Multi-Party Computation with Omnipresent Adversary

Hossein Ghodosi[1] and Josef Pieprzyk[2]

[1] School of Mathematics, Physics and Information Technology
James Cook University, Townsville, Qld 4811, Australia,
[2] Department of Computing
Center for Advanced Computing – Algorithms and Cryptography
Macquarie University, Sydney, NSW 2109 Australia,

Abstract. Secure multi-party computation (MPC) protocols enable a set of n mutually distrusting participants $P_1, \ldots, P_n$, each with their own private input x_i, to compute a function $Y = F(x_1, \ldots, x_n)$, such that at the end of the protocol, all participants learn the correct value of Y, while secrecy of the private inputs is maintained. Classical results in the unconditionally secure MPC indicate that in the presence of an active adversary, every function can be computed if and only if the number of corrupted participants, t_a, is smaller than $n/3$. Relaxing the requirement of perfect secrecy and utilizing broadcast channels, one can improve this bound to $t_a < n/2$.

All existing MPC protocols assume that uncorrupted participants are truly honest, i.e., they are not even curious in learning other participant secret inputs. Based on this assumption, some MPC protocols are designed in such a way that after elimination of all misbehaving participants, the remaining ones learn all information in the system. This is not consistent with maintaining privacy of the participant inputs. Furthermore, an improvement of the classical results given by Fitzi, Hirt, and Maurer indicates that in addition to t_a actively corrupted participants, the adversary may simultaneously corrupt some participants passively. This is in contrast to the assumption that participants who are not corrupted by an active adversary are truly honest.

This paper examines the privacy of MPC protocols, and introduces the notion of an *omnipresent adversary*, which cannot be eliminated from the protocol. The omnipresent adversary can be either a passive, an active or a mixed one. We assume that up to a minority of participants who are not corrupted by an active adversary can be corrupted passively, with the restriction that at any time, the number of corrupted participants does not exceed a predetermined threshold. We will also show that the existence of a *t-resilient* protocol for a group of n participants, implies the existence of a *t'-private* protocol for a group of n' participants. That is, the elimination of misbehaving participants from a *t-resilient* protocol leads to the decomposition of the protocol.

Our adversary model stipulates that a MPC protocol never operates with a set of truly honest participants (which is a more realistic scenario). Therefore, privacy of all participants who properly follow the protocol

will be maintained. We present a novel disqualification protocol to avoid a loss of privacy of participants who properly follow the protocol.

Keywords: Multi-Party Computation, Omnipresent Adversary, Proactive Secret Sharing, *t-resilient* Protocols, *t-private* Protocols.

1 Introduction

Multi-party computation (MPC) protocols provide a general model for secure computation of arbitrary function whose arguments (inputs) are held by a group of participants. The concept of MPC was introduced by Yao [18] for two-party computations and then generalized by Goldreich, Micali, and Wigderson [12] for an arbitrary number of participants. A secure MPC protocol enables a set of n mutually distrusting participants $P_1, \ldots, P_n$, each with their own private input x_i, to compute a function $Y = F(x_1, \ldots, x_n)$, such that at the end of the protocol, all participants learn the correct value of Y, while the confidentiality of the private inputs x_i is maintained. The design of secure MPC protocols has been the subject of investigations by many researchers, and many solutions have been published in the literature. From the security point of view, these protocols can be classified into two broad categories: (i) *computationally secure* MPC protocols, and (ii) *unconditionally secure* MPC protocols. For computationally (conditionally) secure MPC protocols, we assume that the adversary is polynomially bounded. More precisely, breaking the security of the protocol implies that the adversary is able to solve efficiently (in polynomial time) a problem that is believed to be intractable. Unconditionally secure MPC protocols are intrinsically secure, that is, no matter how much time and computing power is available to the adversary, they cannot break the system better than by guessing private inputs. In both computationally and unconditionally secure MPC protocols, the security model includes the adversary, who may corrupt some participants. Two types of adversaries, namely *passive* and *active* have been studied in the literature.

Passive Adversary – Participants who are corrupted by a passive adversary properly follow the protocol but try to learn private information of others. That is, a passive adversary has access to the information of corrupted participants, but will not control their behaviour. In other words, a passive adversary threatens the privacy of uncorrupted participants. Note however, that the correctness of the protocol is preserved. Corrupted participants are also called *honest-but-curious*.

A commonly used parameter to measure the level of security obtained in a multi-party protocol with n participants, is the maximum number of participants that can be corrupted by a passive adversary while the privacy of uncorrupted participants still holds. This parameter is determined by a threshold t ($t < n$). Protocols that can tolerate up to t corrupted participants, are called *t-private*.

Definition 1. *A multi-party protocol is t-private if after completion of the protocol, any subset of up to t participants cannot learn more information (about*

honest participant private inputs) than what they could derive from their private inputs and the output of the protocol.

Active Adversary – A more serious threat for the security of MPC protocols are corrupted participants who not only try to learn additional information but may also wish to disrupt the protocol. This type of participants are called *malicious* and they are said to be corrupted by an *active adversary*, who has access to all information of the corrupted participants, and controls their behaviour. Participants who are corrupted by an active adversary may behave arbitrarily, and may deviate from the protocol at any time.

