



Basic Research in Computer Science

BRICS RS-97-28 Cramer et al.: Span Programs and General Secure Multi-Party Computation

Span Programs and General Secure Multi-Party Computation

Ronald Cramer
Ivan B. Damgård
Ueli Maurer

BRICS Report Series

ISSN 0909-0878

RS-97-28

November 1997

**Copyright © 1997, BRICS, Department of Computer Science
University of Aarhus. All rights reserved.**

**Reproduction of all or part of this work
is permitted for educational or research use
on condition that this copyright notice is
included in any copy.**

**See back inner page for a list of recent BRICS Report Series publications.
Copies may be obtained by contacting:**

**BRICS
Department of Computer Science
University of Aarhus
Ny Munkegade, building 540
DK-8000 Aarhus C
Denmark
Telephone: +45 8942 3360
Telefax: +45 8942 3255
Internet: BRICS@brics.dk**

**BRICS publications are in general accessible through the World Wide
Web and anonymous FTP through these URLs:**

`http://www.brics.dk`
`ftp://ftp.brics.dk`
This document in subdirectory RS/97/28/

Span Programs and General Secure Multi-Party Computation

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

Abstract

The contributions of this paper are three-fold. First, as an abstraction of previously proposed cryptographic protocols we propose two cryptographic primitives: homomorphic shared commitments and linear secret sharing schemes with an additional multiplication property. We describe new constructions for general secure multi-party computation protocols, both in the cryptographic and the information-theoretic (or secure channels) setting, based on any realizations of these primitives.

Second, span programs, a model of computation introduced by Karchmer and Wigderson, are used as the basis for constructing new linear secret sharing schemes, from which the two above-mentioned primitives as well as a novel verifiable secret sharing scheme can efficiently be realized.

Third, note that linear secret sharing schemes can have arbitrary (as opposed to threshold) access structures. If used in our construction, this yields multi-party protocols secure against general sets of active adversaries, as long as in the cryptographic (information-theoretic) model no two (no three) of these potentially misbehaving player sets cover the full player set. This is a strict generalization of the threshold-type adversaries and results previously considered in the literature. While this result is new for the cryptographic model, the result for the information-theoretic model was previously proved by Hirt and Maurer. However, in addition to providing an independent proof, our protocols are not recursive and have the potential of being more efficient.

1 Introduction and Related Work

The main goal of this paper is to propose new efficient secure multi-party computation protocol constructions based on generic primitives and to show that these primitives can be realized by using linear secret sharing schemes based on span programs. It is our hope that this link between span programs and multi-party

*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

‡Basic Research in Computer Science, Center of the Danish National Research Foundation

§Dept. of Computer Science, ETH Zurich, CH-8092 Zurich, Switzerland. Email: maurer@inf.ethz.ch.

Supported by the Swiss National Science Foundation.

computation can be the basis for the discovery of further relations between these two areas that have recently received a lot of attention.

Secure multi-party computation can be defined as the problem of n players to compute an agreed function of their inputs in a secure way, where security means maintaining correctness of the output while keeping the players' inputs private, even in the presence of adversarial behavior by some of the players.

One can distinguish between passive and active cheaters: passive cheaters follow the protocol but pool their information in order to violate the other players' privacy. Active cheaters can use an arbitrary joint strategy in order to violate the correctness and/or privacy of the computation. Usually, this is modeled by assuming the existence of a passive or active *adversary* who can monitor or control some subset of the players. It is (at least initially) unknown to the correct players which subset is affected.

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, i.e. assuming that the adversary cannot solve some computational problem. Privacy here means that an adversary's entire view of the protocol can efficiently be simulated in a manner indistinguishable from a real execution of the protocol. In the information-theoretic (also called secure channels) model, it is assumed that the players can communicate pairwise over secure channels, and the privacy as well as the correctness can then be guaranteed even when the adversary has unbounded computing power. Privacy here means that an adversary's view, when given the output of the computation, is statistically independent of the other player's inputs and hence gives no Shannon information about them. We consider both the cryptographic and the information-theoretic models, assuming synchronous communication and static adversaries¹.

The classical results for the information-theoretic model due to Ben-Or, Goldwasser and Wigderson [4] and Chaum, Crépeau and Damgård [8] state that every function can securely be computed if and only if less than $n/2$ passive or less than $n/3$ active cheaters are present. These results were generalized by Hirt and Maurer [15] who considered as the potential adversaries general sets of subsets of the player set, not necessarily specified by their cardinality.

Using terminology from secret sharing and from [15], we call a set of subsets of the players a *structure* and we consider security (privacy and correctness) of a protocol with respect to an adversary structure, meaning that the protocol remains secure even when an arbitrary set in the structure happens to be controlled by an adversary (which may be passive or active). 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. Note that the threshold situations considered in [4], [8], [14] and [20] are special

¹In the information-theoretic model, our results also hold for adaptive (dynamic) adversaries, i.e. when an adversary decides adaptively which players to corrupt.

cases, where the adversary structure would contain all sets of size less than $n/2$ or $n/3$.

The result of [15] can then be stated as follows: secure multi-party computation is possible in the information-theoretic scenario, while tolerating a passive (active) adversary, if and only if the adversary structure satisfies $Q2$ ($Q3$). One - perhaps somewhat surprising - consequence of this is that in some situations, a *majority* of passive cheaters can be tolerated, provided we do not have to tolerate *any* dishonest majority. The protocols proposed in [15] rely on applying a protocol for the threshold case (e.g. [4]) recursively, so that subsets of players together run threshold protocols to simulate virtual players in higher level protocols. In general, the emphasis in [15] was on existence of protocols rather than on efficiency.

In this paper, we present protocols that directly and perhaps more naturally implement protection against any $Q2$ passive or any $Q3$ active adversary in the information-theoretic scenario. The $Q2$ and $Q3$ conditions arise directly from a natural condition on the underlying secret sharing schemes, and this also leads to a potentially more efficient solution than that of [15]. In particular, the complexity of our protocols is directly related to the size of a monotone span program [16] that rejects all potentially misbehaving player sets and accepts their complements and enjoys a special multiplication property². One "spin-off" from our results that may be of independent interest is a new construction that builds from a monotone span program a verifiable secret sharing scheme for the information-theoretic scenario tolerating any $Q3$ active adversary (the multiplication property is not needed here). Like in [4] and [15], our protocols have zero probability of successful cheating by an active adversary.

In the cryptographic model with an active adversary, the most general previous result was shown by Goldreich, Micali and Wigderson [14] who proved that any minority of active cheaters can be tolerated, assuming that trapdoor one-way permutations exist. In this paper, we show that any $Q2$ active adversary can be tolerated, assuming that one-way group homomorphisms with specific extra properties exist. Particular assumptions sufficient for this include: the RSA assumption, hardness of discrete log in prime order groups, or polynomial security of Diffie-Hellman encryption. To the best of our knowledge, this is the first result for the cryptographic model that goes beyond the threshold case. It can be generalized to rely only on the existence of trapdoor one-way permutations, but only with loss of efficiency.³

²Span program are the most powerful known method for designing linear secret sharing schemes (combine the results of [3] and [16]).

³Even for the threshold case, our protocol yields substantial efficiency improvements over earlier proposals: if the computation is specified as an arithmetic circuit of size m operations over $GF(q)$, where q is a k -bit prime, then with security parameter value k , our protocol has communication complexity $O(kmf(n)^2)$ bits, when based on the most efficient implementations of the primitives, and where $f(n)$ is the size of the monotone span program we need for the particular adversary structure considered. In particular, if we want to tolerate any minority of active cheaters, we will have $f(n) = n$.

This paper draws on three ideas appearing in [13]: the observation that in [4] the product of two shared values is defined as a linear function of the products of the shares, the idea (independently discovered in the course of our research) of using homomorphic cryptographic commitments to build verifiable secret sharing, and the idea of using a specialized zero-knowledge proof for proving correctness of multiplied commitments. One of the achievements of this paper is to show how general linear secret sharing schemes (not only Shamir’s polynomial threshold scheme as in [13]) can be applied in multi-party computation, both in the cryptographic and information-theoretic setting.

2 The Tools: Shared Commitments and Secret Sharing

In this section we introduce some notation and state, at an abstract level, the required properties for our two main tools used in both the cryptographic and the information-theoretic model. *Shared commitments* (often also simply called commitments in this paper) are a generalization of conventional cryptographic commitments, which are in fact used as the implementation of shared commitments in the cryptographic scenario.

We use the following notation throughout the paper. The set of players is denoted $\{P_1, \dots, P_n\}$. In all models, we consider a particular monotone set \mathcal{A} of subsets of $\{P_1, \dots, P_n\}$, called the *adversary structure*, as potential adversaries, and every set in \mathcal{A} is called an *admissible adversary*. The set \mathcal{A} usually satisfies the *Q2* or the *Q3* condition. When considering one particular adversary A in \mathcal{A} , then we refer to the players not in A as correct players. The function to be computed is assumed to be defined as an arithmetic circuit \mathcal{C} over some finite field K .

2.1 Notation and Required Properties for Shared Commitments

To protect against *active* adversaries in a given adversary structure \mathcal{A} , we will need a commitment scheme for elements in K . A shared commitment scheme consists of two protocols, COMMIT and OPEN. Both of these are initiated by one of the players, called the committer, in order to commit to or reveal a value $a \in K$, respectively. During COMMIT the committer distributes some information to the other players, possibly also involving interaction between the players. By the *commitment*, we mean the total information distributed to the correct players. We need the following basic properties:

1. *Binding*: For any committer and any admissible adversary, some value a is uniquely determined from the commitment, and an execution of OPEN based on this commitment results either in all correct players retrieving a , or in all correct players rejecting the outcome of OPEN.
2. *Hiding*: If the committer is a correct player and commits to value a , any admissible adversary learns nothing about a from his view of the COMMIT

protocol. Moreover, executing OPEN based on this commitment always results in all correct players retrieving a .

