Trading Correctness for Privacy in Unconditional Multi-Party Computation*

Corrected Version**

Matthias Fitzi, Martin Hirt, and Ueli Maurer

Department of Computer Science
Swiss Federal Institute of Technology (ETH), Zurich
CH-8092 Zurich, Switzerland,
{fitzi,hirt,maurer}@inf.ethz.ch

Abstract. This paper improves on the classical results in unconditionally secure multi-party computation among a set of n players, by considering a model with three simultaneously occurring types of player corruption: the adversary can actively corrupt (i.e. take full control over) up to t_a players and, additionally, can passively corrupt (i.e. read the entire information of) up to t_p players and fail-corrupt (i.e. stop the computation of) up to t_f other players. The classical results in multi-party computation are for the special cases of only passive ($t_a = t_f = 0$) or only active ($t_p = t_f = 0$) corruption. In the passive case, every function can be computed securely if and only if $t_p < n/2$. In the active case, every function can be computed securely if and only if $t_a < n/3$; when a broadcast channel is available, then this bound is $t_a < n/2$. These bounds are tight.
Strictly improving these results, one of our results states that, in addition to tolerating $t_a < n/3$ actively corrupted players, privacy can be guaranteed against every minority, thus tolerating *additional* $t_p \leq n/6$ passively corrupted players. These protocols require no broadcast and have an exponentially small failure probability. We further show that the bound $t < n/2$ for passive corruption holds even if the adversary is additionally allowed to make the passively corrupted players fail.
Moreover, we characterize completely the achievable thresholds t_a, t_p and t_f for four scenarios. Zero failure probability is achievable if and only if $3t_a + 2t_p + t_f < n$; this holds whether or not a broadcast channel is available. Exponentially small failure probability with a broadcast channel is achievable if and only if $2t_a + 2t_p + t_f < n$; without broadcast, the additional condition $3t_a + t_f < n$ is necessary and sufficient.
In this corrected version, an error pointed out by Damgård [Dam99] is corrected.

Keywords. Secure multi-party computation, unconditional security, verifiable secret sharing, threshold cryptography.

* Research supported by the Swiss National Science Foundation (SNF), SPP project no. 5003-045293.

** This is the corrected version of a paper with the same title which appeared in the proceedings of *CRYPTO '98*, vol. 1462 of LNCS, Springer-Verlag, 1998 [FHM98].

1 Introduction

1.1 Secure Multi-Party Computation

Consider a set of n players who do not trust each other. Nevertheless they want to compute an agreed function of their inputs in a secure way. Security means achieving correctness of the result of the computation while keeping the players' inputs private, even if some of the players cheat. This is the well-known secure multi-party computation problem (e.g. [Yao82,GMW87,BGW88,CCD88,RB89]). For an excellent overview see [Fra93,Can95]. Secure multi-party computation can alternatively, and more generally, be seen as the problem of performing a task among a set of players that is specified by involving a trusted party, where the goal of the protocol is to replace the need for the trusted party. In other words, the functionality of the trusted party is shared among the players. Secure function evaluation described above can be seen as a special case of this more general setting. Most protocols described in the literature in the context of secure function evaluation also apply in the general context. This is also true for the protocols described in this paper.

There exists a rich literature on the subject. Previous approaches can be classified according to a number of criteria that are briefly discussed below.

- *Communication*: The communication models differ with respect to several criteria: Whether or not secure communication channels are available, whether or not broadcast channels are available, and whether the communication channels are synchronous or asynchronous. Some papers also consider incomplete network topologies, i.e. limited connectivity.
- *Adversaries*: Cheating players can be modeled by assuming a central adversary that can corrupt players. One generally distinguishes between actively corrupted players, passively corrupted players, and fail-corrupted players.
- *Security*: The basic security requirements are privacy and correctness. Privacy means that the adversary obtains no information about the uncorrupted players' inputs beyond what he learns from seeing the corrupted players' inputs and the output of the computation. Correctness means that the adversary cannot prevent the uncorrupted players from learning the correct output. Furthermore, robustness means that once the inputs have been committed by the players, the adversary cannot stop the computation. We refer to [Can97,Can98] for a precise definition of security in multi-party computation.

  The types of tolerable adversaries have recently been generalized in a number of directions (adaptive adversaries [CFGN96], uncoercibility [CG96], non-threshold adversaries [HM97]), and some authors have investigated multi-party computation for various minimality and complexity criteria [FKN94,CGT95,FY92,Kus89].