The main challenge in designing secure MPC protocols in the presence of an active adversary, is to equip the protocol with mechanisms that can detect misbehaving participants and eliminate them from the protocol without influencing the correctness of the protocol. Such protocols are called *robust*. A measure for expressing the level of robustness is determined by a threshold parameter t ($t < n$), that is the maximum number of participants that can be corrupted, without an impact on the correctness of the protocol. A protocol that can tolerate up to t malicious participants is called *t-resilient* and is defined as follows.

Definition 2. *A multi-party protocol is t-resilient if no set of up to t malicious participants can influence the correctness of the output produced by the protocol.*

1.1 Preliminaries

Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be a set of n participants who wish to compute a function $Y = F(x_1, \ldots, x_n)$, where P_i holds private input x_i. Without loss of generality, we will assume that all input variables are elements of a finite field E, and the function F can be computed by a circuit over E using the field operations $+$, $\times$, the inverse operations and constants from E.

The model of computation is a complete synchronous network of n participants. The broadcast and pairwise communication channels between participants are secure, that is, they cannot be read or tampered with by other participants.

MPC Protocols with a Passive Adversary

A generic MPC protocol consists of the following three phases [4].

1. **Initialization** – Each participant P_i ($i = 1, \ldots, n$) utilizes Shamir's secret sharing scheme [17], and distributes his private input x_i amongst all participants in a t-private manner. More precisely, P_i chooses a random polynomial $f_i(x) = x_i + a_1 x + \ldots, +a_t x^t$, and gives $s_{j,i} = f_i(j)$ to participant P_j for $j = 1, \ldots, n$.
2. **Computation** – Let $s_{i,k}$ and $s_{i,\ell}$ be P_i's shares, associated with polynomials $f_k(x)$ and $f_\ell(x)$ for the secrets x_k and x_ℓ, respectively. The computation of every linear function is straightforward. In order to compute $x_k + x_\ell$, each cooperating participant, P_i, computes $s_i^{k+\ell} = s_{i,k} + s_{i,\ell}$, which is the share of P_i determined by polynomial $h(x) = f_k(x) + f_\ell(x)$. Since $h(x)$ is a polynomial

of degree (at most) t, a set of at least $t+1$ participants who properly follow the protocol can reconstruct the polynomial, and thus retrieve the constant term of the polynomial $h(x)$, which is $x_k + x_\ell$. Similarly, $c \times x_k$, where c is a known scalar, can be computed by $t+1$ participants (each participant P_i calculates $c \times s_{i,k}$ as its share of $c \times x_k$). That is, for $n > t$, there is a non-interactive protocol for computing every linear function $F(x_1, \ldots, x_n)$.
Computation of non-linear functions, however, is not so straightforward. Assume that we want to compute $x_k \times x_\ell$. Although P_i can compute $s_i^{k\times\ell} = s_{i,k} \times s_{i,\ell}$, where $s_i^{k\times\ell}$ is P_i's share of $x_k \times x_\ell$, there are two problems. The first problem is that $s_i^{k\times\ell}$ is the share of P_i determined by the polynomial $h(x) = f_k(x) \times f_\ell(x)$ whose degree is (at most) $2t$. The second problem is that $h(x)$ is not a random polynomial.
Assuming that $n > 2t$, the computation can be carried out if the participants collectively redistribute the constant term of the $h(x)$ polynomial, in a t-private manner, amongst themselves. This process, also called degree reduction, is necessary, otherwise further multiplications will raise the degree, and once the degree of polynomial is equal to or larger than the number of participants in the system, participants will not have a sufficient number of points to perform the necessary interpolation.
Computation of an additive inverse is straightforward. To compute the additive inverse of x_i, every participant P_j computes the additive inverse of his share, $s_{j,i}$. Computing a multiplicative inverse, however, implies cooperation of all shareholders. Catalano, Gennaro, and Halevi [6] have shown how to compute multiplicative inverses.

3. **Reconstruction of the function value** – The function $F(x_1, \ldots, x_n)$ can be represented as a polynomial containing sum of products and the participants can collectively evaluate first products (product gates) and then sums (sum gates) getting finally the shares of the function value Y. In order to reconstruct Y, a set of a sufficiently large set of participants can pool their shares and recover the value Y.

Thus, in the presence of a passive adversary, a set of n participants can compute every *n-variate* function, in a *t-private* manner, as long as $n > 2t$.

MPC Protocols with an Active Adversary

The main challenge in designing MPC protocols in the presence of an active adversary is how to deal with malicious participants. In general, robust MPC protocols first identify misbehaving participants and then disqualify them. Two different disqualification techniques are being used:

1. *Ignoring the information associated with malicious participants* – This strategy is used in MPC protocols that can be completed without utilizing the information coming fom malicious participants.
2. *Reconstructing the information associated with malicious participants* – This strategy is used in MPC protocols that cannot be completed without using

the information owned by corrupted participants. So, after detection of misbehaving participant P_i, other participants reconstruct the private information of P_i, and re-share it amongst themselves.

