Towards a Theory of Consistency Primitives

Extended Abstract

Ueli Maurer[⋆]

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

Abstract. One of the classical results in the theory of distributed systems is the theorem by Lamport, Shostak, and Pease stating that among n parties, any t of which may be cheaters, one of the parties (the sender) can consistently broadcast a value to the other parties if and only if $t \leq n/3$. This is achieved by use of a protocol among the players, using bilateral channels.

The purpose of this paper is to look at various generalizations of this result and to propose a new concept, called consistency specification, a very general type of consistency guarantee a protocol among n parties $P_1, \ldots, P_n$ can provide. A consistency specification specifies, for every possible set $H \subseteq \{P_1, \ldots, P_n\}$ of honest players and for every choice of their inputs, a certain security guarantee, i.e., a consistency condition on their outputs. This models that security can degrade smoothly with an increasing number of cheaters rather than abruptly when a certain threshold is exceeded, as is the case in the previous literature.

1 Introduction

1.1 Security in Distributed Systems

Distributed systems generally involve a number of entities (for instance called servers, parties, or players), connected by some communication channels, who are supposed to perform a certain task, using a well-defined protocol between the entities. The task can range from (apparently) simple problems like the synchronization of the entities' clocks to highly complex tasks like a distributed on-line auction among the entities. Typically, performing this task is in their mutual interest, but the entities cannot be assumed to perform the protocol correctly since some of them might gain an advantage by cheating.

A key problem in the design of protocols for distributed systems, and more generally for cryptographic applications, is to achieve security against the malicious behavior of some of the involved parties. In other words, there must be a

[⋆] Supported in part by the Swiss National Science Foundation.

certain security guarantee for the honest parties, even if the remaining parties cheat. Security can mean secrecy of certain data (e.g. the parties' inputs) and/or correctness of the parties' outputs.

1.2 The General Setting

Throughout the paper, let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be the set of n parties, also called players. Each player can either be honest or malicious. An honest player performs the prescribed protocol, whereas a malicious player can deviate arbitrarily from the protocol.[1] In order to model the coordinated cheating by several players, one often considers a central adversary who can corrupt some of the players and take full control of their actions. The non-corrupted players are called *honest*. An adversary is hence characterized by the set of corrupted players.

Typical results in the literature on secure distributed protocols provide a security guarantee as long as the number of cheating parties is bounded by some threshold t, accepting complete failure when this bound is exceeded. This threshold-type model can be generalized in many ways. In particular, we propose a more general concept of security where for each set H of honest players (or, equivalently, for any set of cheating players), the achieved level of security is specified. Of course, the achieved security level degrades monotonically when more parties cheat, but it can degrade smoothly rather than abruptly with an increasing set of cheating parties.

1.3 The Broadcast Problem

A classical problem in distributed systems is for one player $P_i \in \mathcal{P}$ (the sender) to broadcast a value (from a certain domain, e.g. $\{0, 1\}$) consistently to the other players, using only (authenticated) bilateral communication channels, with the following security requirements:

1. *Consistency:* All honest players decide on the same value v.
2. *Correctness:* If P_i is honest, then v is the value sent by P_i.

A third requirement, often tacitly assumed, is that the protocol must terminate.

The broadcast problem is a prototypical problem in which a protocol among the players guarantees some form of consistency of their output values. It is used as a subprotocol in many other secure distributed protocols, in particular also in secure multi-party computation (see Section 1.4).

A different variant of the broadcast problem is the so-called *consensus* problem, where each player has an input and the goal is that the players decide on the same value which, if the majority of the players entered the same value, must be equal to this value.

[1] More generally, there could be several levels of cheating. For example, in secure multi-party computation one often considers *passive* cheating which mans that such a cheater performs the protocol correctly but might pool his information with other cheaters to violate secrecy.

One of the most classical results in the theory of distributed systems is the theorem by Lamport, Shostak, and Pease [LSP82], stating that broadcast and consensus among n parties is possible if and only if strictly fewer than $n/3$ of the players cheat (i.e., $t < n/3$).