  Security can also be classified according to the adversary's computational resources (limited, hence cryptographic security, e.g. [CDG87,GMW87], or unlimited, hence unconditional or information theoretic security, e.g. [BGW88,CCD88,RB89]). In the information-theoretic model one can

distinguish between protocols with exponentially small (e.g. [CCD88,RB89]) or with zero failure probability (e.g. [BGW88]). We refer to the latter as *perfect* multi-party computation.
- *Generality*: Many papers describe protocol constructors which, for any given function, generate a protocol for securely computing this function (e.g. [Yao82,GMW87,GHY87,BGW88,CCD88,RB89,Cha89]), while other approaches are tailored to a particular function like voting (e.g. [CFSY96]), auctioning [FR96], sharing of encryption or signature operations [SDFY94,GJKR96a,GJKR96b], or private information retrieval [CGKS95,KO97].[1]

1.2 Previous Results and Motivation

Ben-Or, Goldwasser and Wigderson [BGW88] and independently Chaum, Crépeau and Damgård [CCD88] proved two fundamental results in unconditional multi-party computation among a set of n players. When the adversary only passively corrupts players, secure multi-party computation is possible if and only if the number of corrupted players is less than $n/2$. When the adversary only actively corrupts players, secure multi-party computation is possible if and only if the number of corrupted players is less than $n/3$. T. Rabin and Ben-Or [RB89] proved that when a broadcast channel is available, then the threshold for active corruption is $n/2$. These bounds are tight in the sense that every function can be securely computed when the threshold condition is satisfied, but there exist functions (in fact almost all functions) that cannot be securely computed when the threshold condition is violated.

The protocols of [BGW88] have zero failure probability, i.e. they are perfect. The failure probability can be made exponentially small for the protocols of [CCD88] and [RB89]. The results of [RB89] cannot be achieved with zero failure probability.

This paper introduces and investigates a new model for multi-party computation in which the adversary can simultaneously perform three different kinds of player corruption: active corruption, passive corruption, and fail-corruption.

Special cases of adversaries that perform different kinds of player corruption have previously been considered in the literature. Chaum [Cha89] considered general multi-party protocols that are secure against an adversary that can corrupt either up to d players actively or, alternatively, can corrupt up to c players passively, but in his model only one type of corruption occurs at the same time.[2] In other words, the adversary chooses between active or passive corruption and all corrupted players are corrupted in the same way. The protocols achieve correctness (with negligible failure probability) with respect to any d actively corrupted,

[1] The major reason for designing protocols for special functions compared to applying a general-purpose protocol is the potential gain of efficiency.

[2] For instance, the thresholds $c = 1$ and $d = \lfloor (n-1)/2 \rfloor$ given in the example in Section 3.1 of [Cha89] cannot be tolerated if simultaneous passive and active corruption occur.

or privacy with respect to any c passively corrupted players, where $2c + d < n$ is required. Meyer and Pradhan [MP91], and Garay and Perry [GP92] treated the case of simultaneous active and fail-corruptions for Byzantine agreement. The simultaneous presence of active and passive player corruption was considered by Dolev et al. [DDWY93] in the context of secure message transmission over general networks. That paper also mentions the possibility of extending those results to multi-party computation. In contrast to [Cha89], the models of [MP91,GP92,DDWY93] and also of this paper consider a *mixed* adversary that *simultaneously* performs different kinds of corruption.

1.3 Contributions of this Paper

We introduce a new model for unconditionally secure multi-party computation among a set of n players where the adversary may actively corrupt, passively corrupt, and fail-corrupt some of the players. We refer to this model as the *mixed model.* A (t_a, t_p, t_f)-*adversary* (or simply a (t_a, t_p)-adversary when $t_f = 0$) may actively corrupt up to t_a players and, additionally, passively corrupt up to t_p and fail-corrupt up to t_f other players. To actively corrupt a player means to take full control over the corrupted player, i.e. to receive and send arbitrary messages in this player's name. To passively corrupt a player means to have access to all information that this player receives during the whole protocol, including his inputs to the protocol (if any). To fail-corrupt a player means that the adversary may stop the communication from and to that player at an arbitrary time during the protocol. Once a player is caused to fail, he will not recover. However, the adversary is not allowed to read the internal data of a fail-corrupted player, unless the player is also passively corrupted at the same time. In other words, if the adversary wants both to read the internal information of a player and to be able to make the player fail, then he must both passively corrupt and fail-corrupt that player, and the player is counted for both thresholds t_p and t_f. Note that the cases $t_a = t_f = 0$ and $t_p = t_f = 0$ correspond to the passive and active model, respectively, as considered in the previous literature.

A protocol is (t_a, t_p, t_f)-*secure* (or (t_a, t_p)-secure) if any (t_a, t_p, t_f)-adversary (or (t_a, t_p)-adversary) obtains no additional information about the non-corrupted players' inputs (beyond what is provided by the function output) and cannot falsify the outcome of the computation. For a precise definition of security, see [Can97,Can98].

In this paper, we consider information-theoretic security, i.e. the adversary can use unlimited computational resources. Players are connected pairwise by secure communication channels in a synchronous network. The necessary and sufficient conditions for secure multi-party computation to be achievable for all functions are derived for the following four cases:

– Perfect security, without a broadcast channel, is possible if and only if $3t_a + 2t_p + t_f < n$.
– Perfect security, with a broadcast channel, is possible under the same condition, i.e., broadcast does not help increasing the number of players that can be corrupted when perfect security is required.

- Unconditional security (negligible failure probability), without a broadcast channel, is possible if and only if $2t_a + 2t_p + t_f < n$ and $3t_a + t_f < n$.
- Unconditional security, with a broadcast channel, is possible if and only if $2t_a + 2t_p + t_f < n$.