1.2 Background

In 1987, Goldreich, Micali, and Wigderson [12] gave a solution to the general MPC problem assuming that one-way functions with trapdoor exist (i.e. their protocol is computationally secure). They have shown that in the presence of passive adversaries, every function can be computed by n participants, in such a way that no subset of less than n participants can learn any additional information apart from the function value. They have also shown that if Byzantine faults are allowed (i.e. an active adversary may corrupt some participants and control their behaviour), every function can be computed by n collaborating participants, as long as the majority of participants is honest.

In 1988, Ben-Or, Goldwasser, and Wigderson [4] and Chaum, Crépeau, and Damgård [7], independently studied unconditionally secure MPC protocols. They have shown that:

(a) In the presence of a passive adversary, no set of size $t < n/2$ of participants learns any additional information, other than the function value.
(b) If Byzantine faults are allowed, no set of size $t < n/3$ can learn any additional information or disrupt the protocol.

Relaxing the requirement of perfect security and assuming that broadcast channels exists, Rabin and Ben-Or [16] have shown that in the presence of Byzantine faults, MPC protocols exist if the majority of participants is honest. The privacy achieved is unconditional (with error probability $\epsilon > 0$, which can be exponentially small), and does not rely on any assumption about computational intractability. Beaver [2] utilized the verifiable secret sharing (VSS) scheme of [16] and achieved similar results.

The MPC protocols from [4, 7] are determined for $n \geq 3t + 1$ participants, where up to t of them can be corrupted. The disqualification method used in these protocols simply ignores the misbehaving participants, since at any time, there exists at least $2t + 1$ honest participants who properly follow the protocol.

In contast, the protocols from [16, 2] are defined for $n \geq 2t + 1$ participants. After each multiplication, the polynomial associated with the multiplied shares is of degree at most $2t$. If a malicious participant P_i does not cooperate properly, the remaining participants must reconstruct all information in the hands of P_i, otherwise they cannot interpolate the associated $2t$-degree polynomial. Reconstruction of information in the hands of a participant is possible since the share of each participant is re-shared via the second level of sharing. This procedure, however, reveals one share associated with the secret input of every participant[1]. In these protocols, after detection and elimination of any malicious participant the degree of threshold parameter is decreased by one. So after elimination of

[1] Similar problem in shared generation of digital signatures has been considered in [1].

t malicious participants, the threshold parameter drops to zero (i.e., all private information is disclosed to the remaining participants). One may argue that this is not a security problem, since the remaining participants are assumed to be honest.

Fitzi, Hirt, and Maurer [10] improved the classical results in unconditionally secure MPC by considering a *mixed adversary*. They have shown that in addition to $t_a < n/3$ actively corrupted participants, privacy can be guaranteed against additionally $t_p \leq n/6$ passively corrupted participants. They have also introduced the concept of (t_a, t_p)-secure MPC protocols. In a (t_a, t_p)-secure MPC protocol, correctness of the protocol is guaranteed if up to t_a participants are corrupted actively, and privacy of the participants is ensured if (in addition to t_a actively corrupted participants) up to t_p participants are corrupted passively.

1.3 Motivation

All existing MPC protocols with active adversaries assume that uncorrupted participants are truly honest, i.e., they are not even curious in learning private inputs of others. Based on this assumption, some MPC protocols (e.g. [16] [2], [3], [11], [14], etc.) are designed in such a way that after elimination of all misbehaving participants, the remaining ones learn all private information.

Furthermore, an improvement of the classical results provided by Fitzi, Hirt, and Maurer indicates that in addition to t_a actively corrupted participants, the adversary may simultaneously corrupt some participants passively. This is in contrast to the assumption that participants who are not corrupted by an active adversary are truly honest.

The paper examines the privacy of MPC protocols and argues that the assumption about participants that are not corrupted are truly honest is unrealistic. We will introduce the notion of the *omnipresent adversary*. An omnipresent adversary cannot be eliminated from the protocol and can be either passive, active, or mixed. More precisely, we assume that that up to a minority of participants who are not corrupted by an active adversary can be corrupted passively, with the restriction that at any time, the number of corrupt participants does not exceed a predetermined threshold. We will also show that the existence of a *t-resilient* protocol for a group of n participants, implies the existence of a *t'-private* protocol for a group of n' participants. That is, elimination of misbehaving participants from a *t-resilient* protocol leads to the decomposition of the protocol, and converts it to the *t'-private* protocol.

Our adversary model stipulates that a MPC protocol never operates with a set of truly honest participants (which is a more realistic scenario). Therefore, the privacy of all participants who properly follow the protocol will be maintained. In order to achieve these goals in the existing MPC protocols, we present a novel disqualification protocol that avoids exposing the privacy of participants who properly follow the protocol.

Our results are as follows:

Theorem 1. *Given a set of $n = 2t + 1$ participants in the computationally secure setting, then there exists a t-resilient and t-private MPC protocol provided*

that, at every stage of the protocol, the total number of actively and passively corrupted participants is not larger than t. That is, only one participant may not be corrupted by adversary throughout the execution of the protocol.