One can consider more general consistency guarantees for the output values of the honest players, not only the special type of guarantee that they be equal. To explore this space of possible consistency guarantees is the subject of this paper.

1.4 Secure Multi-Party Computation

The purpose of this paper is to discuss consistency at a very general level, but to discard secrecy requirements relevant in a more general type of secure cooperation of parties. The reasons for restricting ourselves to consistency in this paper are that the definitions are simpler and cleaner, that understanding consistency appears to be an important goal on the way to understanding more general settings, and that consistency primitives are important building blocks in protocols achieving more general security goals.

But it is worthwhile to briefly discuss the more general setting because it sheds light on how one can view results on consistency and because many results in the literature have been derived for this more general setting (and hence also apply in our consistency setting).

One can view a broadcast protocol as the simulation of a trusted party whose task it is to accept an input value from the sender and to send this value to each other player. The simulation works correctly under the condition $t < n/3$. More generally, one can simulate a trusted party who performs more complicated tasks, including the secret storage of information and the computation of functions on the players' inputs. Broadcast is a special case, involving no secrecy requirement.

The general paradigm of simulating a trusted party is called *secure multi-party computation*. A typical example is a voting protocol which allows a set of voters to add their votes correctly while nevertheless keeping their votes secret from the other players, even if some of them cheat. In this case the simulated trusted party takes as inputs all votes of the players, adds them up, and sends the result to each player.

Some classical results on secure multi-party computation are as follows. Goldreich, Micali, and Wigderson [GMW87] proved that, based on cryptographic intractability assumptions, secure multi-party computation is possible if and only if the number t of corrupted players satisfies $t < n/2$. In the information-theoretic model, where bilateral secure channels between every pair of players are assumed and the adversary is assumed to be computationally unrestricted, Ben-Or, Goldwasser, and Wigderson [BGW88] (and independently Chaum, Crépeau, and Damgård [CCD88]) proved that perfect security is possible if and only if $t < n/3$. In a model in which a broadcast channel among the players is assumed, unconditional security (with exponentially small failure probability) is achievable if and only if $t < n/2$ [RB89]. We refer to [Mau02] for a discussion of known results on secure multi-party computation and for a conceptually very simple protocol.

Let us briefly discuss the relation between the two results for threshold $t < n/3$ mentioned above. The result on broadcast [LSP82] is a special case of the result on secure multi-party computation [BGW88], but a broadcast protocol tolerating up to $n/3$ cheaters (and hence the Lamport-Shostak-Pease theorem) is used as a crucial primitive within the secure multi-party computation protocol of [BGW88]. What is surprising is that the same threshold as for broadcast applies for a much more general problem.

1.5 Scope and Limitations of this Paper

In this paper we consider only consistency but not secrecy requirements. In terms of the simulation of a trusted party discussed above, this means that the simulated trusted party is allowed to be transparent in the sense that an adversary can learn the entire internal information. (But it is not transparent for the honest parties.)

The purpose of this extended abstract is to briefly review several types of recently proposed generalizations of the classical theorem by Lamport, Shostak, and Pease, and then to propose a new generalization. The goal of this generalization is to allow for the security (i.e., consistency) guarantee to degrade slowly with an increasing number of cheaters rather than abruptly and completely when a certain threshold is exceeded, as is the case in the previous literature. In other words, for every possible set of players the adversary might corrupt, there is a certain security (consistency) guarantee.

This extended abstract introduces the general concept of a consistency specification, gives a few examples, and discusses the reduction of one consistency specification to some other consistency specifications by use of a protocol. This calls for a general theory explaining which reductions are possible and which are not. However, such a theory is not developed in this paper.

We consider a synchronous setting in which the players are assumed to have synchronized clocks. Moreover, we consider a setting with zero failure probability. Both these points could be generalized to an asynchronous setting and to tolerating a small failure probability.

2 Generalizations of the Classical Results

As mentioned above, results for broadcast and for secure multi-party computation among n players are usually derived for a setting with at most t corrupted players, with no security guarantee if this threshold is exceeded.

This model and result can be generalized in various ways, discussed in this section, both for the case of general secure multi-party computation as well as for the case of broadcast and more general consistency protocols.