1.4 Changes in this Corrected Version

Damgård [Dam99] pointed out that the multiplication protocol of [FHM98] for the model with perfect security is not secure with respect to the weaker conditions $3t_a + t_p + t_f < n$ and $2t_a + 2t_p + t_f < n$ given in [FHM98], but only with respect to the stronger condition $3t_a + 2t_p + t_f < n$. In this corrected version, we prove that the multiplication protocol is indeed perfectly secure for this stronger condition, and additionally, we proof the necessity of this new condition. The corrections concern Lemma 2, Theorem 1 and 4. The results for unconditional security (with negligible error probability) are not affected.

1.5 Outline

In Sections 2 and 3 tight bounds on the existence of perfectly secure (Section 2) or unconditionally secure (Section 3) protocols are given for the mixed model without fail-corruptions. In Section 4, these bounds are generalized for the mixed model with fail-corruptions, and the protocols are extended to achieve these generalized bounds. Some conclusions are given in Section 5.

2 Perfectly Secure Multi-Party Computation

The construction of perfectly secure protocols for the mixed model without fail-corruptions is based on the protocol of [BGW88] for the active model. We first give a very brief summary of that protocol, and then present some modifications needed for tolerating a mixed adversary.

2.1 The Protocol of Ben-Or, Goldwasser, and Wigderson

The protocol of [BGW88] for the active model requires that the function to be computed is represented as an arithmetic circuit over a finite field $(\mathcal{F}, +, *)$, where three types of gates are available: an addition gate takes two inputs and outputs their sum, a multiplication gate takes two inputs and outputs their product, and a randomness gate takes no input and outputs a random field element. The protocol proceeds in three stages: In the *input stage*, every player uses verifiable secret sharing (VSS) to share his input among all players. In the *computation stage*, the agreed circuit is evaluated gate by gate, where all intermediate results are shared among the players using VSS. In the *final stage*, the shared result is reconstructed.

We briefly describe the subprotocols for verifiable secret sharing and for multiplication. Let n denote the number of players, and let t denote the upper bound on the number of players that the adversary may actively corrupt.

Verifiable secret sharing is based on Shamir's secret sharing scheme [Sha79]. To every player a unique field element w^i is associated, where w is a primitive n-th root of unity. In the sharing protocol, the dealer randomly selects a two-dimensional polynomial $f(x, y)$ of degree t in both variables such that $f(0, 0)$ is the value to be shared, and sends the polynomials $f_i(x) = f(x, w^i)$ and $g_i(y) = f(w^i, y)$ to the i-th player. Then the i-th and the j-th player verify the cross-over points $f_i(w^j) \stackrel{?}{=} g_j(w^i)$ and $g_i(w^j) \stackrel{?}{=} f_j(w^i)$ and complain about inconsistencies. Whenever a player recognizes an inconsistency, he asks the dealer to broadcast the value of the inconsistent cross-over points. Whenever a player receives a value from the dealer that is not consistent with a previously received value, he accuses the dealer, who then has to broadcast both polynomials of the accusing player. Then the player uses these broadcast polynomials as his polynomials. If more than t players accused the dealer, then clearly the dealer is corrupted and every player picks the default zero polynomial as the dealer's polynomial. Finally, the share of the i-th player is $f_i(0)$. This sharing protocol guarantees that all the shares lie on a polynomial of degree t, and that if the dealer is honest (uncorrupted), then any t players do not obtain any joint information about the secret. In the reconstruction protocol, each player sends his share of the result to those players who should learn this value. Each of them can then correct faulty shares using the properties of the underlying Reed-Solomon code [MS81], and interpolate the secret. At most t shares are faulty and the minimal distance of the Reed-Solomon code with n shares and degree t is $n-t-1$. Hence $(n-t-1)/2 \geq t$ faults can efficiently be corrected and reconstruction is possible.

Linear Functions. Multiplication by scalars and addition of shared values, and hence the computation of any linear function, can be performed (without communication) by each player doing the corresponding computation on his shares and keeping the result as a share of the new value.

Multiplication of two shared values is more involved. Here we describe the improved protocol of [GRR98]. First, every player verifiably secret shares both shares of the two values he holds. Because a corrupted player could secret share a wrong share, the syndrome of the reshared shares is computed in a distributed manner (syndrome computing is linear) and is then reconstructed (reconstruction involves local error-correction for every player). Using this syndrome vector, each player can derive the error vector and can correct his shares according to the error vector. Then, each player shares the product of his two factor shares and proves by a subprotocol that the shared value indeed is the product of his two factor shares. At this point, every player holds a share of all n product shares, and as shown in [GRR98] the interpolation of these product shares results in the product of the two factors. Interpolation is linear and can be computed locally on the shares of the product shares.

2.2 Protocol Modifications for the Mixed Model

Lemma 1. *For n players and arbitrary thresholds t_a and t_p satisfying $3t_a+t_p < n$, there exists a verifiable secret-sharing scheme that is perfectly secure against a (t_a,t_p)-adversary.*

Proof. The scheme is a variation of the verifiable secret-sharing scheme presented in [BGW88] whereby the degree of the used polynomials is set to $d = t_a + t_p$. This modification was first proposed in [Dwo90] and [DDWY93]. In the sharing protocol, the dealer selects a two-dimensional polynomial $f(x, y)$ of degree d in each variable and sends the polynomials $f_i(x)$ and $g_i(y)$ to the i-th player. Then the players verify the cross-over points and complain about inconsistencies. Whenever a player recognizes that the dealer is misbehaving, he accuses the dealer. If more than t_a players accused the dealer, then clearly the dealer is faulty and every player picks the default zero polynomial as the dealer's polynomial. Therefore, in the reconstruction protocol the parameter t of the maximal number of errors is set to t_a.