Theorem 2. *Given a set of $n = 3t+1$ participants in the unconditionally secure setting with perfect secrecy, then there exists a t-resilient and t-private MPC protocol. That is, up to $2t$ participants may be corrupted by the adversary provided that, at every stage of the protocol, the total number of corrupted participants is not larger than t.*

Theorem 3. *Given a set of $n = 2t + 1$ participants in unconditionally secure setting with a negligible failure probability, then there exists a t-resilient and $t/2$-private MPC protocol. In other words, only $t/2 + 1$ participants may not be corrupted, provided that, at every stage of the protocol, the total number of corrupted participants does not exceed the threshold parameter.*

The rest of this paper is organized as follows. In Section 2, we will give an overview of MPC protocols in the presence of an omnipresent adversary. In Section 3, we will present our approach to disqualification of malicious participants in MPC protocols with an honest majority. In Section 4, we will study the transformation of *t-resilient* protocols into *t'-private* protocols. In Sections 5 and 6 we will show how to modify the existing protocols, in order to simultaneously achieve correctness and privacy. Section 7 gives concluding remarks.

2 An Overview of MPC with Omnipresent Adversary

Designing MPC protocols under an assumption that participants are truly honest is unrealistic. If we could assume that a single trusted party existed, then the designing of MPC protocols would be easy. In this case, all participants first handed their inputs to the trusted party who would compute the function and announce the result to each participant. This scenario, which is known as an *ideal process*, has been studied in order to evaluate the security of *real-life* MPC protocols (see, e.g., [5]). Real-life MPC protocols allow n mutually distrusting participants to evaluate a function for their private inputs assuming that some participants are corrupted. Note that as passively corrupted participants follow the protocol, they cannot be eliminated from it. On the other side, malicious participants deviate from the protocol so they can be identified and eliminated from the protocol.

An omnipresent adversary can be seen as an entity which attempts to break either privacy or correctness of MPC protocols by trying to corrupt (passively or actively) some participants. In the case of passive corruption, the adversary has access to all the information held by the corrupted participant while the participant follows the protocol honestly. In the case of active corruption, the adversary has full control over the behavior of the participant who may deviate from the protocol in an arbitrary way. Observe that if a corrupted participant does not follow the protocol, she can be identified and removed from it.

A good example of an omnipresent adversary is a powerful enemy (such as rogue states, terrorist organizations, intelligence agencies, etc.) who is using its large resources to break MPC protocols by trying to corrupt the participants. Clearly, the adversary is not going to be involved in the protocol directly but it will use the corrupted participants to achieve its goals. From the omnipresent adversary point of view, it would like to achieve its goals with minimum expenses. The expenses are proportional to the number of participants that need to be corrupted. It may also be assumed that a passive corruption may be easier and cheaper than an active corruption.

Our adversarial model is more general and powerful. In particular, it is dynamic so the composition of corrupted participants may change throughout the protocol execution. The number of participants that can be corrupted is larger than the threshold parameter, with the restriction that at any time, the number of corrupted participants does not exceed the threshold parameter. That is, if some actively corrupted participants are detected and eliminated, the adversary is allowed to corrupt some other participants. Strictly speaking, in the computationally secure MPC protocols (e.g. [12]), a *t-resilient* protocol works if $n \geq 2t + 1$. That is, up to t participants can be corrupted actively, and the remaining $t + 1$ participants are assumed to be honest. In our model, up to t participants can be corrupted actively, and up to t participants can be corrupted passively (only one participant may not be corrupted). In the unconditionally secure protocols with perfect secrecy (e.g. [4]), a *t-resilient* protocol works if $n \geq 3t + 1$, where up to $2t + 1$ participants are honest. In our model, up to t participants can be corrupted actively, and up to t participants can be corrupted passively (i.e., there are at least $t+1$ honest participants). Similarly, in majority-honest MPC protocols with small probability of error (e.g., [16, 2]), a *t-resilient* protocol works if $n \geq 2t + 1$, where up to $t + 1$ participants are honest. In our model, up to t participants can be corrupted actively, and up to $t/2$ participants can be corrupted passively, (i.e. there are at least $t/2 + 1$ honest participants).

2.1 Proactive Secret Sharing Scheme

One can see that the number of corrupted participants in the protocol life-time can be greater than the threshold parameter t. If our protocols are implemented using a static secret sharing, then the adversary who learns more than t shares will be able to recover the private information. To prevent this, we employ the well-known *proactive secret sharing* technique [13]. A proactive secret sharing ensures the privacy of a secret by periodically renewing the shares of participants, without changing the secret, in such a way that information gained by an adversary in one time period is useless for the adversary in another time period. In other words, in an ordinary *t-private* secret sharing, its privacy is assured if, throughout the entire life-time of the secret, the adversary is not able to compromise more than t shares. In contrast, for a *t-private* proactive secret sharing, its privacy is guaranteed if at any time period (between two consecutive renewals), the adversary does not compromise more than t shares.