2.1 From Thresholds to General Adversary Structures

A first generalization was proposed in [HM97] where the adversary's corruption capability is modeled by a general so-called *adversary structure* rather than a threshold t.

Definition 1. Consider a finite set $\mathcal{P}$. We call a subset Π of the power set $2^{\mathcal{P}}$ of $\mathcal{P}$ a *(monotone) structure* for $\mathcal{P}$ if Π is closed under taking subsets, i.e.,

$$S \in \Pi \ \wedge \ S' \subseteq S \quad \Longrightarrow \quad S' \in \Pi.$$

Example 1. The most common example of a structure is the threshold structure $\Pi = \{S : S \subseteq \mathcal{P}, |S| \leq t\}$ for some t.

Informally, a protocol is secure against an adversary structure Π if it remains secure as long as the set of players corrupted by the adversary is within Π.

Definition 2. *Let $\sqcup$ be a (commutative and associative) operation on structures, defined as follows: $\Pi_1 \sqcup \Pi_2$ is the structure consisting of all unions of one element of Π_1 and one element of Π_2, i.e.,*

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

It was proved in [HM97] that secure multi-party computation in the information-theoretic setting for a general adversary structure Π is possible if and only if no three sets in Π cover the full player set, i.e., if and only if

$$\mathcal{P} \notin \Pi \sqcup \Pi \sqcup \Pi.$$

This is a strict generalization of the threshold condition ($t < n/3$).

Example 2. For instance, in the case of $n = 6$ players, with $\mathcal{P} = \{P_1, P_2, P_3, P_4, P_5, P_6\}$, one can obtain a protocol secure against the structure

$$\Pi = \{\{P_1\}, \{P_2, P_4\}, \{P_2, P_5, P_6\}, \{P_3, P_5\}, \{P_3, P_6\}, \{P_4, P_5, P_6\}\},$$

whereas in the threshold model one can tolerate only a single cheater, i.e., the adversary structure $\Pi' = \{\{P_1\}, \{P_2\}, \{P_3\}, \{P_4\}, \{P_5\}, \{P_6\}\}$.

The above mentioned result also implies the same generalization for broadcast, but the resulting protocol would not be efficient if there is no short description of the structure Π. An efficient secure broadcast protocol for all adversary structures satisfying $\mathcal{P} \notin \Pi \sqcup \Pi \sqcup \Pi$ was given in [FM98].

2.2 Available Primitives

In most secure distributed protocols in the literature (e.g. for the broadcast result of [LSP82]) it is assumed that authenticated bilateral channels between any pair of players are available. More generally, one can assume other primitives as being available. Such primitives can be stronger than authenticated channels, or they can be weaker when authenticated channels are not assumed to be available. A general problem is to understand reductions between such primitives, i.e., to construct a stronger primitive by application of a protocol involving the given weaker primitives. A natural interesting question is which reductions are possible, and if so, by how many invocations of the given primitives.

For example, the Lamport-Shostak-Pease result states that a broadcast channel among n players can be reduced to authenticated bilateral channels under the condition that the number of cheaters is less than $n/3$. As a more general example, a setting is considered in [FM00] where a broadcast channel is available for any three players, allowing a sender to consistently send a value to (any) two receivers. Under this stronger assumption about the available communication primitive one can prove that broadcast among n players is possible if and only if $t < n/2$, improving on the $t < n/3$ threshold. One can also consider broadcast channels among more than three players as being given, in which case the threshold can be increased even beyond $n/2$ (see [CFF+04]).

2.3 Smooth Security Degradation

Typical results in the literature on secure distributed protocols provide a security guarantee only as long as the set of cheating parties is restricted (by a threshold or a general adversary structure), accepting complete failure when more players are corrupted.

However, this approach ignores the possibility that even if the set of cheaters is outside the adversary structure, there can still be some remaining security guarantee for the honest players, for instance that most (rather than all) honest players receive the same value [KS01]. One may even be willing to sacrifice some security guarantee below a given threshold for the sake of being able to achieve some level of security above the threshold.

