Secure Multi-Party Computation Made Simple*

Ueli Maurer**

Department of Computer Science
ETH Zurich
CH-8092 Zurich, Switzerland,
maurer@inf.ethz.ch

Abstract. Known secure multi-party computation protocols are quite complex, involving non-trivial mathematical structures and sub-protocols. The purpose of this paper is to present a very simple approach to secure multi-party computation with straight-forward security proofs. This approach naturally yields protocols secure for mixed (active and passive) corruption and general (as opposed to threshold) adversary structures, confirming the previously proved tight bounds in a simpler framework. Due to their simplicity, the described protocols are well-suited for didactic purposes, which is a main goal of this paper.

1 Introduction

We propose a new, very simple approach to multi-party computation (MPC) secure against active cheating and, more generally, mixed corruption scenarios. This work is motivated by a protocol of Beaver and Wool [BW98] which achieves security only for a passive adversary setting, without the possibility to enhance it to active adversary settings.

In this section we review the definition of secure MPC, discuss various models for specifying the adversary's capabilities, and review different types of security and communication models. A reader familiar with these topics can skip Section 1 and much of Section 4, where previous results are reviewed. After discussing secret-sharing and other preliminaries in Section 2, our model and results are stated in Section 3. The main parts of the paper are Section 5, where the passively secure protocol and the underlying secret-sharing scheme is presented, and Section 6 which presents the protocol secure in the general corruption model.

1.1 Secure Multi-party Computation

Secure function evaluation, as introduced by Yao [Yao82], allows a set $P = \{p_1, \ldots, p_n\}$ of n players to compute an arbitrary agreed function of their private inputs $x_1, \ldots, x_n$, respectively, even if an adversary may corrupt and control some of the players in various ways, to be discussed below. More generally, secure MPC allows the players to perform an arbitrary on-going computation during which new inputs can be provided and players can interact with an environment. This corresponds to the simulation of a trusted party [GMW87,HM97].

Security in MPC means that the players' inputs remain secret (except for what is revealed by the intended results of the computation) and that the results of the computation are guaranteed to be correct. More precisely, security is defined relative to an ideal-world specification involving a trusted party: anything the adversary can achieve in the real world (where the protocol is executed) he can also achieve in the ideal world [Can00,PSW00].

* The results of this paper were first presented at the Cryptography Workshop in Luminy in Sept. 1999, and later appeared in [Mau02].

** Supported in part by the Swiss National Science Foundation.

Many distributed cryptographic protocols can be seen as special cases of secure MPC. For specific tasks like collective contract signing, on-line auctions, or voting, there exist very efficient protocols. Throughout this paper we consider *general* secure MPC protocols, where general means that any given specification involving a trusted party can be computed securely without the trusted party. In other words, we consider compilers that take as input a specification and generate a secure protocol for realizing the specification.

Most protocols for general secure MPC work roughly as follows: The function (or specification) to be computed is specified by a circuit over some finite field consisting of addition and multiplication gates. This is no essential restriction. Each input value and each intermediate result is shared appropriately among the players so that no cheating player set can learn anything. The circuit is evaluated gate by gate, performing a sub-protocol for each gate. The result(s) of the computation are jointly reconstructed.

General MPC protocols tend to be less efficient than special-purpose protocols, for two reasons. First, the circuit can generally be quite large. Second, the multiplication sub-protocol is rather inefficient as it requires substantial interaction (but see [HM01] for efficiency improvements for general MPC protocols).

1.2 Specifying the Adversary's Capabilities

The potential misbehavior of some of the players is usually modeled by considering a central adversary with an overall cheating strategy who can corrupt some of the players. Two different notions of corruption, passive and active corruption, are usually considered. Passive corruption means that the adversary learns the entire internal information of the corrupted player, but the player continues to perform the protocol correctly. Such players are sometimes also called semi-honest. Active corruption means that the adversary can take full control of the corrupted player and can make him deviate arbitrarily from the protocol. If no active corruptions are considered, then the only security issue is the secrecy of the players' inputs.

A non-adaptive or static adversary must decide before the execution of the protocol which players he corrupts, while an adaptive adversary can corrupt new players during the protocol, as long as the total set of corrupted players is still admissible. A mobile adversary can release some of the corrupted players, thereby regaining corruption power. We consider adaptive, but not mobile adversaries.

In many papers, the adversary's corruption capability is specified by a threshold t, i.e., the adversary is assumed to be able to corrupt up to t (but not more) players. More generally, the adversary's corruption capability could be specified by a so-called adversary structure, i.e., a set of potentially corruptible subsets of players. Even more generally, the corruption capability can be specified by a set of corruption scenarios, one of which the adversary can choose (secretly). For instance, each scenario can specify a set of players that can be passively corrupted and a subset of them that can even be actively corrupted. In Section 4 we describe these models and the results known for them.

1.3 Types of Security and Communication Models

One distinguishes between two types of security. Information-theoretic security means that even an adversary with unrestricted computing power cannot cheat or violate secrecy, while cryptographic security relies on an assumed restriction on the adversary's computing power and on certain unproven assumptions about the hardness of some computational problem, like factoring large integers. The terms "perfect" and "unconditional" security are often used for information-theoretic security with zero and negligible error probability, respectively. In this paper we consider perfect information-theoretic security, i.e., the probability of successful cheating is zero and the information leaked to the adversary is also zero.