In the information-theoretic (cryptographic) scenario, both properties hold unconditionally (relative to some complexity assumption).⁴

We will need the following extra properties, where the Multiplication Protocol is only needed in case of an active adversary:

- *Homomorphic*: From commitments A and B containing $a \in K$ and $b \in K$, respectively, the players can, without communicating, compute a commitment containing $a + b \in K$, or compute one containing $a - b \in K$. Motivated by the particular known realizations of homomorphic cryptographic commitments, we will denote these commitments by $A \cdot B$ and AB^{-1} , respectively. This property also implies that without communication, constants can be multiplied or added into commitments. We will let A^c, cA, cA^{-1} denote commitments to $ca, c + a, c - a$, as computed from A . Each new commitment computed in this way can be opened by the committer, but we require that opening $A \cdot B$ reveals no information about a and b other than $a + b$ (and similar for the other operations).
- *Multiplication Protocol*: This protocol allows a player P_i who has previously committed to values a and b by commitments A and B to commit to a value c using commitment C , such that:
 1. For any P_i and any admissible adversary, either $c = ab$ or all correct players reject the outcome of the protocol (for the cryptographic scenario, a superpolynomially small error probability, taken over the coin flips of correct players, is allowed).
 2. If P_i is a correct player, the protocol always results in $c = ab$, and any admissible adversary learns nothing about c except that $c = ab$. Moreover, opening C reveals no information about a and b other than ab .

2.2 Notation and Required Properties for Secret Sharing

A secret sharing scheme can be defined as a probabilistic polynomial time algorithm which takes as input an element s in K and outputs n shares $\mathbf{s}_1, \dots, \mathbf{s}_n$. The share \mathbf{s}_i will be a d_i -vector of elements in K , and in the course of our protocols, the share \mathbf{s}_i will be given to P_i (but will initially be unknown to all other players). Let $d = d_1 + \dots + d_n$. For $C \subseteq \{P_1, \dots, P_n\}$ we call a vector of length $\sum_{i:P_i \in C} d_i$ a d_C -vector. For any t and any t -vectors \mathbf{u}, \mathbf{v} , the t -vector $\mathbf{u} * \mathbf{v}$ will denote the

⁴In the information-theoretic setting, and without the uniqueness condition on the committed value, this corresponds to Weak Secret Sharing defined in [20, 19].

coordinatewise product of \mathbf{u} and \mathbf{v} . The standard inproduct of \mathbf{u} and \mathbf{v} is denoted by $\langle \mathbf{u}, \mathbf{v} \rangle$.

There is a monotone collection of *qualified* subsets of players, called the *access structure*. Any qualified set can efficiently reconstruct the shared secret while any non-qualified set has no information about the secret (perfect secrecy). A set of values that can be interpreted as the shares of a secret s of a qualified set in a secret sharing scheme will be said to *consistently determine* s .

In our protocols, the set of *non-qualified* subsets (the complement of the access structure) will coincide with \mathcal{A} , the subsets of players that may potentially be under the adversary's control. Because we are dealing with only $Q2$ or $Q3$ adversary structures, the set of correct players is always qualified.

We will need the following extra properties from our secret sharing schemes, where the strong multiplication property is only required in the case of an active $Q3$ -adversary:

- *Linearity*: If values $s, s' \in K$ have been shared resulting in sets of shares $\mathbf{s} = (s_1, \dots, s_n)$ resp. $\mathbf{s}' = (s'_1, \dots, s'_n)$, then the set $(s_1 + s'_1, \dots, s_n + s'_n)$ is a consistent set of shares uniformly chosen among those that determine $s + s'$. As for commitments, this property implies that players can, without communicating, also compute from a sharing of s a set of shares determining cs or $c + s$, for any constant $c \in K$.
- *Multiplication property*: assume values $s, s' \in K$ have been shared resulting in sets of shares \mathbf{s} resp. \mathbf{s}' . Note that these sets of shares can be viewed as d -vectors. We require that there exists a fixed d -vector \mathbf{r} , called the *recombination vector*⁵ such that $\langle \mathbf{r}, \mathbf{s} * \mathbf{s}' \rangle = ss'$.
- *Strong multiplication property*: With respect to an adversary structure \mathcal{A} , the following additional property holds: let C be any set of players with $C = \{P_1, \dots, P_n\} - A$ for some $A \in \mathcal{A}$. Let $(\mathbf{s} * \mathbf{s}')_C$ be the d_C -vector obtained by extracting from $\mathbf{s} * \mathbf{s}'$ the coordinates corresponding to players in C . We then require that there exists a d_C -vector \mathbf{r}_C such that $\langle \mathbf{r}_C, (\mathbf{s} * \mathbf{s}')_C \rangle = ss'$.

2.3 The Commitment Distribution Protocol

This protocol is only required in the case of an active adversary and is defined relative to an adversary structure \mathcal{A} and with respect to a secret sharing scheme for which \mathcal{A} consists of the non-qualified sets. It allows a player P_i who has committed to a value $s \in K$ by commitment C to share s among the players P_1, \dots, P_n such that in the presence of any adversary in \mathcal{A} ,

⁵The definition of this property was motivated by an observation of M. Rabin described in [13], and it can in fact be seen as an abstraction or generalization of the multiplication trick described in [13].

1. Either each player P_j for $1 \leq j \leq n$ is committed to (the coordinates of) a d_j -vector \mathbf{s}_j , such that $\mathbf{s}_1, \dots, \mathbf{s}_n$ is a set of shares consistently determining s ; or all correct players reject the outcome of the protocol⁶.
2. If P_i is a correct player, any admissible adversary learns nothing about s .

Any commitment distribution protocol can be used to realize verifiable secret sharing (VSS).⁷ This is achieved by having the dealer first commit to the secret s and then use the commitment distribution protocol. Reconstruction is immediate by having each P_j open the commitments to \mathbf{s}_j : incorrect players cannot contribute false shares, and shares will be available from at least the correct players, which is a qualified set.

3 Multi-Party Computation: The Main Protocol

In this section, we give a bird's eye view of our multi-party computation protocol based on the tools⁸ of Section 2. This view will be valid both in the information-theoretic and the cryptographic scenario.

A remark on broadcast is appropriate here. Note that we may assume throughout that players can broadcast information. This is trivial in the cryptographic scenario, and can be simulated through a protocol in the information-theoretic scenario. This is non-trivial only with an active Q_3 -adversary. In this case a solution follows from using the result of [15]. We conjecture that an alternative solution is to use the protocol of Feldman and Micali [11] together with our VSS protocol described below, but at the time of writing, this has not yet been completely investigated.

Our main protocol has the same overall structure as many known multiparty computation protocols. There are three main parts: the *Input distribution*, *Computation*, and *Output reconstruction* phases. We note that the description is for the case of an active adversary. A (much simpler) description for the case of a passive adversary can be obtained by removing the commitments and subprotocols that force players to act correctly.

After the input distribution phase, each input value x to the computation is represented by a set of shares $\mathbf{x}_1, \dots, \mathbf{x}_n$, such that each P_i has committed to (the coordinates of) his share \mathbf{x}_i . During the computation phase, we work our way through the circuit \mathcal{C} one field operation (multiplication or addition) at a time. Finally, the outputs can be reconstructed since each output value will be represented in the same way as the inputs.

Here follows a more concrete description:⁹

⁶for the cryptographic scenario, a superpolynomially small error probability, taken over the coin flips of correct players, is allowed.

⁷A VSS can be seen as a shared commitment for which it is additionally guaranteed that the secret can efficiently be reconstructed by the players.

⁸In case of a passive adversary, the Commitment Distribution Protocol, the Strong Multiplication for Secret Sharing and the Multiplication Protocol for Commitments are not needed.

⁹This protocol draws on ideas in [13] (see also the end of the introduction) and [8].

1. For each player P_i and each input value x to be chosen by P_i , P_i commits to x and uses the Commitment Distribution Protocol to ensure that each P_j is committed to a valid share \mathbf{x}_j of x .

If a player P_i fails to execute this phase correctly, he is clearly corrupt, and the correct players assume default values for his inputs and shares.

2. For the computation phase, we maintain an invariant stating that whenever y is an input value or an intermediate result in \mathcal{C} that has already been computed, each P_i is committed to a valid share \mathbf{y}_i of y (initially, no intermediate results are computed).

Let x, y be input values to an operation to be done in \mathcal{C} , where both x, y are either inputs or already computed intermediate results. This means that each P_i is committed to valid shares $\mathbf{x}_i, \mathbf{y}_i$ of x, y by two vectors of commitments $X_{i,1}, \dots, X_{i,d_i}$ and $Y_{i,1}, \dots, Y_{i,d_i}$. Let, as before, \mathbf{x} and \mathbf{y} denote the full sets of shares in x and y , and define $z = xy$.

If the operation is *addition*, the players now locally compute for each i the vector of commitments $X_{i,1} \cdot Y_{i,1}, \dots, X_{i,d_i} \cdot Y_{i,d_i}$. This commits P_i to $\mathbf{x}_i + \mathbf{y}_i$ using linearity of commitments. By linearity of the secret sharing, this set of shares consistently determines $x + y$.

If the operation is a *multiplication*, each P_i uses for $j = 1 \dots d_i$ the Multiplication Protocol with inputs $X_{i,j}, Y_{i,j}$ to produce a commitment $W_{i,j}$ to the j -th coordinate of $\mathbf{x}_i * \mathbf{y}_i$. For each $j = 1 \dots d_i$, player P_i now uses the Commitment Distribution Protocol with input $W_{i,j}$ to have each P_k commit to (the coordinates of) a valid share in the j -th coordinate of $\mathbf{x}_i * \mathbf{y}_i$.