In full generality, one need not even consider a threshold or an adversary structure. To fully characterize a primitive, one must specify the achieved security guarantee for every set of honest players. In other words, one must specify for every potential set of corrupted players what they can achieve in the worst case.

In the next section we consider the formalization of such primitives, referring to them as consistency specifications.

3 Consistency Specifications

3.1 Definition of Consistency Specifications

Let again $\mathcal{P} = \{P_1, \ldots, P_n\}$ be the set of players. For convenience, we sometimes also use i instead of P_i. We investigate protocols in which every player P_i has an input from a finite input domain $\mathcal{D}_i$ and receives an output from a finite output domain $\mathcal{R}_i$. The special case where some players have no input and/or no output can be modeled by letting the corresponding input and/or output domain be a singleton set with a default symbol (e.g. $\perp$). For example, for a broadcast channel only one player (the sender) has an input.

Let $\mathcal{D}_\mathcal{P} := \bigtimes_{i=1}^{n} \mathcal{D}_i$ be the Cartesian product of the input domains, and for a player set $S \subseteq \mathcal{P}$, let

$$\mathcal{D}_S := \bigtimes_{i \in S} \mathcal{D}_i$$

be the Cartesian product of the $\mathcal{D}_i$ with $i \in S$. For $S' \subseteq S$ and $\boldsymbol{x} \in \mathcal{D}_S$, let $\boldsymbol{x}_{|S'} \in \mathcal{D}_{S'}$ be the list $\boldsymbol{x}$ restricted to indices in S'. Similarly, for a set $L \subseteq \mathcal{D}_S$ of lists, let $L_{|S'} \subseteq \mathcal{D}_{S'}$ be the set of lists resulting by restricting each of them to S'. Analogous notation is used for the output domains $\mathcal{R}_i$. We also write $\boldsymbol{x}_{|i}$ instead of $\boldsymbol{x}_{|\{P_i\}}$ to denote the entry in $\boldsymbol{x}$ for player P_i.

The following definition captures the most general type of consistency guarantee that a protocol or primitive can give, if the failure probability is required to be zero. It was first proposed by the author and then investigated in [Asc01] and [KS01]. Special cases were then investigated in [FGH$^+$02] and [FHHW03].

Definition 3. A *consistency specification* $\mathcal{C}$ for player set $\mathcal{P} = \{P_1, \ldots, P_n\}$, input domains $\mathcal{D}_1, \ldots, \mathcal{D}_n$, and output domains $\mathcal{R}_1, \ldots, \mathcal{R}_n$ is a function assigning to every set $H \subseteq \mathcal{P}$ (the honest players) and every list $\boldsymbol{x}_H \in \mathcal{D}_H$ (their input values) a set $\mathcal{C}(H, \boldsymbol{x}_H) \subseteq \mathcal{R}_H$ (of lists of possible output values), satisfying the following monotonicity constraint: For any H' and H with $H' \subseteq H$,

$$\mathcal{C}(H, \boldsymbol{x}_H)_{|H'} \subseteq \mathcal{C}(H', \boldsymbol{x}_{H|H'}).$$

Let us explain this definition in more detail. For every set H of honest players (where the remaining players $\mathcal{P} \setminus H$ are assumed to be potentially cheating), and for every choice of inputs of these honest players, a consistency condition satisfied by the outputs of these players is specified. Such a condition excludes certain combinations of output values or, equivalently, specifies the set $\mathcal{C}(H, \boldsymbol{x}_H)$ of admissible lists of output values. For any honest set H it is only guaranteed that the outputs of the players in H form some list in $\mathcal{C}(H, \boldsymbol{x}_H)$, with no further guarantee as to which of these lists is chosen. The smaller the set $\mathcal{C}(H, \boldsymbol{x}_H)$, the stronger is the consistency guarantee of the specification.

The monotonicity condition states that if one considers two settings with honest set H' and a larger honest set H, respectively, then the consistency guarantee for the players in H' cannot become worse by having more honest players. This is justified by the fact that in a (typical) protocol context, one of the options of a cheating player is to behave as if he were honest.