Clearly, this scheme tolerates a (t_a, t_p)-adversary. The complete information the adversary obtains during the protocol contains at most $t_a + t_p$ shares, which gives no information about the secret (unless the dealer is corrupted). On the other hand, coordinated misbehavior of up to t_a players can be tolerated because in the reconstruction protocol, up to $(n-d-1)/2$ faulty shares can be corrected (half of the minimum distance of the underlying Reed-Solomon code), which is at least t_a. □

Lemma 2. *A set of n players can (t_a,t_p)-securely multiply two shared values if $3t_a + 2t_p < n$.*

Proof. The multiplication protocol is along the lines of the multiplication protocol in [BGW88] (including the improvements of [GRR98]), whereby the VSS of Lemma 1 with polynomials of degree $d = t_a + t_p$ is used.

First, every player verifiably secret shares both shares he holds of the two factors (using the VSS scheme of Lemma 1), then the syndrome of the reshared shares is distributively computed and reconstructed, and the shares are corrected according to this syndrome. Now, every player p_i verifiably shares the product of his two shares by applying the technique described in [BGW88]: First, every player p_j computes the local product of both share-shares of p_i's shares. These product share-shares of all players p_j define a degree-$2d$ polynomial $D(x)$ with $D(0)$ being p_i's product share. Then the player p_i reduces the degree of $D(x)$ to d by sharing d polynomial $D_1(x), \ldots, D_d(x)$ and computing $C(x) = D(x) - x^1D_1(x) - \ldots - x^dD_d(x)$ (cf. [BGW88]), and proves that the resulting polynomial $C(x)$ has indeed degree d. This technique requires the condition $2d < n - t_a$ (see [Dam99]). Finally, the players interpolate the product [GRR98]. The described protocol is secure as long as the underlying verifiable secret-sharing protocol is secure, which is guaranteed for $3t_a + t_p < n$ by Lemma 1. □

Theorem 1. *A set of n players can compute every function perfectly (t_a, t_p)-securely if and only if $3t_a + 2t_p < n$. The computation is polynomial in n and linear in the size of the circuit. This holds whether or not a broadcast channel is available.*

Proof. ($\Longleftarrow$) As shown in Lemma 1, the verifiable secret-sharing scheme of [BGW88] can be modified such that a (t_a, t_p)-adversary is tolerated if $3t_a + t_p < n$. Addition and selection of random field elements need no modification (except that the VSS scheme of Lemma 1 is used). Due to the condition $3t_a + 2t_p < n$, also multiplication can be performed as explained in Lemma 2.
($\Longrightarrow$) In order to prove the necessity of $3t_a + 2t_p < n$, assume for the sake of contradiction that for some t_a, t_p with $3t_a + 2t_p \geq n$ every function can be computed perfectly (t_a, t_p)-securely. Then one can construct a protocol for three players p_1, p_2, and p_3, where p_1 plays for $t_a + t_p$ players, p_2 plays for $t_a + t_p$ other players, and p_3 plays for the remaining at most t_a players. This new protocol is secure with respect to an adversary that passively corrupts either p_1 or p_2, or actively corrupts p_3.

Assume that the specification requires to compute the logical `AND` of two bits x_1 and x_2 held by p_1 and p_2, respectively, and assume for the sake of contradiction that a protocol for this specification is given. Let T denote the transcript of the broadcast channel of a run of that protocol (if no broadcast channel is available, let $T = \emptyset$), and let T_{ij} $(1 \leq i < j \leq 3)$ denote the transcript of the channels between p_i and p_j. Due to the requirement of perfect privacy, p_1 will not send any information about his bit x_1 over T_{12} or over T before he knows x_2 (if p_1 knows that $x_2 = 1$ he can reveal x_1). Similarly, p_2 will not send any information about x_2 over T_{12} or over T before he knows x_1. Hence the only escape from this deadlock would be to use p_3. However, as T_{12} and T jointly give no information about x_2, a random misbehavior of an actively corrupted p_3 (ignore all received messages and send random bits whenever a message must be sent) would with some (possibly negligible) probability make p_1 receive the wrong output, contradicting the perfect security of the protocol.

3 Unconditionally Secure Multi-Party Computation

We now show that by allowing a negligible rather than zero failure probability, the bounds for the number of corrupted players can be improved.

Theorem 2. *Allowing a negligible failure probability and given a broadcast channel, a set of n players can compute every function (t_a, t_p)-securely if and only if $2t_a + 2t_p < n$. The computation is polynomial in n and linear in the size of the circuit.*

Proof. ($\Longleftarrow$) The protocol of [RB89] tolerates up to $t_a < n/2$ actively corrupted players. Hence, for all t_a and t_p that satisfy $2t_a + 2t_p < n$, this protocol is also (t_a, t_p)-secure.[3] The efficiency of this protocol is proven in [RB89].