3. The following only applies when the previous operation was a *multiplication*. If the secret sharing scheme has only the standard multiplication property, and some P_i fails to complete his multiplication step correctly, he is deemed corrupt, the players make public all the information sent to him so far, and simulate him openly after this point (this tells the adversary nothing he did not already know). Thus we proceed as if P_i still participated in the protocol. If the scheme has the strong multiplication property, a player P_i failing to complete his multiplication step correctly, is simply deemed corrupt and is ignored for the rest of the protocol.

Now, each player does the following (our description is for strong multiplication; in the other case, read C as the full player set, d_C as d , and \mathbf{r}_C as \mathbf{r}).

Let C be the set of participants that still participate at this point. Suppose $(\mathbf{x} * \mathbf{y})_C$ (defined as in Section 2.2) is a d_C -vector. Now, for every $P_k \in C$ and every $l = 1 \dots d_k$, $m = 1 \dots d_C$, player P_k has made a commitment $Z_{k,l,m}$ to the l -th coordinate of a share $z_{k,m}$ in the m 'th coordinate of $(\mathbf{x} * \mathbf{y})_C$. Each P_k

now privately computes $\mathbf{z}_k = r_1 \mathbf{z}_{k,1} + \dots + r_{d_C} \mathbf{z}_{k,d_C}$, and the players compute, for $l = 1 \dots d_k$, the commitments $Z_{k,l} = Z_{k,l,1}^{r_1} \dots Z_{k,l,d_C}^{r_{d_C}}$ to the l -th coordinate of \mathbf{z}_k , where $\mathbf{r}_C = (r_1, \dots, r_{d_C})$ is the recombination vector guaranteed by the strong multiplication property.

Note that by the properties of the secret sharing, z_k is a valid share of $z = xy$, and so we have built a correct representation of z .

4. For the output reconstruction phase, we may assume that each output value is represented by a set of d commitments, containing a full set of shares of the value. Reconstruction is therefore straightforward by opening all these commitments, ignoring values that are not opened correctly.

4 Span Programs and Secret Sharing

4.1 Notation and Definitions

Let $f : \{0, 1\}^n \rightarrow \{0, 1\}$, $f \not\equiv 0, 1$, be a monotone function. A *min-term* x of f is a string with $f(x) = 1$ such that no $x' < x$ (usual Boolean ordering) satisfies $f(x') = 1$. A *max-nonterm* y of f is a string with $f(y) = 0$ and for no $y' > y$ satisfies $f(y') = 0$. Its dual $f^* : \{0, 1\}^n \rightarrow \{0, 1\}$ is defined by $f^*(x) = f(x \oplus \mathbf{1}) \oplus \mathbf{1}$ for all $x \in \{0, 1\}^n$, where $\mathbf{1}$ denotes the all-one bit string and \oplus denotes bit-wise xor. A function f is called *self-dual* if $f = f^*$. Let A denote a subset of $\{1, \dots, n\}$. By abuse of notation we shall take $f(A)$ to mean $f(I_A)$, where I_A is bit string characterizing A , i.e. the i -th bit of I_A is set to 1 if $i \in A$ and set to 0 otherwise. If $f(A) = 1$, we shall say that A is *qualified*. Otherwise, A is *non-qualified*. Hence, the min-terms of f correspond to minimal qualified sets of the associated access structure and the max-nonterms correspond to the maximal non-qualified sets.

If V is a finite dimensional vector space over a field K then $\dim_K V$ denotes its dimension. If W is a subspace, 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 example for $V = K^d$, $\mathbf{x} = (x_1, \dots, x_d) \in V$ and $\mathbf{y} = (y_1, \dots, y_d) \in V$, $\mathbf{x} * \mathbf{y}$ denotes $(x_1 y_1, \dots, x_d y_d) \in V$.

If M is a matrix with entries in K then M^t denotes the transpose of M (we will sometimes use $\text{cols}(M)$ and $\text{rows}(M)$ to refer to the number of rows and columns of a matrix M , respectively). The image of M is denoted $\text{Im } M$ and its kernel (null-space) by $\text{Ker } M$. Finally, a useful fact from elementary linear algebra is that $\text{Im } M^t = (\text{Ker } M)^\perp$.

4.2 Monotone Span Programs

Span programs were introduced by Karchmer and Wigderson [16] as a linear algebraic model of computation. In this paper, we will consider only *monotone* span programs. Karchmer and Wigderson also described how span programs give rise to secret sharing schemes.

4.2.1 Definition

Let K be a finite field and let M be a matrix with entries in K , having d rows and e columns.¹⁰ We assume that M is *labelled* in the sense that each row is indexed by some integer i with $1 \leq i \leq n$, for some n , where every index i between 1 and n occurs at least once as the index of a row. Finally, let $\mathbf{a} \in K^e \setminus \mathbf{0}$ be given. A *monotone span program* is a triple (K, M, \mathbf{a}) , defined as above.¹¹ Sometimes we will treat M just as a matrix, ignoring the labelling.

Let $i \leq n$ be a positive integer. By M_i we denote the matrix consisting of the rows in M indexed by i . For each $1 \leq i \leq n$, write d_i for the number of rows in M_i . Similarly, for $\emptyset \neq A \subset \{1, \dots, n\}$, M_A denotes the matrix consisting of all rows in M indexed with elements $j \in A$, and d_A denotes the number of rows in M_A . For a vector $\mathbf{s} = (s_1, \dots, s_d) \in K^d$, we define \mathbf{s}_i and \mathbf{s}_A similarly.

Let $f : \{0, 1\}^n \rightarrow \{0, 1\}$ be a monotone function. If for all $\emptyset \neq A \subset \{1, \dots, n\}$

$$f(A) = 1 \iff \mathbf{a} \in \text{Im } M_A^t,$$

then the monotone span program (K, M, \mathbf{a}) is said to compute f .

We note that any given monotone span program computes a monotone Boolean function in a natural way. Namely, it computes the monotone Boolean function f defined $f(x_1, \dots, x_n) = 1$ if and only if $\mathbf{a} \in \text{Im } M_A^t$ where $A = \{1 \leq i \leq n | x_i = 1\}$. Furthermore, it is well known that any monotone Boolean function can be computed by a monotone span program.

4.2.2 A Secret Sharing Scheme

Let $f(x_1, \dots, x_n)$ be a monotone Boolean function and let (K, M, \mathbf{a}) be a monotone span program computing it. Wlog we may assume that $\mathbf{a} = (10 \cdots 0)$ (by applying a suitable linear transformation on the rows). A perfect secret sharing scheme for f is now constructed as follows [16] (notations as above).

Share Distribution:

(K, M, \mathbf{a}) is public knowledge. Let $s \in K$ be the secret. The dealer chooses $\rho_1, \dots, \rho_{e-1} \in K$ at random and puts $\mathbf{b} \leftarrow (s, \rho_1, \dots, \rho_{e-1})$. For $i = 1 \dots n$, he sends the share \mathbf{s}_i , computed as $\mathbf{s}_i \leftarrow M_i \mathbf{b}$, privately to player i . Note that $\mathbf{s}_i \in K^{d_i}$.

Reconstruction:

Let $A \subset \{1, \dots, n\}$ satisfy $f(A) = 1$, and let $\boldsymbol{\lambda}_A$ satisfy $M_A^t \boldsymbol{\lambda}_A = \mathbf{a}$. Then we have $s = \langle \boldsymbol{\lambda}_A, \mathbf{s}_A \rangle$, where \mathbf{s}_A denotes the superposition of the \mathbf{s}_i with $i \in A$. Note that $\boldsymbol{\lambda}_A, \mathbf{s}_A \in K^{d_A}$, where $d_A = \sum_{i \in A} d_i$.

¹⁰We may assume wlog that $e \leq d$. Keeping only a collection of columns (including the first) that span the column space does not affect the multiplication property defined later.

¹¹Note that the labelling of M is only implicit in this definition.

For completeness, we prove the scheme correct and secure. Correctness: For A with $f(A) = 1$ we have $s = \langle \mathbf{a}, \mathbf{b} \rangle = \langle M_A^t \boldsymbol{\lambda}_A, \mathbf{b} \rangle = \langle \boldsymbol{\lambda}_A, M_A \mathbf{b} \rangle = \langle \boldsymbol{\lambda}_A, \mathbf{s}_A \rangle$. Privacy: Observe that when $f(A) = 0$ $\mathbf{a} \notin \text{Im } M_A^t$ implies that, by our choice of \mathbf{a} and by the fact that $\text{Im } M_A^t = (\text{Ker } M_A)^\perp$, $\text{Ker } M_A$ 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 the ρ_i 's appearing in the definition of \mathbf{b} have been chosen at random by the dealer.

As a final remark, if we do not take $\mathbf{a} = (10 \cdots 0)$, then the dealer chooses \mathbf{b} at random s. t. $\langle \mathbf{a}, \mathbf{b} \rangle = s$. The changes for reconstruction and the proof are trivial.

4.3 Span Programs with Multiplication

Recall the definition of the (strong) multiplication property for linear secret sharing schemes from Section 2.2. For completeness, we now give a formal definition in terms of monotone span programs.

DEFINITION 1 *We say that $(K, M, \mathbf{a}, \mathbf{r})$ is a monotone span program with multiplication if (K, M, \mathbf{a}) is a monotone span program and the recombination vector \mathbf{r} has the property that for all \mathbf{b}, \mathbf{b}' we have*

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

Strong multiplication is defined analogously.