The proactive secret sharing of [13] consists of n participants, where each participant is connected to a common broadcast channel C, where messages sent on C instantly reach each party connected to it. The time is divided into time-periods (e.g. a day, a week, etc.). At the beginning of each time-period the participants update their shares using an interactive share renewal protocol. The adversary can corrupt participants at any moment. If a participant P_j is corrupted during an update phase T_{i+1}, it will be considered as corrupted during both time-periods T_i and T_{i+1}. If the adversary leaves a corrupted participant P_j before the update phase T_{i+1}, then the adversary will not have any control over the communications of P_j, and thus has no information about the updated shares of P_j (i.e. P_j is no longer corrupted). The underlying secret sharing scheme is the Shamir [17] threshold scheme. The number of participants is $n = 2t + 1$, where during each time-period, the adversary can corrupt up to t participants. Assume that a secret x_i is shared amongst the set of n participants, in a *t-private* manner. In the update phase, each participant P_j distributes $s_{j,i}\Pi_{i=1,i\neq j}^{n}\frac{j}{j-i}$ amongst all participants in a *t-private* manner. Each participant P_ℓ adds all new shares received during the update phase, and takes it as his share of the secret x_i, and deletes the old share $s_{\ell,i}$ plus all partial shares. This process is correct, because $x_i = \Sigma_{j=1}^{n} s_{j,i}\Pi_{i=1,i\neq j}^{n}\frac{j}{j-i}$. The verifiable secret sharing (VSS) used in [13] is computationally secure and based on the Feldman VSS [9]. However, unconditional security is achievable by utilizing the Pedersen VSS from [15].

There are some differences between the proactivization used in [13], and in the MPC protocols with omnipresent adversary. They are as follows.

(a) The purpose of the update phase in [13] is to correct the shares of the participants that have been corrupted by an active adversary or alternatively by errors caused by other problems such as system crashes, for instance. Note that for the randomization and degree-reduction, we do not correct the shares of corrupted participants. Instead, we identify the corrupted participants and eliminate them from the protocol.
(b) The update phase of [13] is performed at the beginning of each time-period, while in our MPC protocols, it is done at the randomization and degree reduction stage.
(c) Similarly to [13], we allow the adversary to leave some corrupted participants alone until the execution of the randomization and degree reduction protocols. After proactivization, actively corrupted participants are eliminated from the protocol, while passively corrupted participants who have not been controlled by the adversary will have new shares that are not known to the adversary, and therefore they are not considered as corrupted participants any more. Now, the adversary may wish to corrupt some new participants (from the set of all remaining participants). That is, in our MPC protocols, the set of corrupted participants is dynamic.

It is worth mentioning that the proactivization process will not add too much overhead to our MPC protocols. This is due to the fact that in MPC protocols, the degree reduction procedure is, indeed, a proactivization of the participant shares. The only information that needs to be re-shared is the share of each

participant from the other participant secret information. We observe that this is also done in MPC protocols with an honest majority, since after the threshold parameter is decreased, all information is re-shared using the new threshold. The overhead applies only to MPC protocols with fixed threshold parameter.

3 Disqualification in MPC with Honest Majority

Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be a set of n participants who wish to compute a function $Y = F(x_1, \ldots, x_n)$, where participants P_i hold their private inputs x_i, assuming that $n = 2t + 1$, and the initial threshold parameter is t. Disqualification of a malicious participant P_i requires the reconstruction of the information in the hand of P_i, otherwise the protocol cannot be completed. The elimination of malicious participants, however, has to reduce the threshold parameter, otherwise the current number of participants cannot interpolate the polynomial associated with their shared information. In the existing MPC protocols, elimination of each malicious participant decreases the threshold parameter by one. After elimination of t malicious participants, the threshold parameter becomes zero, i.e., the remaining participants learn all private information.

In this section we will present a new disqualification technique that preserves the privacy of all participants who properly follow the protocol. Let D denote the number of eliminated participants from the system (initially, $D = 0$). After detecting a malicious participant, say P_i $(1 \leq i \leq n)$, increase the value of D by one and perform the following steps.

1. If D is an odd integer, private information of P_i is reconstructed by the other participants. If this occurs in the initialization phase, no further action is required. If this occurs in the computation phase, relevant computations (i.e. multiplication of relevant shares, randomization, and degree reduction procedure) associated with P_i will be performed publicly. This process reveals the private input x_i (which is not an issue, since it is a random value), and one share associated with the private input of each participants who properly follow the protocol. After the threshold parameter is reduced by one and all values are re-shared (see the next item), the knowledge of these shares is redundant.
2. If D is an even integer, only the secret input x_i is reconstructed, and the threshold parameter is decreased by 1. If this occurs in the initialization phase, the remaining participants repeat the initialization phase using a threshold parameter $t' = t - 1$. If this occurs in the computation phase, the remaining participants re-share their partial results using a new threshold parameter $t' = t - 1$, and continue the protocol using this new threshold parameter t'. Implicitly, this is a proactivization of a (t, n)-threshold scheme to a $(t - 1, n - 2)$-threshold scheme.

As the result of applying our disqualification technique, after occurrence of t faults, the number of remaining participants in the system is $t + 1$, and the

threshold parameter is $t' = t - t/2 = t/2$. Therefore, no subset of up to $t/2$ participants learn any additional information about the secret input of participants who have properly followed the protocol.

4 Decomposition of *t-resilient* protocols

According to the definition of *t-resilient* protocols, after elimination of all malicious participants, the remaining participants must be able to complete the protocol. In other words, in the absence of all eliminated participants (even if they voluntarily withdrew from the protocol), the remaining participants must be able to complete the protocol.