Several communication models are considered in the literature. In the standard synchronous model (for information-theoretic security), any pair of players can communicate over a bilateral secure channel. Some papers [RB89,Bea91,CDD$^+$99] assume the availability of a broadcast channel which guarantees the consistency of the received values if a sender sends a value to several players, but in practice a broadcast channel must be simulated by a (quite inefficient) protocol among the players (e.g. [LSP82,BGP89,FM97]). In asynchronous communication models, no guarantees about the arrival times of sent messages are assumed. Here we do not consider asynchronous communication models, although our techniques may also be applied in that context.

2 Preliminaries

2.1 Structures

Definition 1. Consider a finite set P. We call a subset Π of the power set 2^P of P a *(monotone) structure* for P if Π is closed under taking subsets, i.e., if $S \in \Pi$ and $S' \subseteq S$ imply $S' \in \Pi$. We define a (commutative and associative) operation on structures, denoted $\sqcup$: $\Pi_1 \sqcup \Pi_2$ is the structure consisting of all unions of one element of Π_1 and one element of Π_2, i.e.,

$$\Pi_1 \sqcup \Pi_2 := \{S_1 \cup S_2 : \; S_1 \in \Pi_1, S_2 \in \Pi_2\}.$$

Structures will be described by listing only the maximal sets, their subsets being understood as belonging to the structure. The *size* $|\Pi|$ of a structure Π is the number of maximal elements.

Example 1. The most common example of a structure is the threshold structure $\Pi = \{S : S \subseteq P, |S| \leq t\}$ for some t. Note that the description that lists the maximal sets has size $\binom{n}{t}$ which is exponential in n if t is a fixed fraction of n.

2.2 Secret-Sharing and Secrecy Structures

A *secret-sharing scheme* allows a dealer to share a value among a set $P = \{p_1, \ldots, p_n\}$ of players such that only certain qualified subsets of players can reconstruct the secret, i.e., are *qualified*, while certain other subsets of players obtain no information about the secret, i.e., are *ignorant* (a term not used in the previous literature). Natural secret-sharing schemes (and also those of this paper) have the property that every subset of P is either qualified or ignorant, and this is why ignorant sets are usually called *non-qualified.* However, for reasons explained below, we choose the new terminology.

The secrecy condition is actually stronger: Even if any ignorant player set holds any kind of partial information about the shared value, they must not obtain any *additional* information about the shared value. Stated differently, what an ignorant set receives is statistically independent of the information they hold and the shared value. Equivalently, such an ignorant set can simulate their shares with the same probability distribution as that occurring in the actual protocol.

A secret-sharing scheme is usually specified by the so-called *access structure*[1] Γ, the collection of qualified player subsets. In our context, it is more natural to characterize a secret-sharing scheme by the *secrecy structure* Σ consisting of the collection of ignorant player subsets. As mentioned above, the secrecy structure is typically the complement of the access structure, i.e., $\Sigma = 2^P \backslash \Gamma$.

Why is it more natural to consider the secrecy structure instead of the access structure, and why is the term ignorant more natural than non-qualified? When the potential misbehavior of players is considered, one leaves the realm of classical secret-sharing. For instance, players could misbehave by not sending their share when supposed to, or by even sending a false share. In such a case, a qualified set can generally not reconstruct the secret, i.e, the notion of being qualified loses its normal meaning. In contrast, the notion of secrecy is not changed by misbehaving players. If a secret is shared according to a certain scheme, then the secrecy structure remains unchanged, even if players misbehave (except, of course, restricting the secrecy structure to sets containing the corrupted players).

2.3 Adversary Structures and Security Against Active Cheating

As mentioned earlier, as a generalization of specifying the adversary's capabilities by a corruption type and a threshold t, one can describe it by a corruption type and an *adversary structure* Δ meaning that the adversary can choose one of the sets in Δ and corrupt these players [HM97]. For passive corruption we can also call this structure the secrecy structure rather than the adversary structure.

When players can cheat actively, then even the consistency of the value sent by a player to several other players is not guaranteed. In other words, one must use a so-called *broadcast* protocol (e.g. [LSP82,BGP89,FM97]) to assure that all honest players receive the same value, and that if the sender is honest, then the received value is that actually sent by the sender. A classical result [LSP82] in the theory of distributed systems is that such a protocol exists if and only if less than a third of the players cheat.

More generally, a cheating dealer in a secret-sharing scheme could distribute inconsistent shares, resulting in a situation where no value can be reconstructed. The consistency of the shared values must again be guaranteed by a special protocol, called *verifiable secret-sharing (VSS)*.

3 Models and Results of this Paper

We present a very simple approach to secure multi-party computation. Unlike previous approaches, it is based on essentially no mathematical structure (like bivariate polynomials or zero-knowledge proofs), and it naturally yields protocols secure against general mixed adversary structures.

The main focus of the paper is on simplicity of the protocols, which makes them suitable for didactic purposes. However, it is quite possible that the protocol ideas have applications in other contexts and that for certain applications, especially when involving only a small number of players, the protocols are the most efficient known.

[1] But note that, according to our terminology, it is actually an anti-structure, where we call Π a *(monotone) anti-structure* if it is closed under taking supersets, i.e., if the complement $\Pi^c := \{S \in 2^{\mathcal{P}} : S \notin \Pi\}$ is a structure.

The adversary is specified by a secrecy structure Σ *and* an adversary structure $\Delta \subseteq \Sigma$, with the following meaning.