We state a property of the recombination vector that we will use later. Let $f(x_1, \dots, x_n)$ be the monotone Boolean function computed by $(K, M, \mathbf{a}, \mathbf{r})$. Say $\mathbf{r} = (\mathbf{r}_1, \dots, \mathbf{r}_n)$, and define $\text{sup } \mathbf{r} = \{1 \leq k \leq n \mid \mathbf{r}_k \neq \mathbf{0}\}$. Then we have $f(\text{sup } \mathbf{r}) = 1$. It is easy to see that $f(\text{sup } \mathbf{r}) = 0$ would imply that the non-qualified set $\text{sup } \mathbf{r}$ could compute the secret in any execution of the secret sharing scheme, which is a contradiction.¹²

4.4 Constructions of Span Programs

We characterize the monotone Boolean functions f computable by monotone span programs with multiplication. We also give an upper bound of the minimal achievable size of the monotone span program in terms of certain logical formulae computing f .

DEFINITION 2 *Let $f : \{0, 1\}^n \rightarrow \{0, 1\}$ be a monotone function. We say [15] that f is Q2-monotone if for all $A, A' \subset \{1, \dots, n\}$ with $f(A) = f(A') = 0$ we have $A \cup A' \neq \{1, \dots, n\}$.*

¹²Furthermore, by elementary linear algebra, it is easy to show that the multiplication property is preserved under linear transformations on the rows of the monotone span program.

By the above definition, $f(A) = f(A^c) = 0$ is impossible. Also, let $f(B) = 1$ and $f(A) = 0$. Then $A^c \cap B = \emptyset$ would imply $B \subset A^c$. Hence, $f(A^c) = 1$, a contradiction. We have the following lemma.

LEMMA 1 *For any Q2-monotone function f , and for any B with $f(B) = 1$ and A with $f(A) = 0$, we have $f(A^c) = 1$ and $A^c \cap B \neq \emptyset$.*

PROPOSITION 1 *Let $(K, M, \mathbf{a}, \mathbf{r})$ be a monotone span program with multiplication computing the monotone Boolean function f . Then f is Q2-monotone.*

PROOF. Suppose not. Then there exists a non-empty set A such that $f(A) = f(A^c) = 0$. This implies that there exists a vector $\boldsymbol{\kappa}$ whose first coefficient is equal to 1, such that $M_{A^c}\boldsymbol{\kappa} = \mathbf{0}$ (assuming wlog that $\mathbf{a} = 10 \cdots 0$). Then we have for each \mathbf{b} (write s for its first coefficient) that $\langle \mathbf{r}, M\boldsymbol{\kappa} * M\mathbf{b} \rangle = 1 \cdot s$. But since $M_{A^c}\boldsymbol{\kappa} = \mathbf{0}$, s must be computable from \mathbf{r} (public information) and $M_A\mathbf{b}$. But this contradicts the security of the secret sharing scheme from Section 4.2. \triangle

We will also prove that the converse is true. To this end we first address how to construct new monotone span programs (with or without multiplication) from old. Let f, g_1, \dots, g_n be any monotone Boolean functions such that each of them can be computed by span programs with multiplication. Say that f reads n bits, and the g_i 's read m_i bits (i.e. f and the g_i 's have n and m_i literals, respectively). Write $(K, F, \mathbf{a}, \mathbf{r}_0), (K, G_i, \mathbf{a}, \mathbf{r}_i), i = 1 \dots n$, for their respective span programs (assume wlog that for each of these programs \mathbf{a} denotes the first unit vector in each of their respective vector spaces of definition). Write F_i for the rows of F that correspond to the i -th literal of f (i.e. indexed by i). The following proposition is proved in the Appendix.

PROPOSITION 2 *$f(g_1, \dots, g_n)$ is computable by a monotone span program $(K, M, \mathbf{a}, \mathbf{r})$ with multiplication. Furthermore, $\text{rows}(M) \leq \sum_{i=1}^n \text{rows}(F_i) \cdot \text{rows}(G_i)$.*

Remark. The claim in our result also holds if we disregard the multiplication property. Viewed in that way, our result is a slight improvement for the monotone span program complexity upper bound given in [18], Theorem 3.4, 4-th claim. However, our main goal is to incorporate multiplication.

DEFINITION 3 *Let the threshold function ¹³ $f_{t,n} : \{0,1\}^n \rightarrow \{0,1\}$ output 1 if and only if the input string has Hamming-weight at least t . We call $f_{t,n}$ majority accepting if $2t \leq n + 1$. If $3t \leq n + 2$, we call $f_{t,n}$ 1/3-accepting.*

LEMMA 2 *Let $f_{t,n}$ be an arbitrary threshold function. Then $f_{t,n}$ can be computed by a monotone span program (K, M, \mathbf{a}) having n rows, provided that $|K| > n$ if $t > 1$. If additionally $f_{t,n}$ is majority accepting, then the monotone span program is with multiplication.*

¹³Note that in general the dual of a threshold function $f_{t,n}$ is the threshold function $f_{n-t+1,n}$.

PROOF. The first part of the claim (not dealing with multiplication) is well-known. If $t = 1$, we have the trivial monotone span program $(K, 1, 1)$. 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 of the form $1, \alpha_i, \dots, \alpha_i^{t-1}$ and $\alpha_i \neq \alpha_j$ if $i \neq j$. It is easy to see that $M_{t,n}$ can be viewed as a span program computing $f_{t,n}$, with root $\mathbf{a} = (10 \cdots 0)$. In this case the associated secret sharing scheme (as defined in Section 4.2) is identical to Shamir's [21].

To see that $(K, M_{t,n}, \mathbf{a})$ 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 $P_1, \dots, P_n \in K \setminus \mathbf{0}$, and the evaluations $h(P_1), \dots, h(P_n)$, 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) . This part of the claim is due to M. Rabin [13]. We cast it here in our model. \triangle

PROPOSITION 3 *The class of Q2-monotone Boolean functions coincides with the class of functions computable by Boolean formulas consisting of majority accepting gates.*

PROOF. Let Φ be a formula consisting of majority accepting gates. Suppose that (the monotone function computed by) Φ is not Q2. Then there is a set A with $\Phi^*(A) = \Phi^*(A^c) = 1$. But in each gate of Φ^* (obtained by dualizing the gates of Φ), the threshold is larger than in the corresponding gate of Φ . Thus $\Phi^*(A)$ implies $\Phi(A) = 1$, a contradiction. The other direction of the proof is given in the Appendix. \triangle

DEFINITION 4 *The size $|\Phi|$ of a logical formula Φ is defined as the number of input wires to Φ . For any monotone Boolean function f , the size of the smallest formula computing f and consisting of any threshold gates $f_{t,n}$ is denoted $\phi(f)$. If f is computable by a formula with majority accepting gates (resp. 1/3-accepting gates), then $\phi_{ma}(f)$ (resp. $\phi_{1/3}(f)$) is defined similarly to $\phi(f)$.*

We are now ready to prove the main claims for this section.

THEOREM 1 *A monotone Boolean function f has a span program with multiplication if and only if f is Q2-monotone. Any Q2-monotone function f can be computed by a monotone span program with multiplication, having at most $\phi_{ma}(f)$ rows and over any field K with¹⁴ $|K| > \phi_{ma}(f)$.*

¹⁴It is sufficient that $|K| > N$, where N is the largest fan-in among the gates with threshold greater than 1.

PROOF. Proposition 1 takes care of one direction of the first claim. Now if f is Q2-monotone, it can be computed by a formula Φ with majority accepting gates by Proposition 3. By applying Proposition 2 recursively (applying Lemma 2 to the gates of Φ and choosing the field K large enough), we obtain a span program with multiplication computing the same function as Φ , having as many rows as Φ has input wires. \triangle

If we only consider ordinary monotone span programs (not caring about multiplication), then we have from Proposition 2 and the well-known fact that any monotone function can be computed by a monotone span program over any finite field:

PROPOSITION 4 *Every monotone Boolean function f can be computed by a monotone span program with at most $\phi(f)$ rows, over any field K with $|K| > \phi(f)$.*

Finally, we define Q3-monotone functions, a subclass of the Q2-monotone functions. These functions are important for our results in Section 5.

DEFINITION 5 *Let $f : \{0, 1\}^n \rightarrow \{0, 1\}$ be a monotone function. We say [15] that f is Q3 if for all $A, A', A'' \subset \{1, \dots, n\}$ with $f(A) = f(A') = f(A'') = 0$ we have $A \cup A' \cup A'' \neq \{1, \dots, n\}$.*

PROPOSITION 5 *The class of Q3-monotone functions coincides with the class of functions computable by Boolean formulas consisting of 1/3-accepting gates.*

PROOF. Let $\Phi = f_{t,m}(\Phi_1, \dots, \Phi_m)$ where $3t \leq m + 2$ and the Φ_j 's are formulas with 1/3-accepting gates. Note that $f_{t,m}$ is Q3. By induction on the number of gates, we prove that Φ is Q3. Say A_i , $i = 1, 2, 3$, contradict the claim. Then for each A_i , $\Phi^*(A_i^c) = 1$. Let $W_i = \{j | \Phi_j^*(A_i^c) = 1\}$. We have $|W_i| \geq m - t + 1$. But since $3t \leq m + 2$, the intersection W of the W_i 's is non-empty. Let $j \in W$. Then $\Phi_j(A_i) = 0$, $i = 1, 2, 3$, and the A_i cover the literals read by Φ_j . A contradiction. The other part of the proof is quite similar to that of Proposition 3, using $f_{1,2}$ -, $f_{1,3}$ -, and $f_{2,4}$ -gates (in the recursion, use $f_{3,4}$ -gates and replace them by $f_{2,4}$ -gates afterwards). \triangle

The following theorem is proved in the Appendix.