Theorem 4. *Let a t-resilient MPC protocol π realizes task T for a group of n participants, then there exists a t'-private MPC protocol π' that realizes task T for a group of n' participants, where $n' \geq n - t$ and $t/2 \leq t' \leq t$.*

Proof. If no participant misbehaves, the *t-resilient* MPC protocol π realizes task T in a secure manner (they obtain the correct result, where privacy of all inputs is maintained in a *t-private* manner). That is, a *t-resilient* MPC protocol is necessarily a *t-private* MPC protocol. Note that the inverse statement is not true. Having a *t-private* MPC protocol that realizes a task T, does not imply that we can design a *t-resilient* MPC protocol for the task. If all (or some) of the malicious participants are eliminated from the protocol, the remaining participants must be able to complete the protocol. Completion of the protocol means that remaining participants continue to perform the protocol π. If all malicious participants are eliminated, the number of remaining participants will be at most $n' = n - t$ (since at most t participants are eliminated). That is, protocol π continues with n' participants, where no fault occurs. In this case, π is not necessarily a *t-private* protocol, since the threshold parameter may have been reduced to t', where $t/2 \leq t' \leq t$. This completes the proof, assuming that protocol π' is a version of protocol π in which the verification procedures are omitted.

Indeed, a common practice in designing a *t-resilient* protocol is to equip a *t'-private* protocol with mechanisms that can manage malicious participants. That is, a *t-resilient* protocol can be decomposed into two phases, namely, detection and elimination of malicious participants and then the execution of a *t'-private* protocol π' for a group of n' participants.

For example, consider computationally secure MPC protocols given by Goldreich et al. in [12]. They proved that there exists a *t-resilient* MPC protocol for a group of at least $2t+1$ participants, and there exists a *t-private* MPC protocol for a group of at least $n' = t + 1$ participants. In other words, their *t-resilient* protocol can be decomposed into two sub-protocols, namely, detection of malicious participants and running their *t-private* protocol for the participants that honestly follow the protocol.

In an unconditionally secure setting, the protocols studied in [4] and [7] indicate that their *t-resilient* MPC protocol works for a group of at least $3t + 1$ participants, and their *t-private* protocol works for a group of at least $2t + 1$ participants. That is, their *t-resilient* protocols can be decomposed in similar way, first detection and elimination of malicious participants and next the execution of a *t-private* protocol.

In MPC protocols with honest majority (alternatively, with faulty minority), their *t-resilient* MPC protocol works with a group of at least $2t+1$ participants. Although they have not discussed the case of passive adversary, classical results indicate that a *t-private* requires at least $2t + 1$ cooperating participants. However, in their *t-resilient* MPC protocol with at least $2t + 1$ participants, after elimination of t misbehaving participants, the remaining number of participants is $t + 1$. That is, decomposition of their *t-resilient* protocol gives a *t/2-private* protocol.

5 Perfectly Secure MPC with Omnipresent Adversary

Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be a set of n participants who wish to compute a function $Y = F(x_1, \ldots, x_n)$, where each participant P_i holds her private input x_i. To construct a perfectly secure MPC with an omnipresent adversary, we employ the MPC protocol from [4]. In the case of passively corrupted participants, if $n \geq 2t+1$, their *t-private* MPC protocol provides perfect privacy. In the case of actively corrupted (malicious) participants, if $n \geq 3t + 1$, their *t-resilient* MPC protocol provides perfect privacy, assuming that up to t participants can be malicious and other participants honestly follow the protocol. In their *t-resilient* protocol with $3t + 1$ participants, after elimination of all misbehaving participants, there will be $2t+1$ remaining participants in the system, which is large enough to construct a *t-private* protocol. That is, decomposition of their *t-resilient* protocol should lead to a *t-private* protocol. Since the threshold parameter is fixed, the number of corrupt participants at any time must not exceed the threshold parameter t. That is, we modify their disqualification procedure as follows (the rest of the protocol remains unchanged):

1. After the detection and elimination of a malicious participant P_i (for detail procedure, see [4]), the remaining participants perform proactivization of their shares from the other participant secret inputs.
2. After proactivization, the adversary is allowed to corrupt a new participant, either passively or actively, subject to the condition that the number of actively corrupted participants in the life-time of the protocol does not exceed t.
3. The remaining participants continue the protocol as in [4]. After elimination of at most t malicious participants, the system consists of at least $2t + 1$ participants, where up to t participants are corrupted passively. Classical results indicate that an unconditionally secure *t-private* MPC protocol exists for this set of participants. That is, a *t-resilient* protocol is converted to a *t-private* protocol.

Remark 1. In computationally secure MPC, the *t-resilient* protocol from [12] starts with $2t + 1$ participants. Performing their protocol in the presence of an omnipresent adversary, and utilizing the above elimination technique, after elimination of t malicious participants, the system consists of $t + 1$ participants, where t of them are corrupted passively. Results of [12] indicate that a computationally secure *t-private* MPC protocol exists for this set of participants. That is, a *t-resilient* protocol is converted to a *t-private* protocol.