Definition 2. Consider a player set P and two structures $\Sigma \subseteq 2^P$ and $\Delta \subseteq \Sigma$. A (Σ, Δ)-*adversary* is an adversary who can (adaptively) corrupt some players passively and some players actively, as long as the set A of actively corrupted players and the set B of passively corrupted players satisfy both

$$A \in \Delta \qquad \text{and} \qquad (A \cup B) \in \Sigma.$$

In other words, a cheating player set A cannot violate the correctness, and all corrupted players together (the set $A \cup B$) obtain no information not specified by the protocol. This model is the same as that of [FM02] where only verifiable secret-sharing is considered.

The following theorems give increasingly strong conditions for broadcast, for verifiable secret-sharing, and for secure MPC to be possible. The efficiency of the protocols is polynomial in n, $|\Sigma|$, and $|\Delta|$, but this fact is not stated explicitly.

Theorem 1. *The simulation of a broadcast channel secure against a (Σ, Δ)-adversary is possible if and only if $P \notin \Delta \sqcup \Delta \sqcup \Delta$.*

Theorem 2. *Perfect verifiable secret-sharing secure against a (Σ, Δ)-adversary is possible if and only if $P \notin \Sigma \sqcup \Delta \sqcup \Delta$.*

Theorem 3. *General perfect information-theoretically secure MPC secure against a (Σ, Δ)-adversary is possible if and only if $P \notin \Sigma \sqcup \Sigma \sqcup \Delta$.*

Theorem 1 follows from a more general result in [HM97] and the efficient broadcast protocol given in [FM98]. This theorem is used, but not considered further in this paper. Theorem 3 is equivalent to Theorem 1 of [FHM99], as will be explained in Section 4.4.

As stated in these theorems, all these results are known to be tight in the sense that larger adversary structures cannot be tolerated. We do not discuss such impossibility proofs here. They work by proving the impossibility for a small player set ($n = 2$ or $n = 3$) and showing that any protocol violating the stated bounds could be transformed into an impossible protocol for a small player set. For example, broadcast among three players with one cheaters can be proved to be impossible. This implies the necessity of the $P \notin \Delta \sqcup \Delta \sqcup \Delta$ condition. Similarly, the secure computation of the OR function of two input bits held by two players ($n = 2$) is impossible, even if both players are only passive cheaters. This implies the necessity of the condition $P \notin \Sigma \sqcup \Sigma$.

4 Review of Results on General Secure Multi-Party Computation

In this section we review the previous results on necessary and sufficient conditions for general secure MPC to be possible, for various models and degrees of generality.

4.1 Classical Threshold Results

In the original papers solving the general secure MPC problem, the adversary is specified by a single corruption type (active or passive) and a threshold t on the tolerated number of corrupted players. Goldreich, Micali, and Wigderson [GMW87] proved that, based

setting	adversary type	condition	reference
cryptographic	passive	$t < n$	[GMW87]
cryptographic	active	$t < n/2$	[GMW87]
information-theoretic	passive	$t < n/2$	[BGW88],[CCD88]
information-theoretic	active	$t < n/3$	[BGW88],[CCD88]
i.t., with broadcast	active	$t < n/2$	[RB89,Bea91]

Table 1. Necessary and sufficient threshold conditions for general secure MPC to be possible.

on cryptographic intractability assumptions, general secure MPC is possible if and only if $t < n/2$ players are actively corrupted. The threshold for passive corruption is $t < n$. In the information-theoretic model, where bilateral secure channels between every pair of players are assumed, Ben-Or, Goldwasser, and Wigderson [BGW88] proved that perfect security is possible if and only if $t < n/3$ for active corruption, and if and only if $t < n/2$ for passive corruption.[2] In a model with a physical broadcast channel, which helps only in case of active corruption, unconditional security is achievable if and only if $t < n/2$ [RB89,Bea91,CDD$^+$99]. These classical results are summarized in Table 1.

4.2 Mixed Adversary Models

The exact threshold conditions for mixed models under which secure MPC is possible were proved in [FHM98], including fail-corruption as a third corruption type. Here we state the results without considering fail-corruption. Let t_a and t_p be the number of players that can be actively and passively corrupted, respectively. Perfect security is achievable if and only if $3t_a + 2t_p < n$, whether or not a broadcast channel is available. This is a special case of Theorem 3.[3]

4.3 General Adversary Structures

The threshold adversary models were extended to a non-threshold setting in [HM97] (see also [HM00]), for either passive or active, but not for mixed corruption. The adversary's capability is characterized by a structure, called secrecy structure Σ for passive corruption and adversary structure Δ for active corruption. Again, generalizing the model leads to strictly stronger results compared to those achievable in the threshold model. For instance, in the case of 6 players and active corruption, with $P = \{A, B, C, D, E, F\}$, one can obtain a protocol secure against the structure with $\Delta = \{\{A\}, \{B, D\}, \{B, E, F\}, \{C, E\}, \{C, F\}, \{D, E, F\}\}$, whereas in the threshold model one can tolerate only a single active cheater, i.e., the adversary structure $\Delta = \{\{A\}, \{B\}, \{C\}, \{D\}, \{E\}, \{F\}\}$.

Let $Q^2(\Pi)$ be the condition on a structure Π that no two sets in Π cover the full player set P, i.e.,

$$Q^2(\Pi) \iff P \notin \Pi \sqcup \Pi.$$

[2] The same result was obtained independently by Chaum, Crépeau, and Damgård [CCD88], but with an exponentially small error probability.