THEOREM 2 *A monotone Boolean function f has a monotone span program with strong multiplication if and only if f is Q3-monotone. Any Q3-monotone function f can be computed by a monotone span program with strong multiplication, having at most $\phi_{1/3}(f)$ rows and over any field K with $|K| > \phi_{1/3}(f)$.*

5 Multi-Party Computation in the Information-Theoretic Scenario

5.1 Passive Adversaries

Let $f(x_1, \dots, x_n)$ be a Q_2 -monotone Boolean function and let $(K, M, \mathbf{a}, \mathbf{r})$ with multiplication be a monotone span program computing it. We show how a multi party computation in the secure channels setting among n players allows them to securely compute the product of s and s' . Refer to Section 3 for the the general description of the protocol.

Let M_i denote the d_i rows in M (with d rows and e columns) that are associated with x_i , and let R_i denote their indices, $i = 1, \dots, n$. For any $s, s' \in K$ and for any $\sigma, \sigma' \in K^{e-1}$, let $\mathbf{s} = (s_1, \dots, s_d)$ denote $M(s, \sigma)$ and let $\mathbf{s}' = (s'_1, \dots, s'_d)$ denote $M(s', \sigma')$. Let $\mathbf{t} = (t_1, \dots, t_d)$ denote $\mathbf{s} * \mathbf{s}'$. Let $\mathbf{r} = (r_1, \dots, r_d)$ be the recombination vector. Each player i initially holds $\mathbf{s}_i = M_i(s, \sigma)$ and $\mathbf{s}'_i = M_i(s', \sigma')$, $i = 1, \dots, n$. Each player i does the following

Re-Sharing: For each $k \in R_i$ choose $\rho_k \in K^{e-1}$ at random. Then, for each $1 \leq j \leq n$, compute $\mathbf{u}_{kj} \leftarrow M_j(t_k, \rho_k)$, and send \mathbf{u}_{kj} privately to player j .

Recombination: Each player i computes

$$\mathbf{v}_i \leftarrow \sum_{k=1}^d r_k \mathbf{u}_{ki}.$$

Product-Reconstruction: Each player i broadcasts \mathbf{v}_i .

As for its *correctness*, note that:

$$\begin{aligned} (\mathbf{v}_1, \dots, \mathbf{v}_n) &= (\sum_{k=1}^d r_k \mathbf{u}_{k1}, \dots, \sum_{k=1}^d r_k \mathbf{u}_{kn}) \\ &= (\sum_{k=1}^d r_k M_1(t_k, \rho_k), \dots, \sum_{k=1}^d r_k M_n(t_k, \rho_k)) \\ &= \sum_{k=1}^d r_k M(t_k, \rho_k) = M(ss', \sum_{k=1}^d r_k \rho_k). \end{aligned}$$

As to *security*, let A be a non-qualified set, i.e. $f(A) = 0$. The situation for A is equivalent to the following. The passive collusion A receives $M_A(s, \sigma)$, $M_A(s', \sigma')$ and $M_A(ss', \sum_{k=1}^d r_k \rho_k)$, where the ρ_j with $j \in A$ are chosen by the players in A . First, by the observations in Section 4.3 there exists k such that $r_k \neq 0$ and the k -th row of M does not belong to any of the players in A . Hence, from the point of view of A the latter is just a random (independent from anything else) sharing. The sharings of s and s' are random as well. Hence, right after the re-sharing stage, the players in A have no information about s , s' or ss' . Finally, after the product-reconstruction, the players in A indeed learn ss' . But since the sharings of s and s' are independent of that of ss' , they learn nothing beyond the product of s and s' .

5.2 Active Adversaries

5.2.1 How to Do Commitments

We first present a generic transformation that converts a span program secret sharing scheme ¹⁵ (see Section 4.2) into another such scheme. Schemes of this

¹⁵The multiplication property for monotone span programs is not needed here.

new type will serve as the basis of our commitment and commitment distribution protocols. They are in fact *robust against cheaters* when the dealer is honest, but this will not be needed here. The proofs are given in the Appendix.

Let $f(x_1, \dots, x_n)$ be a $Q3$ -monotone function and let (K, M, \mathbf{a}) be a monotone span program computing f , where M has d rows and e columns.

Share Distribution:

(K, M, \mathbf{a}) is public knowledge. 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 $i = 1 \dots n$, the dealer puts $U_i \leftarrow M_i R$, and sends the matrix U_i privately to player i . The actual share \mathbf{s}_i of player i is defined as the first column of U_i . Note that $\mathbf{s}_i = M_i \mathbf{b}$, where \mathbf{b} is the first column of R (whose first entry is s).

Reconstruction:

Each player i broadcasts \tilde{U}_i (supposedly what the dealer sent). 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 as in Section 4.2.

PROPOSITION 6 *The transformed scheme is a secret sharing scheme for f as well. For each i, j , we have $M_i U_j^t = U_i M_j^t$. Moreover, the scheme is robust against cheaters if the dealer is honest.*

We will now extend the above scheme such that even with a faulty dealer, 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 player i . Hence, by sending $M_j U_i^t$ to player j , player i demonstrates that he has shares in share \mathbf{s}_j belonging to player j .

The protocol for distributing and checking shares is defined as follows.

Step 1: Let $s \in K$ be the secret to be distributed by the dealer. Privately to each player i , the dealer sends the matrix U_i . The actual share \mathbf{s}_i of player i is again defined as the first column of U_i .

Step 2: Each player i puts, for $j = 1 \dots n$, $\text{check}_{ij} \leftarrow U_i M_j^t$, and sends the matrix check_{ij} privately to player j , who verifies that $\text{check}_{ij} = M_i U_j^t$. Note that check_{ij} has d_i rows and e columns. If player i did not receive U_i of the right format, player i sends ϵ (representing the empty string) to player j .

Step 3: For each player j , if player j finds disagreements in values received from a qualified set of players, he broadcasts an *accusation against the dealer*, asking him to make public all information sent to player j (since he can now conclude that the dealer must be faulty). If there are only disagreements with values

received from a non-qualified set, player j broadcasts a complaint, asking the dealer to make public those check_{ij} for which he received a wrong value from player i .

Step 4: In response to the complaints and accusations broadcast, the dealer must make public all values asked for. All players check what they received against the information made public, and accuse the dealer if there is a disagreement. Now, the dealer must make public everything he sent to the new accusing players. This process goes on until no new accusations are made.

Step 5: If at this point a qualified set has accused the dealer, or if the information made public by the dealer contradicts itself, the players take a fixed default set of shares to represent the dealer's secret (in this case the dealer is clearly faulty). Otherwise, the complaining players take the public information as their shares.

The following two theorems are proved in the Appendix.

THEOREM 3 *If the dealer is a correct player, the adversary learns nothing about s , and the shares held by correct players consistently determine s .*

THEOREM 4 *For any dealer and any active adversary, the above protocol results in the correct players holding consistent shares \mathbf{s}_i of some secret s .*

These two theorems enable us to establish the commitments we need. The COMMIT protocol works as follows. To commit to value a , we execute the above share distribution protocol for $s = a$ where the committer acts as the dealer. From this phase, the players only need to remember the actual shares $(\mathbf{s}_1, \dots, \mathbf{s}_n)$ that they received.

As to the OPEN protocol, to open a commitment the committer broadcasts a , $\mathbf{s}'_1, \dots, \mathbf{s}'_n$, and claims that a is the value committed to. Each player i checks whether the set of \mathbf{s}'_j 's consistently determines a and whether $\mathbf{s}'_i = \mathbf{s}_i$. If any of these verifications fail, player i broadcasts a complaint. If all but a non-qualified set of players agree, the opening is accepted, otherwise it is rejected.

Theorems 3, 4, and the Q3 property trivially implies the hiding and binding properties required, and the linearity of the span program used directly translates into the homomorphic property we need for commitments: to add committed values, the players add their shares. The multiplication protocol required is described below.

5.2.2 Commitment Distribution Protocol

A dealer be committed to value $z \in K$ by commitment Z .

Step 1: The dealer makes shares $\mathbf{z}_1, \dots, \mathbf{z}_n$ of secret z (using the underlying secret sharing scheme) and commits to each of them using the commitment protocol.

Step 2: For $i = 1, \dots, n$, the commitment to \mathbf{z}_i is opened to P_i , and all players send their own share of (each coordinate of) \mathbf{z}_i to P_i . We now do the following stages to verify that P_i agrees with all correct players on the values received from the dealer:

1. P_i compares what the dealer sent with the shares received from the players. If there is disagreement w.r.t. a qualified set, P_i asks the dealer to open the commitments to \mathbf{z}_i publically. Otherwise P_i asks the dealer to make public the shares for which there was disagreement.
2. The dealer broadcasts the requested information. If this contradicts itself, or is incomplete, the dealer is deemed corrupt. Otherwise, if P_i finds that the public information contradicts his own, he accuses the dealer, who must then open the commitments to \mathbf{z}_i publically. Again, the dealer is deemed corrupt, if the openings fail, otherwise the players take whatever is now public to be their share of \mathbf{z}_i .

Step 3: At this point each P_i knows a correct share \mathbf{z}_i , he knows all shares of (coordinates of) \mathbf{z}_i distributed, and agrees with all correct players on these values. The only missing thing is that we don't know if the set of \mathbf{z}_i 's is consistently determining z . However, a set of \mathbf{z}_i 's is consistent precisely if some set of linear combinations of them are all zero. The coefficients of these combinations can be computed from the span program matrix.

Therefore, using the linearity property, the players compute the corresponding linear combinations of the commitments to the \mathbf{z}_i 's, and the dealer has to open all of the resulting commitments as zero. If this fails, the dealer is deemed corrupt. Finally we compute a linear combination of commitments to a qualified set of \mathbf{z}_i 's, which should reconstruct a commitment C to z . We check that Z and C contain the same value by computing a commitment containing the difference of values in C and Z , which the dealer must open to reveal 0.

It is straightforward to verify that this protocol has the properties required in Section 2. Details are left to the reader.