[3] However, the VSS reconstruction protocol of [RB89] needs a slight modification such that it is possible to reveal a shared value only to some of the players (the original

($\Longrightarrow$) If for some t_a and t_p satisfying $2t_a + 2t_p \geq n$, every function were (t_a, t_p)-securely computable, then every function would also be securely computable against a passive majority, contradicting Theorem 2 of [BGW88]. □

Theorem 3. *Allowing a negligible failure probability, without a broadcast channel, a set of n players can compute every function (t_a, t_p)-securely if and only if $2t_a + 2t_p < n$ and $3t_a < n$. The computation is polynomial in n and linear in the size of the circuit.*

Proof. ($\Longleftarrow$) The protocol of [RB89] is (t_a, t_p)-secure if $2t_a + 2t_p < n$. Provided that $t_a < n/3$, a broadcast channel can be simulated among the players [LSP82,FM88,GM93]. Simulating broadcast in the protocol of [RB89] yields a protocol that is $(t_a + t_p)$-secure for all t_a and t_p that satisfy $2t_a + 2t_p < n$ and $3t_a < n$. The efficiency of this protocol immediately follows from the efficiency of the protocols in [RB89] and in [FM88,GM93].
($\Longrightarrow$) The necessity of the condition $2t_a + 2t_p < n$ has already been shown in the proof of Theorem 2. On the other hand, let $3t_a \geq n$, and suppose that for every function there exists an unconditionally secure multi-party protocol with negligible failure probability. Since broadcast is one particular function, there would also exists a broadcast protocol that is unconditionally secure against one third of the players, contradicting a result of [LSP82,KY]. □

4 Multi-Party Computation Tolerating Fail-Corruptions

In this section we generalize the results of the previous two sections by allowing the adversary to fail-corrupt up to t_f players, and we give tight bounds for the number of actively corrupted, passively corrupted and fail-corrupted players that can be tolerated (where players that are passively corrupted and fail-corrupted at the same time must be counted for both thresholds). First, we consider a perfect model (with or without a broadcast channel), then an unconditional model with a broadcast channel, and last, an unconditional model without a broadcast channel.

Theorem 4. *A set of n players can compute every function perfectly (t_a, t_p, t_f)-securely if and only if $3t_a + 2t_p + t_f < n$. The computation is polynomial in n and linear in the size of the circuit. This holds whether or not a broadcast channel is available.*

Proof. ($\Longleftarrow$) The protocol described in the proof of Theorem 1 needs further modifications in order to be (t_a, t_p, t_f)-secure.
Invariant. The degree of the polynomials used for secret sharing is set to $d = t_a + t_p$. The invariant during the computation is that every (intermediate) value is shared among the players by a polynomial of degree d, and that every player is committed to his share. This means that whenever a player has to

protocol reveals the secret always to all players in parallel, thus only applies to the secure function evaluation model).

reveal his share, every player can verify whether the revealed share is the correct share or not.

VSS. Note that the described VSS of Lemma 1 either achieves this invariant, or the dealer is detected to be corrupted: After the sharing protocol, either there were more than t_a accusations against the dealer (in this case, the dealer is disqualified, see below), or every player holds his share, and a share of the share of each other player (called a share-share). The set of the share-shares of a given share can be considered as a shared commitment of that share (cf. [CDM98]). In order to reconstruct the secret, every player must reveal his share by broadcasting the polynomial used for sharing his share, and every player verifies that his corresponding share-share really lies on the broadcast polynomial. If not, the player broadcasts a complaint against the revealing player. If the revealing player is honest, there are at most t_a complaints. Now consider the case that the revealing player is actively corrupted and broadcasts a wrong polynomial. There are at most $t_a + t_f$ players who possibly do not complain even if their shares do not lie on the broadcast polynomial. However, if the polynomial is wrong, it differs from the correct polynomial at least at $n - d = n - t_a - t_p$ positions, and hence at least $(n - t_a - t_p) - (t_a + t_f)$ complaints are reported, which is strictly more than t_a (since $3t_a + 2t_p + t_f < n$). Once every player has revealed his share, the secret is reconstructed by interpolating the shares for which at most t_a complaints were reported (there are at least $n - t_a - t_f$ such shares).[4]

Computations. Addition and other linear functions can easily be computed on the shared values (preserving the invariant), since both the secret sharing and the commitments are linear. The multiplication protocol is along the lines of the protocol described in Lemma 2. Due to the invariant every share is also shared among the players, and there is no need for having every player share his factor shares as in [BGW88]. Every player can directly verifiably secret share his product share and use the technique described in [BGW88] to prove that this shared value is equal to the product of his two shares (if the player fails to prove this, he is disqualified; see below). Due to the linearity of the rest of the computation, the invariant is preserved. Note that, as used in [CDM98], this multiplication protocol does not involve error correction but error detection (in contrast to the multiplication protocol of [BGW88]).