There are two ways of looking at consistency specifications. On one hand, one can assume one or several primitives satisfying certain consistency specifications as being given, defining a specific communication setting. A very common communication setting is the assumed availability of bilateral communication channels. On the other hand, one can consider a consistency specification as the goal of a protocol construction.

3.2 Some Examples

We give a few examples of consistency specifications.

Example 3. The weakest possible consistency specification is when no consistency guarantee is given, i.e., when $\mathcal{C}(H, \boldsymbol{x}_H) = \mathcal{R}_H$ for all H and $\boldsymbol{x}_H \in \mathcal{D}_H$. Obviously, a primitive satisfying only this trivial specification is useless.

Example 4. A bilateral authenticated channel from P_i to P_j, denoted $\textsc{Auth}_{i,j}$, is a consistency specification (for the n players under consideration), where only P_i has an input (i.e., $\mathcal{R}_k = \{\perp\}$ for $k \neq i$) and where the only consistency constraint on the outputs is that P_j's output is equal to P_i's input, for any set H containing P_i and P_j. No guarantee is given for the honest players' outputs except for P_j (if he is honest). More formally, we have

$$\textsc{Auth}_{i,j}(H, \boldsymbol{x}_H) = \left\{ \boldsymbol{y} \in \mathcal{R}_H \;\middle|\; i \in H \Rightarrow \boldsymbol{y}_{|j} = \boldsymbol{x}_{H|i} \right\}$$

for all $H \subseteq \mathcal{P}$.

Let $\textsc{Auth}$ denote the set of consistency specifications consisting of all $n(n-1)$ bilateral authenticated channels from any player to any other player. Note that $\textsc{Auth}$ could also be interpreted as a single consistency specification with a separate selection input for choosing the particular channel $\textsc{Auth}_{i,j}$ to be used.

Example 5. A weaker primitive than $\textsc{Auth}_{i,j}$ is a channel for which the ouput is not guaranteed to be equal to the input. For example, some error could be added to the input. Another example is a binary channel for which when the input is 0, then the output is also guaranteed to be 0, but if the input is 1, then the output can be 0 or 1.

Example 6. A broadcast channel from a sender $P_i \in \mathcal{P}$ to the remaining players $\mathcal{P} \setminus P_i$, secure for up to t cheaters among them, is denoted as BC_i^t and can be formalized as follows:

$$\mathrm{BC}_i^t(H, \boldsymbol{x}_H) = \left\{ \boldsymbol{y} \in \mathcal{R}_H \;\middle|\; \exists v \left((\forall j \in H : \boldsymbol{y}_{|j} = v) \;\wedge\; (P_i \in H \Rightarrow v = \boldsymbol{x}_{H|i}) \right) \right\}$$

if $|H| \leq t$ and

$$\mathrm{BC}_i^t(H, \boldsymbol{x}_H) = \mathcal{R}_H$$

if $|H| > t$.

Example 7. Detectable broadcast [FGH+02] among n players is a consistency specification where, again, only one player P_i has an input. If all players are honest ($H = \mathcal{P}$), then all players receive the sender's input, but if one or more players cheat, then all player either output the sender's input or a special failure symbol $\perp$. It is obvious how this specification can be formalized

Example 8. Two-threshold broadcast [FHHW03] is a consistency specification with two thresholds, t and T, satisfying $1 \leq t < T \leq n$. If the number of cheaters is at most t, then broadcast is achieved. In one of the flavors of two-threshold broadcast, consistency (but not correctness) is guaranteed even if the number of cheaters is between $t + 1$ and T. This means that all honest players receive the same value v, but even if the sender is honest, v need not be equal to the sender's input. It was proved in [FHHW03] that this is achievable if and only if $t + 2T < n$. Note that this is a strict generalization of the $t < n/3$ bound (when $t = T$).

Example 9. Consider a setting where a value in the domain $[1, \ldots, d]$ should be broadcast by a sender P_i. A less demanding goal is that all honest players' outputs lie in a certain (small) interval of length at most m, say, where m can depend on the number of cheaters. In other words, the fewer cheaters there are, the more accurate are the received values, capturing the idea of smooth security degradation. If the sender is honest, then the interval must contain his input value. It is an open problem to determine for which choices of parameters this consistency specification can be achieved.