5.2.3 Multiplication Subprotocol

LEMMA 3 *Suppose values s, s' have been shared with sets of shares \mathbf{s} , resp. \mathbf{s}' , using span program matrix M , with root vector $(1, 0, \dots, 0)$. Consider the vector $\mathbf{s} * \mathbf{s}'$. Then this is a set of shares of the value ss' in a new span program based secret sharing scheme, with matrix M' and root vector $(1, 0, \dots, 0)$, where M' can be computed from the span program matrix M that we started from. Moreover, if M has the strong multiplication property, then any potential set of correct players is qualified w.r.t. M' .*

PROOF. Let $\mathbf{v} = (v_1, \dots, v_e)$ be any e -vector over K . Define the $(e^2 + e)/2$ -vector $\tilde{\mathbf{v}}$ by:

$$\tilde{\mathbf{v}} = (v_1v_1, v_1v_2, \dots, v_1v_e, v_2v_2, v_2v_3, \dots, v_2v_e, \dots, v_e v_e)$$

Also, for any two e -vectors \mathbf{a}, \mathbf{b} , let

$$\mathbf{a} \diamond \mathbf{b} = (a_1b_1, a_1b_2 + a_2b_1, \dots, a_1b_e + a_eb_1, \\ a_2b_2, a_2b_3 + a_3b_2, \dots, a_2b_e + a_eb_2, \dots, a_eb_e)$$

It is now straightforward to check that

$$\langle \mathbf{v}, \mathbf{a} \rangle \cdot \langle \mathbf{v}, \mathbf{b} \rangle = \langle \tilde{\mathbf{v}}, \mathbf{a} \diamond \mathbf{b} \rangle$$

Now, let $\mathbf{v}_1, \dots, \mathbf{v}_d$ be the rows of M . Define M' by letting its rows be $\tilde{\mathbf{v}}_1, \dots, \tilde{\mathbf{v}}_d$. The lemma now follows by direct inspection. \triangle

We are now ready to give the multiplication protocol, for which we are given commitments A, B made by P_i to values a, b

Step 1: P_i uses the commitment distribution protocol twice with inputs A , resp. B , so that now each P_j is committed to shares $\mathbf{a}_j, \mathbf{b}_j$ of a, b .

Step 2: P_i makes a vector of commitments to $\mathbf{a}_j * \mathbf{b}_j$, for $j = 1..n$. These commitments are opened to P_j , i.e. all shares involved in committing to $\mathbf{a}_j * \mathbf{b}_j$ are sent to P_j , and P_i sends the values he claims for all these shares. If P_j finds that a qualified set of players disagrees with the values received from P_i , or if the commitments did not contain the right values ($\mathbf{a}_j * \mathbf{b}_j$), P_j accuses the dealer and proves his case by opening publically his own commitments to $\mathbf{a}_j, \mathbf{b}_j$. P_i must now open the commitments to $\mathbf{a}_j * \mathbf{b}_j$ in public; if this is not correct, he is deemed corrupt.

Step 3: Let D be vector of all commitments made by P_i in step 2. D should contain a consistent set of shares in generated from span program matrix M' , by the above lemma. This can be checked by verifying that some fixed set of linear combinations of the entries in D all equal 0. The coefficients of these linear combinations can be computed from the matrix of M' . Hence P_i can prove that the shares committed to in D are consistent by opening a number of commitments to reveal 0. Finally we compute a linear combination of the commitments in D using the entries in the recombination vector of M as coefficients. The resulting commitment, C is the output of the protocol (and is guaranteed to contain ab).

This protocol has the required properties, since by the consistency check in step 3, P_i has distributed in D consistent shares of some secret s using matrix M' . But by step 2, P_i must in fact distribute $\mathbf{a}_j * \mathbf{b}_j$ to each correct P_j , or be disclosed as corrupt. And since the set of correct players is qualified in M' , we must have $s = ab$. It is also straightforward to check that if P_i is correct, C is a random commitment with the only constraint that it contains ab .

6 Cryptographic Multi-Party Computations

Let f be any Q_2 -monotone Boolean function. We will prove that in the cryptographic setting, one can perform general multiparty computations tolerating any single non-qualified (w.r.t. f) set of *active* corrupted players. This is what Hirt and Maurer [15] prove to be the tolerance against *passive* adversaries in the *secure channels setting*.

According to Section 3 and given the results in Section 4, it is sufficient to describe the necessary commitment scheme and the Commitment Distribution Protocol.

6.1 How to Do Commitments

We will use the cryptographic commitment schemes from [10], which are based on *q-one-way group homomorphisms* (q -OWGH). These commitments satisfy the requirements from Section 2.1, i.e. they are homomorphic and have a zero-knowledge multiplication subprotocol. The RSA, DL or Diffie-Hellman assumptions are sufficient for efficiently realizing these commitments. In some implementations, the prime q can be chosen independently of the security parameter for the commitment scheme.

We will assume that commitments are *unconditionally binding*, i.e. the value committed to is uniquely determined from the commitment. In [10], also commitments that are *unconditionally hiding* are proposed. We may also use that type of commitments to support our protocol, but this requires a somewhat different set-up phase and is omitted here for lack of space.

Each of the n players in the multiparty computation protocol generates his own instance of the commitment scheme (but all for the same q) and all broadcast the “public key” of their instances: this amounts to fixing, for $i = 1 \dots n$, a (probabilistic) commitment function ϕ_i , i.e., player i commits to a value $a \in \mathbb{Z}_q$ by broadcasting $C \leftarrow \phi(\alpha, a)$, where α is a random string.

We note that a slight variation of the multiplication sub-protocol guaranteed by [10] can be used to show in zero-knowledge that one knows how to open a given commitment, or that two commitments contain the same value, even if the public keys involved differ.

Finally, these zero knowledge protocols require random challenges to be generated such that they are unpredictable by the prover. This can for instance be done by straight-forward multi-party coin-flipping based on the commitment schemes.

6.2 Commitment Distribution Protocol

We build a commitment distribution protocol generalizing Pedersen’s non-interactive verifiable secret sharing scheme [17]. A related construction for the threshold case was independently discovered in [13]. Our scheme works for any Q_2 -monotone function instead of “minority” only and is based on the existence of any q -homomorphic commitment scheme.

We will assume that each player initially publishes a public key for a (polynomially) secure public-key encryption system. Existence of such a system follows from any of the particular assumptions we can base our commitments on. In some cases, such as when commitments are based on Diffie-Hellman encryption, this public key can be the same as the one for commitments, i.e. ϕ_i can be used as encryption function when sending messages to player i . In such a case, the description below can be simplified considerably.

For technical reasons (i.e. to be able to prove security of our protocol by a simulation argument), we assume that whenever a player commits to a value or sends an encryption in the protocol below, he also proves in zero-knowledge that he knows the committed or encrypted value.

Let $f(x_1, \dots, x_n)$ be Q_2 -monotone and say that $(GF(q), M, \mathbf{a})$ is a monotone span program computing f . Wlog $\mathbf{a} = (10 \dots 0)$. As usual M has d rows and e columns. The (i, j) -th entry of M is represented by an integer m_{ij} with $0 \leq m_{ij} \leq q - 1$. Suppose we are given a q -homomorphic commitment scheme. Let \mathcal{C} denote the collection of all possible commitments. Then M induces a map $M^{\mathcal{C}}$ from \mathcal{C}^e to \mathcal{C}^d as follows. Let $\mathbf{C} = (C_1, \dots, C_e) \in \mathcal{C}^e$. For $i = 1, \dots, d$ the i -th coordinate of $M^{\mathcal{C}}\mathbf{C}$ is equal to $\prod_{j=1}^e C_j^{m_{ij}}$.

The Commitment Distribution Protocol is as follows, where $C = \phi_i(\alpha, a)$ is the input to the protocol:

Distribution: Player i chooses $\rho_2, \dots, \rho_e \in GF(q)$ at random, puts $\mathbf{b} \leftarrow (a, \rho_2, \dots, \rho_e)$, and computes $(s_1, \dots, s_d) \leftarrow M\mathbf{b}$.

For $j = 2 \dots e$, he chooses α_j as a random string and puts $\mathbf{C} \leftarrow (C, C_2, \dots, C_n)$, where $C_j \leftarrow \phi_i(\alpha_j, \rho_j)$ for $j = 2, \dots, n$.

Finally, for $j = 1, \dots, d$ he computes some strings β_j satisfying that $M^{\mathcal{C}}\mathbf{C} = (\phi_i(\beta_1, s_1), \dots, \phi_i(\beta_d, s_d))$. It follows by definition of the commitments in Section 2.1 (when viewed as *cryptographic* commitments) that computation of these β_j is feasible.

For $j = 1 \dots n$, he puts $\mathbf{s}_j \leftarrow (\cdot, \dots, s_k, \dots, \cdot)_{k \in R_j}$ and $\mathbf{t}_j \leftarrow (\cdot, \dots, \beta_k, \dots, \cdot)_{k \in R_j}$, where R_j is the set of rows in M associated with player j as in Section 5.1. Finally, player i broadcasts \mathbf{C} , and for $j = 1 \dots n$, he sends \mathbf{s}_j and \mathbf{t}_j privately to player j , using secure public key encryption.

Verification: Each player j verifies that $M_j^{\mathcal{C}}\mathbf{C} = (\cdot, \dots, \phi_i(\beta_k, s_k), \dots, \cdot)_{k \in R_j}$. If there is an inconsistency, broadcast a complaint. The dealer must then make public \mathbf{s}_j and \mathbf{t}_j , and open the encryption sent to P_j in the previous step. If this is correct and consistent, P_j takes \mathbf{s}_j as his share, otherwise the dealer is deemed corrupt.