[3] Exponentially small error probability with a broadcast channel is achievable if and only if $2t_a + 2t_p < n$. Without broadcast, the additional condition $3t_a < n$ is necessary and sufficient. This strictly improves on non-mixed threshold results: In addition to tolerating $t_a < n/3$ actively corrupted players, secrecy can be guaranteed against every minority, thus tolerating additional $t_p \leq n/6$ passively corrupted players.

Similarly, let $Q^3(\Pi)$ be the condition that no three sets in Π cover the full player set P, i.e.,

$$Q^3(\Pi) \Longleftrightarrow P \notin \Pi \sqcup \Pi \sqcup \Pi.$$

The main results of [HM97] state that for passive corruption, $Q^2(\Sigma)$ is the necessary and sufficient condition for general secure MPC to be possible. For active corruption, the condition is $Q^3(\Delta)$, and if a broadcast channel is available, then the condition is $Q^2(\Delta)$. The first two results are again special cases of Theorem 3. These results were achieved by a recursive player substitution technique, yielding quite complex (but polynomial in the size of Δ) protocols. The protocols of this paper are much simpler, more intuitive, and considerably more efficient.

4.4 Mixed General Adversary Structures

Finally, general mixed adversary specifications were considered in [FHM99] and the exact conditions for general secure MPC to be possible were given for a general mixed passive/active model. For each admissible choice, the adversary can actively corrupt a subset $D \subseteq P$ of the players, and, additionally, can passively corrupt another subset $E \subseteq P$ of the players. The adversary specification Ψ is hence a set of pairs (D, E), i.e.,

$$\Psi = \{(D_1, E_1), \ldots, (D_k, E_k)\},$$

for some k, and the adversary may select one arbitrary pair (D_i, E_i) from Ψ and corrupt the players in D_i actively and, additionally, corrupt the players in E_i passively. The adversary's choice is not known before and typically also not after execution of the protocol. It was proved in [FHM99] that, with or without broadcast channels, perfect general MPC is achievable if and only if the adversary specification Ψ satisfies the following condition $Q^{(3,2)}(\Psi)$:

$$Q^{(3,2)}(\Psi) \Leftrightarrow \forall (D_1, E_1), (D_2, E_2), (D_3, E_3) \in \Psi : D_1 \cup E_1 \cup D_2 \cup E_2 \cup D_3 \neq P .$$

At first sight, these results look more general than Theorem 3 since the adversary specification consists of a general set of pairs rather than two structures. However, they are equivalent, which can be seen as follows. For an adversary specification $\Psi = \{(D_1, E_1), \ldots, (D_k, E_k)\}$ we can define naturally an associated secrecy structure

$$\Sigma(\Psi) = \{D \cup E : (D, E) \in \Psi\}$$

and an associated adversary structure

$$\Delta(\Psi) = \{D : (D, E) \in \Psi \text{ for some } E\}.$$

Now we can define the closure $\overline{\Psi}$ of Ψ as

$$\overline{\Psi} := \{(D, E) : D \in \Delta(\Psi) \wedge (D \cup E) \in \Sigma(\Psi)\}.$$

It is not difficult to show that $Q^{(3,2)}(\Psi) \Leftrightarrow Q^{(3,2)}(\overline{\Psi})$. Therefore secure MPC is possible for a given adversary specification Ψ if and only if it is possible for $\overline{\Psi}$. In other words, one can enlarge any specification Ψ to its closure $\overline{\Psi}$ for free.[4] This justifies the consideration of (Σ, Δ)-adversaries as discussed above. To see this, take any (D_i, E_i) and (D_j, E_j), add the new pair $(D_i, (D_j \cup E_j) \backslash D_i)$ to Ψ, and check that the condition $Q^{(3,2)}(\Psi)$ is still satisfied.

[4] However, there may exist protocols secure for Ψ but not for $\overline{\Psi}$. But in such a case there would exist a different protocol secure for $\overline{\Psi}$, with possibly (much) higher complexity.

5 Secure MPC: The Passive Case

5.1 The Format of the Protocol

The computation to be performed is specified by a circuit over some finite field consisting of addition and multiplication gates, whose inputs are the players inputs into the computation.[5] Each input value and each intermediate result is shared among the players, according to the secrecy structure, using a linear secret-sharing scheme.

Due to the linearity, secure addition and more generally computing any linear function of shared values is trivial: every player locally computes the linear function of his shares and keeps the result as a share of the new value. Secrecy is trivially guaranteed because this step involves no communication. Correctness is also trivially guaranteed because due to the lack of communication there is no chance for a corrupted player to cheat. Hence the only remaining problem is the secure multiplication of shared values.

5.2 The Secret-Sharing Scheme

As a building block, we need a k-out-of-k secret-sharing scheme, i.e., one for k players such that only the complete set of players (but no proper subset) can reconstruct the secret. Such a scheme (actually linear) for any k and any domain $\mathcal{D}$ of the secret s is obtained by splitting s into a random sum.[6]

> **k-out-of-k secret-sharing:**
>
> Select $k-1$ shares $s_1, \ldots, s_{k-1}$ at random from $\mathcal{D}$ and let $s_k := s - \sum_{i=1}^{k-1} s_i$.
> The ith share is s_i.