3.3 A Partial Order on Consistency Specifications

It is natural to define the following relation between consistency specifications.

Definition 4. Consider two consistency specifications $\mathcal{C}$ and $\mathcal{C}'$ for the same player set $\mathcal{P}$, input domains $\mathcal{D}_1, \ldots, \mathcal{D}_n$ and output domains $\mathcal{R}_1, \ldots, \mathcal{R}_n$. Then $\mathcal{C}$ is *stronger* than $\mathcal{C}'$, denoted $\mathcal{C} \geq \mathcal{C}'$, if

$$\mathcal{C}(H, \boldsymbol{x}_H) \subseteq \mathcal{C}'(H, \boldsymbol{x}_H)$$

for all $H \subseteq \mathcal{P}$.

Note that this is only a partial order relation, i.e., two consistency specifications are generally incomparable.

4 Protocols and Reductions Among Consistency Specifications

A fundamental principle in computer science is to construct a complex system from simpler subsystems with well-defined interfaces and specifications. In our context, the basic question is if and how one can achieve a certain consistency primitive by invoking some weaker primitives.

Example 10. A trivial example is that one can use a broadcast primitive for a certain input (and output) domain $\mathcal{D}$ directly as the corresponding primitive for an input domain $\mathcal{D}' \subseteq \mathcal{D}$. For this purpose, the players must reject the output (and take as output a fixed default value in $\mathcal{D}'$, e.g. 0) if the actual received value is outside of $\mathcal{D}'$.

Example 11. Conversely, one can use a broadcast primitive multiple times to enlarge the domain. For example, one can directly use a binary ($\mathcal{D}_i = \{0,1\}$) broadcast primitive k times to obtain the corresponding broadcast primitive for k-bit strings ($\mathcal{D}_i = \{0,1\}^k$).

If a consistency primitive, described by consistency specification $\mathcal{C}$, can be achieved by some protocol invoking some weaker primitives described by consistency specifications $\mathcal{C}_1, \ldots, \mathcal{C}_m$ (for some m), one can say that $\mathcal{C}$ is *reduceable* to $\mathcal{C}_1, \ldots, \mathcal{C}_m$, denoted

$$\{\mathcal{C}_1, \ldots, \mathcal{C}_m\} \to \mathcal{C}.$$

As mentioned before, the main result of [LSP82] can be stated as

$$\textsc{Auth} \to \mathrm{BC}_i^{\lfloor n/3 \rfloor}$$

for any i. Similarly, the main result of [FM00] can be rephrased as

$$\mathrm{BC}(3) \to \mathrm{BC}_i^{\lfloor n/2 \rfloor}$$

for any i, where $\mathrm{BC}(3)$ denotes the set of all broadcast channels from one sender to two other receivers. Note that, trivially, $\mathrm{BC}(3) \to \textsc{Auth}$ and hence authenticated channels need not explicitly be mentioned on the left side of the above formula. Note also that $\mathcal{C}_1 \geq \mathcal{C}_2$ trivially implies $\mathcal{C}_1 \to \mathcal{C}_2$.

A general protocol for achieving a consistency specification $\mathcal{C}$ with input domains $\mathcal{D}_1, \ldots, \mathcal{D}_n$ and output domains $\mathcal{R}_1, \ldots, \mathcal{R}_n$, based on some available consistency specifications $\mathcal{C}_1, \ldots, \mathcal{C}_m$, can be described as follows. The protocol consists of some ℓ rounds, where in each round one of the given primitives $\mathcal{C}_1, \ldots, \mathcal{C}_m$ is invoked. Let $\mathcal{C}^{(j)}$ be the primitive invoked in the jth round. In this round, each player computes the input to $\mathcal{C}^{(j)}$ as a function of his current state, which consists of the input to $\mathcal{C}$ as well as all the outputs of the previous calls to primitives $\mathcal{C}^{(1)}, \ldots, \mathcal{C}^{(j-1)}$. At the end of the protocol, each player computes the output (of $\mathcal{C}$) as a function of the final state.