Finally, P_j makes commitments to each value in \mathbf{s}_j using his own function ϕ_j and new independently chosen random input. He then proves (in zero-knowledge) that these new commitments contain the same values as $M_j^{\mathcal{C}}\mathbf{C}$.

The analysis of this scheme is essentially the same as that of Pedersen's, taken into account that we work with a secret sharing scheme based on span programs for a Q_2 -monotone function and abstract q -homomorphic commitments, instead of with Shamir's secret sharing scheme and homomorphic commitments based on discrete logarithms.

Acknowledgements

We would like to thank Anna Gál, Rosario Gennaro, Martin Hirt, 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] J. Benaloh, J. Leichter: *Generalized Secret Sharing and Monotone Functions*, Proc. of Crypto '88, Springer Verlag LNCS series, pp. 25–35.
- [4] M. Ben-Or, S. Goldwasser, A. Wigderson: *Completeness theorems for Non-Cryptographic Fault-Tolerant Distributed Computation*, Proc. ACM STOC '88, pp. 1–10.
- [5] E. F. Brickell: *Some Ideal Secret Sharing Schemes*, J. Combin. Maths. & Combin. Comp. 9 (1989), pp. 105–113.
- [6] R. Canetti: *Studies in Secure Multiparty Computation and Applications*, Ph. D. thesis, Weizmann Institute of Science, 1995.
- [7] R. Canetti, U. Feige, O. Goldreich, M. Naor: *Adaptively Secure Multi-Party Computation*, Proc. ACM STOC '96, pp. 639–648.
- [8] D. Chaum, C. Crépeau, I. Damgård: *Multi-Party Unconditionally Secure Protocols*, Proc. of ACM STOC '88, pp. 11–19.
- [9] 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.
- [10] R. Cramer, I. Damgård: *Zero Knowledge for Finite Field Arithmetic or: Can Zero Knowledge be for Free?*, manuscript, June 1997.
- [11] P. Feldman, S. Micali: *An Optimal Probabilistic Protocol for Synchronous Byzantine Agreement*, SIAM J. Comp. Vol. 26, No. 4, pp. 873–933, August 1997.
- [12] R. Gennaro: *Theory and Practice of Verifiable Secret Sharing*, Ph.D. thesis, MIT, 1996.
- [13] R. Gennaro, M. Rabin, T. Rabin, *Simplified VSS and Fast-Track Multiparty Computations with Applications to Threshold Cryptography*, submitted to STOC'98.
- [14] 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.
- [15] M. Hirt, U. Maurer: *Complete Characterization of Adversaries Tolerable in General Multiparty Computations*, Proc. PODC '97.
- [16] M. Karchmer, A. Wigderson: *On Span Programs*, Proc. of Structure in Complexity, 1993.
- [17] T. P. Pedersen: *Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing*, Proc. CRYPTO '91, Springer Verlag LNCS, vol. 576, pp. 129–140.

- [18] P. Pudlák, J. Sgall: *Algebraic Models of Computation and Interpolation for Algebraic Proof Systems* Proc. Feasible Arithmetic and Proof Complexity, Springer Verlag LNCS series.
- [19] T. Rabin: *Robust Sharing of Secrets when the Dealer is Honest or Cheating*, J. ACM, 41(6):1089-1109, November 1994.
- [20] T. Rabin, M. Ben-Or: *Verifiable Secret Sharing and Multiparty Protocols with Honest majority*, Proc. ACM STOC '89, pp. 73–85.
- [21] A. Shamir: *How to Share a Secret*, Communications of the ACM 22 (1979) 612–613.

Appendix

Proof of Proposition 2. Write $h = f(g_1, \dots, g_n)$. Think of the following secret sharing scheme for h . Let s be a secret to be distributed. Then first distribute s using the matrix F to get shares $\mathbf{s}_1, \dots, \mathbf{s}_n$. Second, for $i = 1 \dots n$, take the share \mathbf{s}_i and distribute its coordinates one by one and independently using the matrix G_i . The matrix M to be defined hereafter captures exactly this linear map. With this image in mind, it is not so difficult to convince oneself that the multiplication property is also satisfied.

Let x_{il} denote the l -th literal of g_i , for $i = 1 \dots, n$, $l = 1 \dots m_i$. Let $(\cdot, \dots, \cdot, \dots, \cdot)$ be a vector whose first location we number 0 and subsequent locations (i, k) with $1 \leq i \leq n$ and $1 \leq k \leq \text{rows}(F_i)$ (hence there are $1 + \text{rows}(F)$ locations). The rows of M will be vectors whose locations are indexed as above. Next, let i, j, k be any integers satisfying $1 \leq i \leq n$, $1 \leq j \leq \text{rows}(G_i)$, $1 \leq k \leq \text{rows}(F_i)$.

The rows of M will consist of vectors \mathbf{m}_{ijk} to be defined hereafter. Consider $(w_{ij}\mathbf{v}_{ik}, \mathbf{w}_{ij})$, where \mathbf{v}_{ik} is the k -th row of F_i , and $(w_{ij}, \mathbf{w}_{ij})$ is the j -th row of G_i (i.e. w_{ij} is the first coordinate of that row and \mathbf{w}_{ij} collects the remaining coordinates). The row \mathbf{m}_{ijk} of M is constructed as follows. Place $w_{ij}\mathbf{v}_{ik}$ in the first location and \mathbf{w}_{ij} in location (i, k) . Fill up all other locations with $\mathbf{0}$. The dimensions of vectors placed in specific locations should be clear from the context. Note that the rows in M corresponding to the l -th literal x_{il} of function g_i are exactly those \mathbf{m}_{ijk} where j is the index of a row in G_i associated with x_{il} .

Let $A \subset \{x_{11}, \dots, x_{1m_1}, \dots, x_{n1}, \dots, x_{nm_n}\}$ be a collection of literals, and assume that the span of the rows in M associated with the literals in A contains $(10 \dots 0)$. For fixed (i, k) define \mathbf{z}_{ik} as the contribution (in the linear combination that yields $(10 \dots 0)$) of the rows \mathbf{m}_{ijk} associated with the $x_{il} \in A$. Then \mathbf{z}_{ik} is of the form $(w_i\mathbf{v}_{ik}, \mathbf{0}, \dots, \mathbf{w}_i, \mathbf{0}, \dots)$ where \mathbf{w}_i is in location (i, k) , $(w_i, \mathbf{w}_i) \in \text{Im } G_i^t$ and \mathbf{v}_{ik} is a multiple of the k -th row of F_i . By the choice of the locations we must have $\mathbf{w}_i = \mathbf{0}$, since the sum over all \mathbf{z}_{ik} is equal to $(10 \dots 0)$ and the other contributions have $\mathbf{0}$ in location (i, k) . Note that if $\mathbf{z}_{ik} \neq \mathbf{0}$, we must have $g_i(A) = 1$, since in this case the span of the rows in G_i corresponding to the $x_{il} \in A$ contains $(w_i, 0 \dots, 0)$ and $w_i \neq 0$.

For each i define \mathbf{z}_i as the sum over the contributing \mathbf{z}_{ik} 's. Then \mathbf{z}_i is of the form $(\mathbf{v}_i, \mathbf{0}, \dots)$ where $\mathbf{v}_i \in \text{Im } F_i$. Note that the sum over the \mathbf{z}_i 's is $(10 \dots 0)$. Let B denote the collection of i such that $\mathbf{z}_i \neq \mathbf{0}$. For $i \in B$, there must exist a

contributing $\mathbf{z}_{ik} \neq \mathbf{0}$. Then clearly we must have $f(B) = 1$. Similarly we conclude that $g_i(A) = 1$ when $i \in B$, by the remarks above. Hence (K, M, \mathbf{a}) computes h .

As to the multiplication property, keep in mind the workings of the secret sharing scheme sketched at the beginning of this proof. Let $(\mathbf{t}_1, \dots, \mathbf{t}_n)$ and $(\mathbf{t}'_1, \dots, \mathbf{t}'_n)$ denote $M\mathbf{b}$ and $M\mathbf{b}'$, respectively, for arbitrary \mathbf{b} and \mathbf{b}' . Denote the first coordinates of \mathbf{b} and \mathbf{b}' by s and s' , respectively. Here, \mathbf{t}_i and \mathbf{t}'_i stand for the union of the shares (in s and s' respectively) “given” to the literals x_{il} for $l = 1 \dots m_i$. For a proper fixed ordering of the coordinates of \mathbf{t}_i and \mathbf{t}'_i , we have

$$\langle \mathbf{r}_0, (\langle \mathbf{r}'_1, \mathbf{t}_1 * \mathbf{t}'_1 \rangle, \dots, \langle \mathbf{r}'_n, \mathbf{t}_n * \mathbf{t}'_n \rangle) \rangle = ss',$$

where for $i = 1 \dots n$, \mathbf{r}'_i denotes the repetition of \mathbf{r}_i rows(F_i) times. The recombination vector \mathbf{r} is now easily obtained from this expression.

Proof of Proposition 3 (second part). We have show that any $Q2$ -monotone function can be computed by a formula with three-input majority gates ($f_{2,3}$ -gates) and OR-gates ($f_{1,2}$ -gates).

We first describe an induction step that works for any monotone f . Let \mathcal{M} denote any collection of f 's qualified sets such that \mathcal{M} 's monotone closure coincides with the collection of all qualified sets of f (in particular, \mathcal{M} can consist of all minimal qualified sets). Define three smaller subsets \mathcal{M}_i of \mathcal{M} such that each member of \mathcal{M} occurs in two of the \mathcal{M}_i 's. For $i = 1, 2, 3$, let f_i be the monotone function corresponding to the monotone closure of \mathcal{M}_i . Define $\Phi = f_{2,3}(f_1, f_2, f_3)$. We have $\Phi = f$. Indeed, by construction $f(A) = 1$ implies $\Phi(A) = 1$. If $f(A) = 0$, then clearly $f_i(A) = 0$, $i = 1, 2, 3$. So $\Phi^*(A^c) = f_{2,3}^*(f_1^*(A^c), f_2^*(A^c), f_3^*(A^c)) = f_{2,3}(1, 1, 1) = 1$. Apply this inductively. Note that if any of the $f_{2,3}$ -gates thus constructed is satisfied, the complete formula is satisfied. Eventually, there will be \mathcal{M}_i 's with one or two members and the induction can no longer be applied.