Lemma 1. *The above scheme is a k-out-of-k secret-sharing scheme.*

Proof. All shares together obviously determine the secret, hence the set of all k players is qualified. Any set of $k-1$ players (with, say, p_i missing) is ignorant because these $k-1$ shares $(s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_k)$ are independent and uniformly random, independently of s. This follows from the fact that for any fixed s and any fixed (missing) share s_i, the mapping from $(s_1, \ldots, s_{k-1})$ to $(s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_k)$ is one-to-one. The shares can be simulated by generating a set of uniform and independent shares. ◇

The most natural approach to designing a secret-sharing scheme for a given access structure Γ (or the secrecy structure $\Sigma = 2^P \setminus \Gamma$) is due to Itoh et al. [ISN87] who introduced general access structures in secret-sharing. In this scheme, the secret is shared, independently, to each minimal qualified player set $S \in \Gamma$, with an $|S|$-out-of-$|S|$ secret-sharing scheme. This trivially guarantees that any qualified set can reconstruct the secret and that no ignorant set $S \notin \Gamma$ gets any information about the secret.

In this paper we use a different, in a sense dual approach. Let k be the number of maximal sets in Σ, i.e. $\Sigma = \{T_1, \ldots, T_k\}$,[7] and let $\overline{T_i} := P \setminus T_i$ be the complement of the set T_i.

[5] More generally, the circuit could contain any gates for linear functions, plus (non-linear) multiplication gates.
[6] It is trivial to impose an addition operation on $\mathcal{D}$ which makes it into an additive group, for instance the group isomorphic to the cyclic group $Z_{|\mathcal{D}|}$.
[7] Recall that a structure is specified by the maximal sets.

Secret-sharing for secrecy structure $\Sigma = \{T_1, \ldots, T_k\}$:

1. Split the secret using the k-out-of-k secret-sharing scheme, resulting in shares $s_1, \ldots, s_k$.
2. Send s_i (secretly) to each player in $\overline{T_i}$.
   (The share of player p_m is hence the set $\{s_i : m \in \overline{T_i}\}$.)

Lemma 2. *The above scheme is a secret-sharing scheme for secrecy structure $\Sigma = \{T_1, \ldots, T_k\}$.*

Proof. The scheme is trivially Σ-secure because for any set $T \in \Sigma$, at least one share (namely that given to the complement of a maximal set of Σ containing T) is missing. Hence, according to Lemma 1, the set T has no information about the secret. Moreover, for any ignorant set S, the obtained information consists of some (but not all) shares s_i and k-out-of-k sharings thereof. This information is independent of anything else and could actually be simulated by S.

Reconstruction by any qualified set in $\Gamma = 2^P \backslash \Sigma$ is simple. Any set $S \in \Gamma$ contains, for every maximal set $T_i \in \Sigma$, a player not in T_i. This player knows s_i, and hence the players in S know all the shares s_i and are thus qualified. ◇

5.3 The Multiplication Protocol

As mentioned earlier, the condition $Q^2(\Sigma)$, which is equivalent to

$$P \notin \Sigma \sqcup \Sigma, \tag{1}$$

is necessary and sufficient for information-theoretically secure MPC for passive corruption. Condition (1) means that for any two maximal sets $T_1, T_2 \in \Sigma$ we have $T_1 \cup T_2 \neq P$, which is equivalent to the condition that for any $T_1, T_2 \in \Sigma$, their complements intersect, i.e., $(P \backslash T_1) \cap (P \backslash T_2) \neq \{\}$. A set of sets, no two of which are disjoint, is also called a *quorum system.* Condition (1) is thus equivalent to the statement that the sets $P \backslash T_i$ for $i = 1, \ldots, k$ form a quorum system.

The product of two shared values s and t can be computed as

$$st = \left(\sum_{i=1}^{k} s_i\right) \cdot \left(\sum_{j=1}^{k} t_j\right) = \sum_{i=1}^{k} \sum_{j=1}^{k} s_i t_j, \tag{2}$$

i.e., as the sum of k^2 share products. Therefore we can use the following observation by Beaver and Wool [BW98], used originally for a different secret-sharing scheme (see Section 5.4). For every term $s_i t_j$ in the above sum, there exists at least one player who knows both s_i and t_j. This player (or one of them) can compute the product $s_i t_j$ and share it among the players (using the basic secret-sharing scheme). Since st is a linear combination of these shared values $s_i t_j$, the sharing of st can be computed non-interactively. An efficiency improvement is obtained if each player first adds all terms assigned to him and then shares the sum. Note that terms of the form $s_i t_i$ (i.e., $i = j$) can be assigned to any player knowing the ith share. In summary, we have:

Multiplication protocol (passive):

Preparation (once and for all): Partition the set $\{(i,j) : 1 \le i,j \le n\}$ into n sets $U_1, \ldots, U_n$ such that for all $(i,j) \in U_m$ we have $m \in \overline{T_i} \cap \overline{T_j}$.[8]

Precondition: Two values $s = \sum_{i=1}^{k} s_i$ and $t = \sum_{i=1}^{k} t_i$ are shared.

Postcondition: st is shared independently.

1. Each player p_m (for $1 \le m \le n$) computes $v_m := \sum_{(i,j)\in U_m} s_i t_j$ and shares v_m among all players (using independent randomness).
2. Each player (locally) adds all n shares received in step 1.