Let us give a more formal description of a protocol π. Let the sequence of ℓ primitives $\mathcal{C}^{(1)}, \ldots, \mathcal{C}^{(\ell)}$ to be called in the protocol be fixed, and let the input and output domains of $\mathcal{C}^{(j)}$ be $\mathcal{D}_1^{(j)}, \ldots, \mathcal{D}_n^{(j)}$ and $\mathcal{R}_1^{(j)}, \ldots, \mathcal{R}_n^{(j)}$, respectively.

Definition 5. An ℓ-round protocol π for input domains $\mathcal{D}_1^{(j)}, \ldots, \mathcal{D}_n^{(j)}$ and output domains $\mathcal{R}_1^{(j)}, \ldots, \mathcal{R}_n^{(j)}$ consists of a list of functions $f_i^{(j)}$ for $1 \leq j \leq \ell$ and $1 \leq i \leq n$ as well as a list of functions g_i for $1 \leq i \leq n$, where

$$f_i^{(j)} : \mathcal{D}_i \times \mathcal{R}_i^{(1)} \times \cdots \times \mathcal{R}_i^{(j-1)} \to \mathcal{D}_i^{(j)}$$

and

$$g_i : \mathcal{D}_i \times \mathcal{R}_i^{(1)} \times \cdots \times \mathcal{R}_i^{(\ell)} \to \mathcal{R}_i.$$

Note that this definition refers only to the domains, but not to the primitives called in the protocol. The following definition captures when a certain protocol achieves a certain consistency specification when a given list of (domain-compatible) primitives is invoked.

Definition 6. Protocol π with a given schedule for calling primitives in the set $\{\mathcal{C}_1, \ldots, \mathcal{C}_m\}$ *reduces* $\mathcal{C}$ to $\{\mathcal{C}_1, \ldots, \mathcal{C}_m\}$, denoted

$$\{\mathcal{C}_1, \ldots, \mathcal{C}_m\} \xrightarrow{\pi} \mathcal{C},$$

if for all $H \subseteq \mathcal{P}$, for all $\boldsymbol{x}_H \in \mathcal{D}_H$, and for all choices of functions $f_i'^{(j)}$ and g_i' for replacing $f_i^{(j)}$ and g_i in π, for indices i with $P_i \notin H$, the list of output values of the players in H, when executing the modified protocol, is in $\mathcal{C}(H, \boldsymbol{x}_H)$.

The quantification over all functions $f_i'^{(j)}$ and g_i' for i with $P_i \notin H$ is needed because the cheating players can use arbitrary cheating strategies.

Acknowledgments

I would like to thank Reto Aschwanden, Matthias Fitzi, Martin Hirt, Fabian Kuhn, Renato Renner, Reto Strobl, and Jürg Wullschleger for interesting discussions.

References

[Asc01] R. Aschwanden. Consistency specifications. Diploma Thesis, Dept. of Computer Science, ETH Zurich, May 2001.

[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.

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

[CFF$^+$04] J. Considine, M. Fitzi, M. Franklin, L. A. Levin, U. Maurer, and D. Metcalf. Byzantine agreement in the partial broadcast model. Manuscript, July 2004.

[FGH$^+$02] M. Fitzi, D. Gottesman, M. Hirt, T. Holenstein, and A. Smith. Detectable Byzantine agreement secure against faulty majorities. In *Proc. 21st ACM Symposium on Principles of Distributed Computing (PODC)*, July 2002.

[FHHW03] M. Fitzi, M. Hirt, T. Holenstein, and J. Wullschleger. Two-threshold broadcast and detectable multi-party computation. In *Advances in Cryptology — EUROCRYPT '03*, Lecture Notes in Computer Science, Springer-Verlag, vol. 2656, pp. 51-67, 2003.

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

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

[FM00] M. Fitzi and U. Maurer. From partial consistency to global broadcast. In *Proc. 32nd Annual ACM Symposium on Theory of Computing (STOC '00)*, pp. 494–503. ACM Press, 2000.

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

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

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

[KS01] F. Kuhn and R. Strobl. Towards a general theory for consistency primitives. Term project report, Dept. of Computer Science, ETH Zurich, Nov. 2000.

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

[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.