Now we will apply the above to a $Q2$ -monotone function f (if $f \equiv x_i$ we are done). Let \mathcal{M} consist of all minimal qualified sets of f and all complements of maximal non-qualified sets of f and carry out the above recursion. At any point in the recursion where we have $(\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3)$ with, say wlog, \mathcal{M}_1 having one or two members, do the following. If \mathcal{M}_1 has one member, say A , let $i \in A$ and put $f_{2,3}(x_i, \cdot, \cdot)$. Else, say it contains B, B' , if neither B nor B' is the complement of a max-nonterm of f , let $i \in B$ and $j \in B'$, and put $f_{2,3}(f_{1,2}(x_i, x_j), \cdot, \cdot)$. In the other cases, by Lemma 1, there is an i with $i \in B \cap B'$. Then put $f_{2,3} = (x_i, \cdot, \cdot)$. Now we have a formula Φ consisting of $f_{2,3}$ - and $f_{1,2}$ -gates, where clearly for all C , $f(C) = 1$ implies $\Phi(C) = 1$. But does $f(A) = 0$ imply $\Phi(A) = 0$? We may assume that A is maximal. By the way we treated inputs of the form A^c with $f(A) = 0$, the values produced by the $f_{1,2}$ -gates “do not influence” the value of $\Phi(A^c)$. And since all other gates are self-dual, we have $\Phi^*(A^c) = \Phi(A^c) = 1$. Hence, $\Phi(A) = 0$. Note that if f is self-dual, the $f_{2,3}$ -gates suffice and no $f_{1,2}$ -gates are needed. In this case our proof is similar to a method given in [1].

Proof of Theorem 2. Similar argument as in the proof of Proposition 1 yields the first part of the first claim: let disjoint, non-qualified sets A, B, C contradict the claim, and let κ (with first coordinate 1) satisfy $M_B \kappa = \mathbf{0}$. Let \mathbf{r} be the recombination vector for the complement of C (the coordinates corr. to C are 0). Then $\langle (M\kappa * \mathbf{r})_A, M_A \mathbf{b} \rangle = s$ for all \mathbf{b} , contradicting $f(A) = 0$ as before. Thus f is Q3.

Next, let f be Q3 and let Φ be a formula with 1/3-accepting gates computing f . Denote by $\hat{\Phi}$ the formula obtained from Φ by replacing each gate f_{t_k, n_k} by f_{2t_k-1, n_k} . From the proof of Lemma 2, one easily sees that a single 1/3-accepting gate $f_{t, n}$ has a monotone span program with strong multiplication, and that by extending the rows one can easily construct a monotone span program for $f_{2t-1, n}$. Let (K, M^Φ, \mathbf{a}) and $(K, M^{\hat{\Phi}}, \mathbf{a})$, wlog $\mathbf{a} = (10 \cdots 0)$, be the respective monotone span programs by applying Proposition 2.

By induction one can show that for all \mathbf{b}, \mathbf{b}' there exists \mathbf{c} such that $M^\Phi \mathbf{b} * M^\Phi \mathbf{b}' = M^{\hat{\Phi}} \mathbf{c}$ and $ss' = t$, where s, s' and t' are the respective secrets. Now let $\Phi(A) = 0$. Then $\Phi^*(A^c) = 1$. But in each gate of Φ^* , we have $n_k - t_k + 1 \geq 2t_k - 1$ since $3t_k \leq n + 2$. Hence $\hat{\Phi}(A^c) = 1$ and a linear combination of the coordinates of $M_{A^c}^\Phi \mathbf{b} * M_{A^c}^\Phi \mathbf{b}'$ yields ss' .

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 $f(A) = 0$, and let M_A denote the rows of M corresponding to the players $i \in A$. 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 $\boldsymbol{\mu} = (\mu_1, \dots, \mu_e) \in \text{Ker} M_A$ with $\mu_1 = 1$ (see Section 4.2). Let $\boldsymbol{\mu} \otimes \boldsymbol{\mu}$ denote the e by e matrix whose i -th column is $\mu_i \boldsymbol{\mu}$. Note that it has 1 in its upper left corner and that it is symmetric. 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 $\mathbf{a} \notin \text{Im} M_D^t$, 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 Q3, 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 reconstruct¹⁶ the secret s (at least those corresponding to the players in A^c).

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

Proof of Theorem 4. By assumption the set A of corrupted players is non-qualified. The theorem is clearly true in case the protocol ends with the players using the default set of shares. Otherwise, only some non-qualified set B of players have complained. This means that the set C of non-complaining uncorrupted players must be qualified since A , B and C together form the complete player set (f is Q3).

Consider the matrix H whose (i, j) -entry is what the dealer claims is the correct check_{ij} (state of affairs as in Step 5). The correct players agree on the entries in those rows and columns of H (note that H is “matrix-valued”) that correspond to them, either because they agreed from the start, or because the dealer was forced to make the values public.

Consider an arbitrary player $j \in C$. From the dealer he has received U_j , and since $j \in C$, he has been able to verify that the j -th column of H is of form $(M_1 U_j^t, \dots, M_n U_j^t)$, and that the j -th row of H is of form $(U_j M_1^t, \dots, U_j M_n^t)$.

Hence, for each $j \in C$, the j -th column of H forms consistent sets of shares, where the secrets are the elements in the first column of U_j (call these \mathbf{s}_j). And those entries in the j -th column of H corresponding to the players in C are sufficient for reconstructing those secrets. Thus, there exists a vector $\boldsymbol{\lambda}_C$ (only depending on C) such that for each $j \in C$, $\langle \boldsymbol{\lambda}_C, M_C U_j^t \rangle = U_j M_C^t \boldsymbol{\lambda}_C = \mathbf{s}_j \in K^{d_j}$, where M_C denotes the superposition of those rows j of M with $j \in C$ (note that $M_C^t \boldsymbol{\lambda} \in K^e$).

Now observe that for each $j \in C$ we have $\mathbf{s}_j = U_j M_C^t \boldsymbol{\lambda}_C = M_j U_C^t \boldsymbol{\lambda}_C$, where U_C is the superposition of the U_j with $j \in C$. Hence, the \mathbf{s}_j thus computed form a consistent set of shares for the players in C in the secret sharing scheme. To complete these to a full set of shares, note that for $j \notin C$ we can define $\mathbf{s}_j = M_j U_C^t \boldsymbol{\lambda}_C$ as well (which can be computed from the entries (i, j) in H with $i \in C$ and $j \notin C$). Hence

¹⁶It is easy to show that the scheme from Section 4.2 already has the right error correction property if f is Q3. But there, the cost of recovery may be superpolynomial in the size of the span program (as opposed to our modified scheme here).

$(\mathbf{s}_1, \dots, \mathbf{s}_n)$ is a collection of consistent shares in the secret sharing scheme.

Note also that an uncorrupted player k not in C will be able to determine the correct share from the entries in his column: he agrees with the players in C on the entries in the columns corresponding to C and the rest of the entries must be consistent with this, either because the dealer made the entire column public without causing more complaints, or because some selected values were complained about and corrected by the dealer to player k 's satisfaction.

Recent BRICS Report Series Publications

- RS-97-28 Ronald Cramer, Ivan B. Damgård, and Ueli Maurer. *Span Programs and General Secure Multi-Party Computation*. November 1997. 27 pp.
- RS-97-27 Ronald Cramer and Ivan B. Damgård. *Zero-Knowledge Proofs for Finite Field Arithmetic or: Can Zero-Knowledge be for Free?* November 1997. 33 pp.
- RS-97-26 Luca Aceto and Anna Ingólfssdóttir. *A Characterization of Finitary Bisimulation*. October 1997. 9 pp. To appear in *Information Processing Letters*.
- RS-97-25 David A. Mix Barrington, Chi-Jen Lu, Peter Bro Miltersen, and Sven Skyum. *Searching Constant Width Mazes Captures the AC^0 Hierarchy*. September 1997. 20 pp.
- RS-97-24 Søren B. Lassen. *Relational Reasoning about Contexts*. September 1997. 45 pp. To appear as a chapter in the book *Higher Order Operational Techniques in Semantics*, eds. Andrew D. Gordon and Andrew M. Pitts, Cambridge University Press.
- RS-97-23 Ulrich Kohlenbach. *On the Arithmetical Content of Restricted Forms of Comprehension, Choice and General Uniform Boundedness*. August 1997. 35 pp.
- RS-97-22 Carsten Butz. *Syntax and Semantics of the logic $\mathcal{L}_{\omega\omega}^\lambda$* . July 1997. 14 pp.
- RS-97-21 Steve Awodey and Carsten Butz. *Topological Completeness for Higher-Order Logic*. July 1997. 19 pp.
- RS-97-20 Carsten Butz and Peter T. Johnstone. *Classifying Toposes for First Order Theories*. July 1997. 34 pp.
- RS-97-19 Andrew D. Gordon, Paul D. Hankin, and Søren B. Lassen. *Compilation and Equivalence of Imperative Objects*. July 1997. iv+64 pp. Appears also as Technical Report 429, University of Cambridge Computer Laboratory, June 1997. To appear in *Foundations of Software Technology and Theoretical Computer Science: 17th Conference, FCT&TCS '97 Proceedings, LNCS, 1997*.