Disqualifying players. One major issue of the above protocol is how to deal with disqualified players. In [BGW88], the disqualified players' data is reconstructed (implicitly by computing the error vector). But in this protocol, a disqualified player can either be actively corrupted or fail-corrupted, and his share must not be reconstructed if the player is fail-corrupted. It turns out that, as

[4] This construction only covers global reconstruction where every player learns the shared secret. If some result must be revealed to only certain players, then the reconstruction protocol must be slightly modified: Every player sends his share, the polynomial with which his share was shared (committed), and all his share-shares to the players who are supposed to learn the output, who then locally determine which of the shares are correct (at most t_a wrong share-shares), and interpolate the verified shares.

a consequence of the (corrected) bound $3t_a + 2t_p + t_f < n$, data reconstruction is not needed at all, i.e., that disqualified players' data can simply be ignored since all protocol steps can still be carried out correctly even if all actively and fail-corrupted players are disqualified. For two reasons, the following player elimination technique of the original paper is still described here although being obsolete for this particular protocol. First, the same technique will be needed for the protocol in the scenario with unconditional security (in the proof of Theorem 6) and, second, it might be an important contribution by itself with potential applications to different contexts.

If during the verifiable secret sharing protocol or during the multiplication protocol a player is disqualified, then he is excluded from all further computation. Generally, the shares of a disqualified player must not be revealed. Indeed, the shares of up to t_f disqualified players can simply be omitted (this is tolerated in the multiplication protocol). Thus, the first t_f times that some player is disqualified the player is excluded from the computation, but no further steps are taken. If more than t_f players are disqualified, then there are at most t_f fail-corrupted players among them, and the other disqualified players (say k) must be actively corrupted. Hence, among the remaining (non-disqualified) players there are at most $t_a - k$ actively corrupted players. At this point, every intermediate value is reshared using a scheme for $n' = n - k$ players, tolerating only $t'_a = t_a - k$ actively corrupted players. In order to do so, for every intermediate value, every player verifiably secret shares his share of that value among the n' remaining players by using polynomials of degree $d - k$, and proves that the shared share is equal to his original share. This can easily be verified by computing the difference of these two shares (in a distributive manner, i.e. every player subtracts one share-share from the other) and by verifying that the difference is zero. Finally, the secret is interpolated in a distributive manner (interpolation is linear). This results in every player having a reduced-degree share of the secret which is committed by a reduced-degree polynomial. If during this degree-reduction protocol some players refuse to cooperate, these players are also disqualified and the reduction protocol restarts (with an increased k). The number of restarts is limited by t_a because at most $t_a + t_f$ players can ever be disqualified.

The broadcast channel used in the protocol can be simulated by the broadcast protocol of [GP92] tolerating t_a active and t_f fail corruptions when $3t_a + t_f < n$.

($\Longrightarrow$) In order to prove the necessity of the conditions of the theorem, assume that for some t_a, t_p, and t_f there exists for every function a (t_a, t_p, t_f)-secure protocol among n players. Trivially, each such protocol is also a (t_a, t_p)-secure protocol among $n' = n - t_f$ players, simply by considering the case that the adversary fail-corrupts the last t_f players at the very beginning of the protocol. However, as proven in Theorem 1, a (t_a, t_p)-secure protocol among n' players exists for every function only if the condition $3t_a + 2t_p < n'$ is satisfied, which implies the conditions of the theorem. □

The following theorem shows that in the passive model, resilience against fail-corruptions is for free, i.e. the same bound that holds in the passive model also holds in a model in which the adversary may make the passively corrupted players fail.

Theorem 5. *A set of n players can compute every function perfectly securely with respect to an adversary that may passively corrupt and fail-corrupt up to t players if and only if $2t < n$. The computation is polynomial in n and linear in the size of the circuit.*

Proof. The protocol described in the proof of Theorem 4 achieves this bound after some slight modifications: The degree of all polynomials is set to $d = t$, and whenever a player fails, then his share is reconstructed using the share-shares, and becomes common knowledge.

Theorem 6. *Allowing a negligible failure probability and given a broadcast channel, a set of n players can compute every function (t_a, t_p, t_f)-securely if and only if $2t_a + 2t_p + t_f < n$. The computation is polynomial in n and linear in the size of the circuit.*

Proof (sketch). ($\Longleftarrow$) The protocol described in Theorem 2 needs to be modified along the lines of the modifications in the proof of Theorem 4. During the protocol, the shares of disqualified players are never reconstructed. Instead, the first t_f disqualified players are simply excluded from the protocol, and their shares are omitted (are considered as erasures when interpolating). However, after more than t_f players have been disqualified, it is impossible to perform further multiplications (there are not enough shares to interpolate the polynomial with double degree), and therefore all shared values are reshared using polynomials of smaller degree. In order to do so, the WSS of every share is upgraded to VSS, then the player verifiably secret shares the same share using VSS with reduced degree and proves that the same share was reshared. Finally, the secret is reconstructed in a distributive manner (interpolation is linear), using this new reduced-degree VSS, which results in each player having shared his reduced-degree share by WSS. If during this degree-reduction process a player refuses to cooperate, then this player is also disqualified and the degree-reduction process restarts (where the degree of the polynomials is decremented once more).
($\Longrightarrow$) The optimality of the conditions follows directly from the proof of Theorem 2 by assuming an adversary that corrupts the last t_f players at the very beginning of the protocol (obtaining a protocol among $n - t_f$ players that is (t_a, t_p)-secure).

