a, and the shares held by correct players uniquely determine a.

The remarks about the modified secret sharing scheme assuming an honest dealer suffice up to and including Step 2. The set of values that corrupted players receive from honest ones in Step 3 could already be computed from the information that they got from the dealer earlier. Finally note that honest players will only receive incorrect values from corrupted players, and so will never accuse the dealer. At most they will complain about values received from corrputed players. This means that the dealer will only have to broadcast information sent to corrupted players, so that they learn nothing new from this phase. It is also clear that the honest players all agree on the values sent by the dealer, in particular their shares consistently determine the secret a the dealer meant to commit to.

Next we prove for the COMMIT protocol that for any dealer and any admissible adversary, the above protocol results in the correct players holding consistent shares $\mathbf{a}_i$ of some secret a.

The claim is clearly true in case the protocol ends with the players using the default set of shares (which cannot happen if the dealer is honest). Otherwise, the set of accusing players is in $\mathcal{A}_f$. Note that the union of that set and the set of corrupted players cannot be the full set by the Q^3-condition.

Hence, the set of non-accusing honest players is not empty. These must have pairwise consistent U_i's. Moreover, the U_i's of the accusing honest players were broadcast so if there would have been any inconsistency among those or with the U_i's of the non-accusing honest players, the latter would have accused the dealer. Thus the honest players hold pairwise consistent U_i's.

Write B for the set of honest players. They're qualified so there is $\boldsymbol{\lambda}_B$ with $M^t\boldsymbol{\lambda}_B = \boldsymbol{\epsilon}$. Write U for the superposition of the U_i with $i \in B$. By consistency we have $M_B U^t = U M_B^t$, and $M_B U^t \boldsymbol{\lambda}_B = U M_B^t \boldsymbol{\lambda}_B = U\boldsymbol{\epsilon}_B$. The latter is just the first columns of the U_i's of the honest players, i.e. the $\mathbf{a}_i$ with $i \in B$, and the former denotes a consistent set of shares for B.

The above two claims and the Q^3 property imply the hiding and binding properties. Opening the commitment in a different way would cause at least one inconsistency in every qualified set and hence the set of accusations during OPEN would correspond to a set not in $\mathcal{A}_f$.

Appendix 4: Completion of Proofs of Theorems on Multi-Party Computation Protocols

Several general definitions of secure multiparty computation have been proposed (see e.g. [3, 26, 10]), technically different, but all designed to capture the same basic intuition. While either definition could have been used for a formal proof of security of our protocols, any such proof would by far exceed the space limitations. Instead, we have included sketches of proofs, generic enough to fit any of the definitions.

Proof Sketch for Theorem 4, active case.

The protocol claimed is simply the main protocol described above using the commitment scheme from Section 4.2.

To prove correctness, first note that the commitments and the CSP, CMP subprotocols force even corrupted players to perform as in the passive case, or stop participating. This is true unconditionally, and with zero error probability. But as we have seen, such early stopping will not prevent the remaining players from computing a result from the values that are VSS'ed in the Input Distribution Stage, and hence correctness of results basically follow from these observations and correctness in the passive case. For privacy, note that the subprotocols release no information to the adversary on any inputs of honest players, and hence the adversary learns no more information than in the passive case. The claim on complexity of the protocol follows by simple inspection.

Proof Sketch for Theorem 7.

The protocol claimed is simply the main protocol described above using the commitment scheme from Appendix 1. Correctness of the result can be argued in much the same way as for Theorem 4, except that the adversary is polynomially bounded, and a super-polynomially small error probability is allowed.

For privacy, the proof can take several different forms, depending on the definition one adopts, but the basic ingredient in any case is a simulator that gets access to an oracle. The oracle accepts as input a set of input values from the corrupted players, and returns a result of the computation consistent with these inputs. The simulator must efficiently produce a view for the adversary that is computationally indistinguishable from a view obtained in a real protocol execution. The simulator has access to the public keys of honest players, but not the secret keys.

The strategy for a simulator in our case is (ignoring numerous technical details) to first use the proofs of knowledge given in the first stage to extract the input values used by corrupted players. The oracle is then called to obtain the result to target. For all other subprotocols, the following strategy is used: when a corrupt player acts as dealer, the simulator simply plays on behalf of the honest players while following the rules. When an honest player acts as the dealer, the simulator may not have sufficient information to play normally, but it can use the fact that the subprotocols are zero-knowledge to make indistinguishable simulated conversations. Finally, to arrange for the correct result to be "computed" in the simulation, the simulator arranges the values it commits to on behalf of honest players such that the desired result is guaranteed to come out. This may involve doing the CMP protocol on committed values that do not satisfy the right multiplicative relation, but this can be shown to still lead to indistinguishable simulated conversations.

The claim on complexity of the protocol follows by simple inspection. Note that the CSP and CMP protocols require that the dealer answers a random challenge. This challenge can be generated from standard multi-party coinflipping based on the

error correction property if f is Q^3. But there, the cost of recovery may be superpolynomial in the size of the span program (as opposed to our modified scheme here).

commitment scheme used, and the same challenge can reused by all players that act as dealers at a given point in the protocol.

Proof Sketch for Theorem 6.

The protocol claimed follows the same basic 3-stage framework as the main protocol above, but with some changes in each stage. First, note that the constructions of [23] imply that (cryptographic) commitment and oblivious transfer between any pair of players can be done under our assumption on existence of one-way trapdoor permutations. Hence, a player can VSS a value a by sharing it according to $\mathcal{S}$, encrypting each share under the public key of the receiver, and proving in zero-knowledge that this was done correctly. Thus, in the Input Distribution Stage, each player VSSes his inputs in this way. For the computation stage, we can for instance use a protocol from [15], which shows how to do MPC based on commitments and oblivious transfer. This protocol makes no assumption on the adversary structure, but on the other hand stops completely if a cheater is detected. In our case, however, such an event just means that the players go back to the VSS's from the first stage, reconstruct the inputs of the cheater, and redo the computation without him. Correctness and privacy is proved in basically the same way as for Theorem 7.