5.1 Security Discussion

This modified protocol is as secure as the original MPC protocol from [4]. This is because the adversary cannot learn any additional information (due to proactivization, information obtained in one time period is useless for another time period, and at each time period the scheme is *t-private*). Also, correctness of the result will not be affected, because up to t additional passively corrupted participants honestly follow the protocol.

Moreover, if t_a and t_p denote the number of actively and passively corrupted participants at any time period, the following conditions always hold:

(a) $3t_a + t_p < n$.
Considering the fact that $t_a \leq t$ and after elimination of any misbehaving participant, one participant will be corrupted passively, if k $(0 \leq k \leq t)$ participants are eliminated, $3t_a + t_p \leq 3(t - k) + k = 3t - 2k \leq n - k$.

(b) $2t_a + 2t_p < n$.
Similarly, $2t_a + 2t_p \leq 2(t - k) + 2(k) = 2t < n - k$.

That is, at every stage, our protocol satisfies the results of [10].

Theorem 5. *Given an MPC protocol defined in [4]. A set of n participants can compute every function perfectly (t_a, t_p)-securely if and only if $3t_a + t_p < n$ and $2t_a + 2t_p < n$. The computation is polynomial in n and linear in the size of the circuit. This holds whether a broadcast channel is available or not.*

6 Honest Majority MPC with Omnipresent Adversary

In MPC protocols with honest majority, a set of $\mathcal{P} = \{P_1, \ldots, P_n\}$ participants wish to compute a function $Y = F(x_1, \ldots, x_n)$, where each participant P_i holds her private input x_i. Although existing MPC with honest majority do not consider the case of a passive adversary, the condition $n \geq 2t + 1$ is the tight bound for designing a *t-private* MPC protocol, even if a broadcast channel is available. In the case of an active adversary, assuming that a public channel exists, a *t-resilient* MPC is achievable if $n \geq 2t + 1$. The secrecy of these protocols is unconditional, with error probability ϵ, which can be exponentially small.

To construct an MPC with honest majority in the presence of an omnipresent adversary, we employ the protocol from [16]. However, we utilize our disqualification technique (see Section 3), that ensures the privacy of participants who properly follow the protocol. So, our construction for MPC with honest majority works as follows:

1. After the detection and elimination of every two misbehaving participants, the threshold parameter is reduced by one (see our disqualification in Section 3). The process of re-sharing all information with a new threshold parameter $t' = t - 1$, implicitly, is a proactivization of the shares associated with the function value and the participant inputs.
2. After reducing the threshold, the adversary is allowed to corrupt a new participant, either in passive or active mode, provided that the number of actively corrupted participants in the life-time of the protocol does not exceed t.
3. The remaining participants continue the protocol as in [16], using the new threshold t'. After elimination of at most t misbehaving participants, the system consists of at least $t + 1$ participants, where up to $t/2$ participants are corrupted passively. Classical results indicate that an unconditionally secure *t/2-private* MPC protocol exists for this set of participants. That is, a *t-resilient* protocol is converted to a *t/2-private* protocol.

We observe that the "honest majority" (alternatively "faulty minority") title is more suitable for our protocol (see above), since at every stage of the protocol, *only* the minority/majority of participants in the system are corrupt/honest. While in [16, 2], after elimination of corrupt players, all remaining participants are honest.

6.1 Security Discussion

The modified protocol is as secure as the original MPC protocol from [16]. This is true because the adversary cannot learn any additional information since at each time period the scheme is *t'-private*. Also, correctness of the result will not be affected, because up to $t/2$ additional passively corrupted participants honestly follow the protocol.

Moreover, if t_a and t_p denote the number of actively and passively corrupted participants at any time period, the condition $2t_a + 2t_p < n$, is satisfied. This is because $2t_a + 2t_p \leq 2(t - k) + 2k/2 \leq 2t - k < n - k$. So, at every stage, our protocol satisfies the results of [10].

Theorem 6. *Given the MPC protocol defined in [16]. Then allowing an negligible failure probability and given a broadcast channel, a set of n participants 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.*

7 Conclusions

We have investigated the privacy of MPC protocols in the presence of omnipresent adversary. The omnipresent adversary can be either passive, active, or mixed. We have shown that up to a minority of participants who are not corrupted actively, can be corrupted passively, with the restriction that at any time, the number of corrupt participants does not exceed a predetermined threshold.

Our adversary model stipulates that MPC protocols never run with a set of truly honest participants (which is a more realistic assumption). Therefore, privacy of all participants who properly follow the protocol will be maintained.

Table 1. Comparison of existing *t-resilient* MPC protocols and MPC protocols with an omnipresent adversary.

Security Model	Adversary model	Number of participants	Actively corrupted	Passively corrupted	Uncorrupted participants
Computational	Active	$n = 2t + 1$	t	0	$t + 1$
	Omnipresent	$n = 2t + 1$	t	t	1
Unconditional without broadcast channel	Active	$n = 3t + 1$	t	0	$2t + 1$
	Omnipresent	$n = 3t + 1$	t	t	$t + 1$
Unconditional with broadcast channel	Active	$n = 2t + 1$	t	0	$t + 1$
	Omnipresent	$n = 2t + 1$	t	$t/2$	$t/2 + 1$

Remark 2. In this paper we have used the protocols from [4] and [16] (that are perfectly secure with a negligible failure probability) and showed how the omnipresent adversary works for these protocols. For the perfect security case, the maximum number of corrupted participants at any time is t, and for MPC protocols with a negligible failure probability, the maximum number of corrupted participants at any time is t' ($t/2 \leq t' \leq t$), see Table 1. Applying the protocol from [10] improves these bounds, but will not increase the total number of participants that can be corrupted in the protocol life-time. This is because, in the presence of a passive adversary, $n > 2t + 1$ is a tight bound regardless of a type of the protocol.