Theorem 7. *Allowing a negligible failure probability, without a broadcast channel, a set of n players can compute every function (t_a, t_p, t_f)-securely if and only if $2t_a + 2t_p + t_f < n$ and $3t_a + t_f < n$. The computation is polynomial in n and linear in the size of the circuit.*

Proof (sketch). The additional condition $3t_a + t_f < n$ allows to simulate the assumed broadcast channel in the proof of Theorem 6 by a protocol [GP92]. The optimality of the condition follows immediately from the optimality of the condition in Theorem 6 and of the protocol in [GP92].

5 Conclusions

We have proposed a new model for multi-party computation that, by considering the simultaneous presence of actively, passively and fail-corrupted players, is more general than those in the previous literature. We proved the exact conditions for the existence of unconditionally secure multi-party computation for four natural cases. Moreover, we gave constructions of efficient protocols for all cases.

One of the main problems in multi-party computation is that it is not known *a priori* which players are corrupted. Trivially, if it is known in advance that a certain subset of the players are corrupted (the type of corruption is irrelevant), then the exact conditions for secure protocols to exist are those of this paper applied to the set of remaining players after elimination of the known cheaters.

It is surprising that regarding fail-corruptions, this condition holds even when it is not known in advance which players will be fail-corrupted. This follows from the fact that in all bounds t_f appears with coefficient only 1. In contrast, t_p appears with coefficient 2, and hence every passive corruption must be compensated by one additional non-corrupted player. In the model with exponentially small error probability and a broadcast channel, misbehavior is detected and hence active corruptions count equally as passive corruptions. In all other models, t_a appears with coefficient 3, and hence every active corruption must be compensated by two additional non-corrupted players.

Figure 1 summarizes our results. Because the conditions for protocols with failures are the same as those without failures when the number n of players is reduced by t_f to $n' = n - t_f$, the bounds are illustrated only in two dimensions (t_a and t_p) for a constant t_f. The brightly shaded area is achievable by perfectly secure multi-party computation. By accepting an exponentially small failure probability, one can additionally achieve the semi-dark shaded area. Finally, if a broadcast channel is available, one can even achieve the dark shaded area. These bounds are tight.

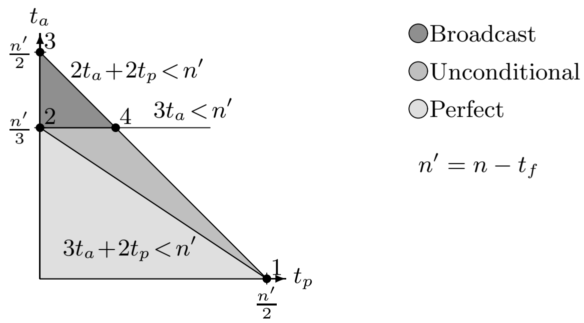

Fig. 1. Graphical representation of the results.

For the following considerations we restrict our view to the special case with no fail-corruptions ($t_f = 0$, and hence $n' = n$). The classical results in unconditional multi-party computation are special cases of the mixed model. The results of Ben-Or, Goldwasser and Wigderson [BGW88] correspond to the points 1 (passive model) and 2 (active model) in Figure 1. The result of Rabin and Ben-Or [RB89] corresponds to point 3.

Point 4 ($t_a = n/3$, $t_p = n/6$) illustrates that the results of [BGW88] for the active as well as for the passive model can be improved. In addition to $t_a < n/3$ actively corrupted players (point 2), one can tolerate additional $t_p \leq n/6$ passively corrupted players (point 4). However, this protocol has a negligible failure probability.

Finally, by approaching point 1 from point 2 in the case of perfect security, we can improve the resilience for privacy by reducing the resilience for correctness: In perfectly secure multi-party computation, three additional passive corruptions can be tolerated instead of two active corruption, thus trading correctness for privacy.

Acknowledgments

We would like to thank Ran Canetti for sending us early versions of [Can97,Can98] and David Chaum, Ronald Cramer, Ivan Damgård, Rosario Gennaro, Tal Rabin, Markus Stadler, and Michael Waidner for interesting discussions. Finally, we are grateful to the anonymous referees for their constructive remarks.

References

[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.

[Can95] R. Canetti. *Studies in Secure Multiparty Computation and Applications.* PhD thesis, Weizmann Institute of Science, Rehovot 76100, Israel, June 1995.

[Can97] R. Canetti. Modular composition of multi-party cryptographic protocols, Nov. 1997. Manuscript.

[Can98] R. Canetti. Security and composition of multi-party cryptographic protocols, June 1998. Manuscript.

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

[CDG87] D. Chaum, I. Damgård, and J. van de Graaf. Multiparty computations ensuring privacy of each party's input and correctness of the result. In *Advances in Cryptology — CRYPTO '87*, volume 293 of *Lecture Notes in Computer Science*, pp. 87–119. Springer-Verlag, 1987.

[CDM98] R. Cramer, I. Damgård, and U. Maurer. Span programs and general multi-party computation. Manuscript, 1998.

[CFGN96] R. Canetti, U. Feige, O. Goldreich, and M. Naor. Adaptively secure multi-party computation. In *Proc. 28th ACM Symposium on the Theory of Computing (STOC)*, pp. 639–648, Nov. 1996.

[CFSY96] R. Cramer, M. K. Franklin, B. Schoenmakers, and M. Yung. Multi-authority secret-ballot elections with linear work. In *Advances in Cryptology — EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pp. 72–83. IACR, Springer-Verlag, May 1996.