Lemma 3. *In any given context where s and t are shared, the above protocol results in the product st being shared, and nothing else. More precisely, the new information obtained by any ignorant set is independent of any information held (by this set) prior to the execution of the protocol.*

Proof. The correctness of the new sharing of st follows from (2). According to Lemma 2, for any ignorant set, every sharing (by some player) of a value results in independent information. Hence this is true for the entire protocol. $\diamond$

5.4 Comparison with the Beaver-Wool Scheme

The secret-sharing scheme of [BW98] works as follows: The secret is split by a sum sharing into l shares, where $l = |\Gamma|$ is the number of minimal qualified sets. Then each share is given to the players in one of the minimal qualified sets. While this sharing looks similar to ours (in our scheme we consider the maximal non-qualified sets rather than the minimal qualified sets) it differs in a crucial way: Condition (1) is required not only for the multiplication protocol, but even for the mere reconstruction of shared secrets. A qualified set can reconstruct the secret because it overlaps with any other minimal qualified set and hence knows all the shares, i.e., one must start with an access structure (and corresponding secrecy structure) which satisfies (1) in the first place. This is the reason why the scheme of [BW98] cannot be enhanced to tolerate active corruption.

6 Secure MPC for a General (Σ, Δ)-Adversary

The basic structure of the protocol is as described in Section 5.1. Two changes are required to tolerate also active corruption: The secret-sharing scheme and the multiplication protocol must be made robust against cheating by a set of players in Δ (including possibly the dealer in case of the secret-sharing scheme). These two protocols are described in the following two subsections.

[8] Such a partition exists because of condition (1). Some of the U_m may be empty.

6.1 Verifiable Secret Sharing

As mentioned above, a secret-sharing scheme is useless if not all players can be assumed to behave correctly. A first problem is that some players may contribute false shares during reconstruction. This can be solved by distributing the secret in a redundant manner, allowing for error correction. Since this requires the set of shares to satisfy a certain consistency condition, a second problem arises, namely that a cheating dealer can distribute inconsistent shares. Verifiable secret-sharing solves both these problems.

Definition 3. A *verifiable secret-sharing* (VSS) scheme for a set P of players with secrecy structure Σ and secure for adversary structure Δ consists of two protocols, **Share** and **Reconstruct**, such that even if the adversary corrupts players according to Δ, the following conditions hold:

1. If **Share** terminates successfully, then the **Reconstruct** protocol yields the same fixed value for all possible adversary strategies, i.e., the dealer (even if corrupted) is committed to a single value.
2. If the dealer is honest during **Share**, then **Reconstruct** always yields his input value.
3. If the dealer is honest, then the information obtained by any ignorant set in Σ after the sharing phase is independent of any information held (by this set) prior to the execution of the protocol. The information obtained in the reconstruction phase is nothing beyond the reconstructed value. More precisely, given the reconstructed value, the information obtained by any ignorant set in the reconstruction phase is independent of any information held (by this set) prior to the execution of the protocol.

We show how the secret-sharing scheme of Section 5.2 for the secrecy structure Σ can be extended to a VSS scheme secure in presence of a (Σ, Δ)-adversary, provided that Σ and Δ satisfy the following condition:

$$P \notin \Sigma \sqcup \Delta \sqcup \Delta. \tag{3}$$

We first describe the VSS sharing protocol, which only depends on Σ but not on Δ, and then discuss condition (3) together with the reconstruction protocol.

To assure that the dealer correctly shares a value, we only need to guarantee, independently for each of the k shares, that all honest players receiving this share obtain the same value.[9] This is easily achieved as follows. For each share, say s_i, all players receiving that share (those in $\overline{T_i}$) check pairwise whether the value received from the dealer is the same. If any inconsistency is detected, the players detecting it complain using broadcast, and the dealer must broadcast s_i to all the players. Secrecy cannot be violated because a complaint is sent only if either the dealer is corrupted or a corrupted player received s_i, hence the adversary knew s_i already. After these checks it is guaranteed that all honest players knowing s_i hold the same value for s_i.

[9] Guaranteeing this condition independently for each of the k shares suffices because the k shares are completely general and need not satisfy a consistency condition like in schemes based on polynomials. Every set of k shares uniquely determines a secret.

VSS Share (Σ):

1. Share the secret s using the scheme of Section 5.2.[10]
2. For each share s_i: Each pair of players in $P\backslash\overline{T_i}$ check (over a secure channel) whether their received values for s_i agree.
   If any inconsistency is detected, the players complain, using (possibly simulated) broadcast.
3. The dealer broadcasts all shares for which complaints were raised, and the players accept these shares.
   If the dealer refuses any of these broadcasts, the protocol is aborted.

Condition (3) implies that for a given collection of values received during reconstruction of a share, say s_i, there is only one consistent explanation for which is the correct value of s_i, namely that value v for which the set of differing values corresponds to a set in Δ. For a given list of (partially false) values for s_i, if there were two possible values v' and v'' with corresponding sets A' and A'' in Δ, then the set $P\backslash(A' \cup A'')$ alone could not be qualified in the secret-sharing scheme, since otherwise the secret would be uniquely determined by those values, contradicting the assumption. But if $P\backslash(A' \cup A'')$ is not qualified, it is in Σ, and this contradicts condition (3). Hence the following protocol works.