Acknowledgments

We are grateful to the anonymous referees for their constructive comments. The second co-author was supported by Australian Research Council grant DP0663452.

References

1. J. Almansa, I. Damgård, and J. Nielsen, "Simplified Threshold RSA with Adaptive and Proactive Security," in *Advances in Cryptology - Proceedings of EUROCRYPT 2006* (S. Vaudenay, ed.), vol. 4004 of *Lecture Notes in Computer Science*, pp. 593–611, Springer-Verlag, 2006.
2. D. Beaver, "Multiparty Protocols Tolerating Half Faulty Processors," in *Advances in Cryptology - Proceedings of CRYPTO'89* (G. Brassard, ed.), vol. 435 of *Lecture Notes in Computer Science*, pp. 560–572, Springer-Verlag, 1990.
3. D. Beaver, "Secure Multiparty Protocols and Zero-Knowledge Proof Systems Tolerating a Faulty Minority," *Journal of Cryptology*, vol. 4, pp. 75–122, 1991.

4. M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness Theorem for Non-Cryptographic Fault-Tolerant Distributed Computation," in *Proceedings of the 20th ACM Annual Symposium on the Theory of Computing (STOC'88)*, pp. 1–10, 1988.
5. R. Canetti, "Security and Composition of Multiparty Cryptographic Protocols," *Journal of Cryptology*, vol. 13, pp. 143–202, 2000.
6. D. Catalano, R. Gennaro, and S. Halevi, "Computing Inverses over a Shared Secret Modulus," in *Advances in Cryptology - Proceedings of EUROCRYPT 2000* (B. Preneel, ed.), vol. 1807 of *Lecture Notes in Computer Science*, pp. 190–206, Springer-Verlag, 2000.
7. D. Chaum, C. Crépeau, and I. Damgård, "Multiparty Unconditionally Secure Protocols," in *Proceedings of the 20th ACM Annual Symposium on the Theory of Computing (STOC'88)*, pp. 11–19, 1988.
8. D. Dolev, C. Dwork, O. Waarta, and M. Yung, "Perfectly Secure Message Transmission," *Journal of the ACM*, vol. 40, no. 1, pp. 17–47, 1993.
9. P. Feldman, "A Practical Scheme for Non-interactive Verifiable Secret Sharing," in 28^{th} *IEEE Symposium on Foundations of Computer Science*, pp. 427–437, oct 1987.
10. M. Fitzi, M. Hirt, and U. Maurer, "Trading Correctness for Privacy in Unconditional Multi-Party Computation," in *Advances in Cryptology - Proceedings of CRYPTO'98* (H. Krawczyk, ed.), vol. 1462 of *Lecture Notes in Computer Science*, pp. 121–136, Springer-Verlag, 1998.
11. R. Gennaro, M. Rabin, and T. Rabin, "Simplified VSS and Fast-track Multiparty Computations with Applications to Threshold Cryptography," in *17th Annual ACM Symposium on Principles of Distributed Computing*, pp. 101–111, 1998.
12. O. Goldreich, S. Micali, and A. Wigderson, "How to Play any Mental Game," in *Proceedings of the 19th ACM Annual Symposium on the Theory of Computing (STOC'87)*, pp. 218–229, May 25–27, 1987.
13. A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung, "Proactive Secret Sharing Or: How to Cope With Perpetual Leakage," in *Advances in Cryptology - Proceedings of CRYPTO'95* (D. Coppersmith, ed.), vol. 963 of *Lecture Notes in Computer Science*, pp. 339–352, Springer-Verlag, 1995.
14. M. Hirt, U. Maurer, and B. Przydatek, "Efficient Secure Multi-party Computation," in *Advances in Cryptology - Proceedings of ASIACRYPT 2000* (T. Okamoto, ed.), vol. 1976 of *Lecture Notes in Computer Science*, pp. 143–161, Springer-Verlag, 2000.
15. T. Pedersen, "Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing," in *Advances in Cryptology - Proceedings of CRYPTO'91* (J. Feigenbaum, ed.), vol. 576 of *Lecture Notes in Computer Science*, pp. 129–140, Springer-Verlag, 1992.
16. T. Rabin and M. Ben-Or, "Verifiable Secret Sharing and Multiparty Protocols with Honest Majority," in *Proceedings of the 21th ACM Annual Symposium on the Theory of Computing (STOC'89)*, pp. 73–85, 1989.
17. A. Shamir, "How to Share a Secret," *Communications of the ACM*, vol. 22, pp. 612–613, Nov. 1979.
18. A. Yao, "Protocols for Secure Computations," in *the 23rd IEEE Symposium on the Foundations of Computer Science*, pp. 160–164, 1982.