[CG96] R. Canetti and R. Gennaro. Incoercible multiparty computation. In *Proc. 37th IEEE Symposium on the Foundations of Computer Science (FOCS)*, 1996.

[CGKS95] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *Proc. 36th IEEE Symposium on the Foundations of Computer Science (FOCS)*, pp. 41–51, Oct. 1995.

[CGT95] C. Crépeau, J. van de Graaf, and A. Tapp. Committed oblivious transfer and private multi-party computation. In *Advances in Cryptology — CRYPTO '95*, volume 963 of *Lecture Notes in Computer Science*, pp. 110–123. Springer-Verlag, 1995.

[Cha89] D. Chaum. The spymasters double-agent problem. In *Advances in Cryptology — CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pp. 591–602. Springer-Verlag, 1989.

[Dam99] I. Damgård. Manuscript, to appear as a technical report of BRICS, 1999.

[DDWY93] D. Dolev, C. Dwork, O. Waarts, and M. Yung. Perfectly secure message transmission. *Journal of the ACM*, 40(1):17–47, Jan. 1993.

[Dwo90] C. Dwork. Strong verifiable secret sharing. In *Proc. 4th International Workshop on Distributed Algorithms '90*, volume 486 of *Lecture Notes in Computer Science*, pp. 213–227. Springer-Verlag, 1990.

[FHM98] M. Fitzi, M. Hirt, and U. Maurer. Trading correctness for privacy in unconditional multi-party computation. In *Advances in Cryptology — CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, 1998.

[FKN94] U. Feige, J. Kilian, and M. Naor. A minimal model for secure computation. In *Proc. 26th ACM Symposium on the Theory of Computing (STOC)*, pp. 554–563, 1994.

[FM88] P. Feldman and S. Micali. Optimal algorithms for Byzantine agreement. In *Proc. 20th ACM Symposium on the Theory of Computing (STOC)*, pp. 148–161, 1988.

[FR96] M. K. Franklin and M. K. Reiter. The design and implementation of a secure auction service. *IEEE Transactions on Software Engineering*, 22(5):302–312, May 1996.

[Fra93] M. K. Franklin. *Complexity and Security of Distributed Protocols*. PhD thesis, Columbia University, 1993.

[FY92] M. K. Franklin and M. Yung. Communication complexity of secure computation. In *Proc. 24th ACM Symposium on the Theory of Computing (STOC)*, pp. 699–710, 1992.

[GHY87] Z. Galil, S. Haber, and M. Yung. Cryptographic computation: Secure fault-tolerant protocols and the public-key model. In *Advances in Cryptology — CRYPTO '87*, volume 293 of *Lecture Notes in Computer Science*, pp. 135–155. Springer-Verlag, 1987.

[GJKR96a] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust and efficient sharing of RSA functions. In *Advances in Cryptology — CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pp. 157–172. Springer-Verlag, Aug. 1996.

[GJKR96b] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust threshold DSS signatures. In *Advances in Cryptology — EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pp. 354–371. Springer-Verlag, May 1996.

[GM93] J. A. Garay and Y. Moses. Fully polynomial Byzantine agreement in $t+1$ rounds (extended abstract). In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, pp. 31–41, San Diego, California, May 1993.

[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.

[GP92] J. A. Garay and K. J. Perry. A continuum of failure models for distributed computing. In A. Segall and S. Zaks, editors, *Distributed Algorithms, 6th International Workshop, WDAG '92*, volume 647 of *Lecture Notes in Computer Science*, pp. 153–165, Haifa, Israel, 2–4 Nov. 1992. Springer.

[GRR98] R. Gennaro, M. O. Rabin, and T. Rabin. Simplified VSS and fast-track multiparty computations with applications to threshold cryptography. In *Proc. 17th ACM Symposium on Principles of Distributed Computing (PODC)*, 1998.

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

[KO97] E. Kushilevitz and R. Ostrovsky. Replication is not needed: Single data base computationally-private information retrieval. In *Proc. 38th IEEE Symposium on the Foundations of Computer Science (FOCS)*, pp. 364–373, 1997.

[Kus89] E. Kushilevitz. Privacy and communication complexity (extended abstract). In *Proc. 30th IEEE Symposium on the Foundations of Computer Science (FOCS)*, pp. 416–421, 1989.

[KY] A. Karlin and A. C. Yao. Unpublished manuscript.

[LSP82] L. Lamport, R. Shostak, and M. Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, July 1982.

[MP91] F. J. Meyer and D. K. Pradhan. Consensus with dual failure modes. In *IEEE Transactions on Parallel and Distributed Systems '91*, volume 2, pp. 214–221, 1991.

[MS81] R. J. McEliece and D. V. Sarwate. On sharing secrets and Reed-Solomon codes. *Communications of the ACM*, 24:583–584, 1981.

[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.

[SDFY94] A. de Santis, Y. Desmedt, Y. Frankel, and M. Yung. How to share a function securely. In *Proc. 26th ACM Symposium on the Theory of Computing (STOC)*, pp. 522–533, 1994.

[Sha79] A. Shamir. How to share a secret. *Communications of the ACM*, 22:612–613, 1979.

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