VSS Reconstruct (Σ, Δ):

1. All players send all their shares (bilaterally) to all other players.[11]
2. Each player reconstructs (locally) each of the k shares $s_1, \ldots, s_k$ and adds them up to obtain the secret $s = s_1 + \cdots + s_k$.
   Reconstruction of share s_i (same for each player): Let v_j for $j \in \overline{T_i}$ be the value (for s_i) sent by player p_j. Take the (unique) value v such that there exists $A \in \Delta$ with $v_j = v$ for all $j \in \overline{T_i} - A$.

Note that the reconstruction is performed independently for each share s_i. This fact is used in the robust multiplication protocol discussed below.

The proof of the following lemma follows from the above discussion.

Lemma 4. *The above sharing and reconstruction protocols form a VSS protocol for secrecy structure Σ and adversary structure Δ, if $P \notin \Sigma \sqcup \Delta \sqcup \Delta$.*

Two shared values can be added by each player adding the corresponding shares. More generally we have:

Lemma 5. *Linear functions of values shared according to the VSS scheme can be computed by each player computing the linear function on the corresponding shares.*

Proof. Since the secret sharing scheme is linear, the resulting sharing is a correct sharing of the linear function of the shared values. Since the same consistency guarantees hold as after the VSS sharing phase, the reconstruction protocol also works. Secrecy cannot be violated in this protocol since it involves no communication. $\diamond$

[10] If a player does not receive a share because the dealer is corrupted, then he can take a default share, say 0.
[11] No broadcast is required.

6.2 Robust Multiplication Protocol

The approach of Section 5.3 fails if active cheating occurs because a single false term $s_i t_j$ can change the result arbitrarily. Hence we need a method for guaranteeing that a player correctly computes and shares such a term. This is achieved by assigning each term $s_i t_j$ to all players knowing both s_i and t_j, and having each of these players share the value by VSS. After each of these (say r) players has shared $s_i t_j$, the players open $r-1$ differences of these values to verify that they are all equal to 0. This does not violate secrecy because if no cheater is involved, no information will be leaked. On the other hand, if at least one cheater is involved, secrecy need not be guaranteed since the adversary knew s_i and t_i beforehand. If any of the differences is not 0, then s_i and t_j are reconstructed and $s_i t_j$ is computed openly and shared with a default sharing. Correctness is guaranteed as long as one of the involved players is honest since successful cheating requires to pass the checking phase without any complaints. This is guaranteed if the condition

$$P \notin \Sigma \sqcup \Sigma \sqcup \Delta \tag{4}$$

is satisfied because each term $s_i t_j$ is known to the players in the complement of a set in $\Sigma \sqcup \Sigma$. This condition is also necessary. In summary, we have:

Multiplication protocol:

Precondition: Two values $s = \sum_{i=1}^{k} s_i$ and $t = \sum_{i=1}^{k} t_i$ are shared by VSS.

Postcondition: st is shared by VSS.

1. Each player p_m computes all terms $s_i t_j$ he can (i.e. those for which $m \in \overline{T_i} \cap \overline{T_j}$) and shares them using VSS.
2. For each (i, j), let $(p_{m_1}, \ldots, p_{m_r})$ be the ordered list of the players who computed $s_i t_j$ in step 1 (where r depends on i and j).
   The players (collectively) compute[12] and open the $r-1$ differences of the value shared by p_{m_1} and the value shared by p_{m_i}, for $i = 2, \ldots, r$.
3. If all these opened values are 0, then the sharing by p_{m_1} is used as the sharing of $s_i t_j$. Otherwise, s_i and t_j are reconstructed and the k-out-of-k sharing for the term $s_i t_j$ is defined (arbitrarily) as the list $(s_i t_j, 0, \ldots, 0)$ of shares.
4. The players (locally) compute the sum of their shares of all terms $s_i t_j$, resulting in a sharing of st.

The proof of the following lemma follows from the above discussion.

Lemma 6. *The above protocol is a secure multiplication protocol for secrecy structure Σ and adversary structure Δ, if $P \notin \Sigma \sqcup \Sigma \sqcup \Delta$.*

7 Conclusions

Because of the simplicity of the presented protocols, it is easy to verify that their complexity is polynomial in n, $|\Sigma|$, and $|\Delta|$. Although for a threshold adversary the complexity is exponential in n, for a very small number of players the protocol is very efficient and can possibly lead to the preferred protocol from a practical viewpoint.

[12] Computing the difference is achieved by each player computing the difference locally.

One advantage of the described protocol is that it works over any field or ring, in particular also over the binary field $GF(2)$. This is significant in view of the fact that a digital circuit can easily, and without essential loss of efficiency, be transformed into a circuit using only XOR and AND gates, hence into an arithmetic circuit over $GF(2)$. In contrast, other protocols require a field $GF(q)$ of size $q > n$, resulting possibly in a complexity overhead for translating the digital circuit into an arithmetic circuit over $GF(q)$.

A theme of general interest in secure MPC is to design protocols that are efficient in the size of the descriptions of the secrecy and the adversary structures (or, more generally, the adversary specification). Obviously, this task depends on which type of description one uses, i.e., on the adversary specification language. The specification language of this paper is the list of all maximal sets of a structure. Assuming $\Sigma = \Delta$, a protocol that is efficient for a substantially more powerful specification language was given in [CDM00]: Σ can be described by any linear secret-sharing scheme with secrecy structure Σ. It is an open problem to find other specification languages for which efficient protocols exist.

Acknowledgments

The author would like to thank Matthias Fitzi and Martin Hirt for interesting discussions on secure multi-party computation which were the motivation for this work. Martin also gave much appreciated feedback on this paper.

References

[Bea91] D. Beaver. Secure multi-party protocols and zero-knowledge proof systems tolerating a faulty minority. *Journal of Cryptology*, vol. 4, no. 2, pp. 75–122, 1991.

[BW98] D. Beaver and A. Wool. Quorum-based multi-party computations. *Advances in Cryptology — EUROCRYPT '98*, Lecture Notes in Computer Science, Springer-Verlag, vol. 1403, pp. 25–35, 1998.

[BGW88] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proc. 20th ACM Symposium on the Theory of Computing (STOC)*, pp. 1–10, 1988.

[BGP89] P. Berman, J. A. Garay, and K. J. Perry. Towards optimal distributed consensus (extended abstract). In *Proc. 21st ACM Symposium on the Theory of Computing (STOC)*, pp. 410–415, 1989.

[Can00] R. Canetti. Security and composition of multi-party cryptographic protocols. *Journal of Cryptology*, vol. 13, no. 1, pp. 143–202, 2000.

[CCD88] D. Chaum, C. Crépeau, and I. Damgård. Multi-party unconditionally secure protocols (extended abstract). In *Proc. 20th ACM Symposium on the Theory of Computing (STOC)*, pp. 11–19, 1988.

[CDD+99] R. Cramer, I. Damgård, S. Dziembowski, M. Hirt, and T. Rabin. Efficient multi-party computations with dishonest minority. *Advances in Cryptology — EUROCRYPT '99*, Lecture Notes in Computer Science, Springer-Verlag, vol. 1592, pp. 311–326, 1999.

[CDM00] R. Cramer, I. Damgård, and U. Maurer. General secure multi-party computation from any linear secret sharing scheme. *Advances in Cryptology – EUROCRYPT 2000*, B. Preneel (Ed.), Lecture Notes in Computer Science, Springer-Verlag, vol. 1807, pp. 316–334, 2000.

[FM02] S. Fehr and U. Maurer. Linear VSS and Distributed Commitment Schemes Based on Secret Sharing and Pairwise Checks. *Advances in Cryptology – CRYPTO '02*, M. Yung (Ed.), Lecture Notes in Computer Science, Springer-Verlag, vol. 2442, pp. 565–580, 2002.

[FM97] P. Feldman and S. Micali. An optimal probabilistic protocol for synchronous Byzantine agreement. *SIAM Journal on Computing*, 26(4):873–933, Aug. 1997.

[FHM98] M. Fitzi, M. Hirt, and U. Maurer. Trading correctness for secrecy in unconditional multi-party computation. *Advances in Cryptology — CRYPTO '98*, Lecture Notes in Computer Science, Springer-Verlag, vol. 1462, pp. 121–136, 1998. (See corrected version at `http://www.crypto.ethz.ch/publications`).

[FHM99] M. Fitzi, M. Hirt, and U. Maurer. General adversaries in unconditional multi-party computation, *Advances in Cryptology – ASIACRYPT '99*, K.Y. Lam et al. (Eds.), Lecture Notes in Computer Science, Springer-Verlag, vol. 1716, pp. 232–246, 1999.

[FM98] M. Fitzi and U. Maurer. Efficient Byzantine agreement secure against general adversaries. *Distributed Computing — DISC '98*, Lecture Notes in Computer Science, Springer-Verlag, vol. 1499, pp. 134–148, 1998.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game — a completeness theorem for protocols with honest majority. In *Proc. 19th ACM Symposium on the Theory of Computing (STOC)*, pp. 218–229, 1987.

[HM97] M. Hirt and U. Maurer. Complete characterization of adversaries tolerable in secure multi-party computation. *Proc. 16th ACM Symposium on Principles of Distributed Computing (PODC)*, pp. 25–34, Aug. 1997.

[HM00] M. Hirt and U. Maurer. Player simulation and general adversary structures in perfect multi-party computation. *Journal of Cryptology*, vol. 13, no. 1, pp. 31–60, 2000.

[HM01] M. Hirt and U. Maurer. Robustness for free in unconditional multi-party computation. *Advances in Cryptology – CRYPTO '01*, J. Kilian (Ed.), Lecture Notes in Computer Science, Springer-Verlag, vol. 2139, pp. 101–118, 2001.

[ISN87] M. Ito, A. Saito, and T. Nishizeki. Secret-sharing scheme realizing general access structure. *Proceedings IEEE Globecom '87*, pp. 99–102. IEEE, 1987.

[LSP82] L. Lamport, R. Shostak, and M. Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, vol. 4, pp. 382–401, July 1982.

[Mau02] U. Maurer. Secure multi-party computation made simple. *Security in Communication Networks (SCN'02)*, G. Persiano (Ed.), Lecture Notes in Computer Science, Springer-Verlag, vol. 2576, pp. 14–28, 2003.

[PSW00] B. Pfitzmann, M. Schunter, and M. Waidner. Secure Reactive Systems. IBM Research Report RZ 3206, Feb. 14, 2000.

[RB89] T. Rabin and M. Ben-Or. Verifiable secret-sharing and multiparty protocols with honest majority. In *Proc. 21st ACM Symposium on the Theory of Computing (STOC)*, pp. 73–85, 1989.

[Yao82] A. C. Yao. Protocols for secure computations. *Proc. 23rd IEEE Symposium on the Foundations of Computer Science (FOCS)*, pp. 160–164. IEEE, 1982.