Rationality and Adversarial Behavior in Multi-Party Computation
(Extended Abstract)

Anna Lysyanskaya and Nikos Triandopoulos

Department of Computer Science, Brown University
{anna, nikos}@cs.brown.edu

Abstract. We study multi-party computation in the model where none of n participating parties are honest: they are either rational, acting in their selfish interest to maximize their utility, or adversarial, acting arbitrarily. In this new model, which we call *the mixed-behavior model*, we define a class of functions that can be computed in the presence of an adversary using a trusted mediator. We then give a protocol that allows the rational parties to emulate the mediator and jointly compute the function such that (1) assuming that each rational party prefers that it learns the output while others do not, no rational party has an incentive to deviate from the protocol; and (2) the rational parties are protected from a malicious adversary controlling $\lceil \frac{n}{2} \rceil - 2$ of the participants: the adversary can only either cause all rational participants to abort (so no one learns the function they are trying to compute), or can only learn whatever information is implied by the output of the function.

1 Introduction

Multi-party computation (MPC) has emerged as the central problem in cryptography, of which many scenarios that arise in practice are simply a special case. In multi-party computation, we have n parties, each party P_i with its input x_i, wishing to jointly compute a function $f(x_1, \ldots, x_n)$. In the traditional secure MPC scenario (SMPC) (e.g., [5, 6]), there are two kinds of parties: honest parties that will follow a prescribed protocol, and adversarial parties that will collude and deviate from the protocol in an arbitrary way. An SMPC protocol should guarantee that no adversary controlling up to t parties will be able to learn anything about the honest parties' inputs not implied by the value $f(x_1, \ldots, x_n)$.

In this paper, we introduce the mixed-behavior model for multi-party computation (MMPC). Here, some of the parties will be *rational*, and will only follow a protocol if they have a clear incentive to do so. Others will be *adversarial*, and will collude and exhibit arbitrary, possibly irrational, behavior. The goal is to design protocols that compute the function, guarantee the same security properties as in usual SMPC and that rational parties will choose to follow.

Rational MPC vs. Mixed-Behavior Model. Halpern and Teague [8] considered rational MPC (RMPC), i.e., the setting where n *rational* parties wish to securely compute a joint function of their inputs, and there is no adversary. They showed a surprising impossibility result: suppose that each party P_i holds an additive secret share x_i of a secret $x = \sum x_i$. Suppose that each party prefers to learn x to not learning it, and prefers that as few as possible other parties learn x. In this case it was shown that there is no deterministic protocol for reconstructing x that rational parties will have an incentive to follow (or, put another way, for any deterministic strategy, there will be another one that each rational party will prefer). On the other hand, they gave a randomized protocol that did the job. The idea was to first give a protocol for $n = 3$. Then for $n > 3$, the proposed solution was to first partition the parties into three groups, have each group select a leader and have each group member send his secret share to his leader. It is easy to see that the approach above immediately fails if an adversary controls three of the parties: the adversary controlling the leaders will learn the secret, and no one else ever will. Thus, Halpern and Teague leave the problem of MPC in the mixed-behavior model totally open.

Independently of our work, Abraham *et al.* [1] study general function evaluation in the RMPC model, in an extended setting where rational parties can collude with each other or are allowed to have non-standard utilities, and they design protocols that are robust against this type of scenarios (e.g., resilient equilibria). Gordon and Katz [7] study rational secret sharing and MPC and provide a protocol that improves on the original protocol in [8] (e.g., it is simpler and symmetric and captures the case $n = 2$). The protocols of both works are conceptually similar to our protocol; both are for RMPC, though.

Assumptions on Communication Model vs. Assumptions on Utilities. Recently, Izmalkov *et al.* [9] introduced rational secure MPC (RSMPC). Their goal is to realize games without a trusted mediator, no matter what the underlying utility functions of the players are. Here, there is no adversary, but any subset of parties may be maliciously trying to collude in an attempt to maximize their utility. The main challenges of RSMPC are ensuring that players, first of all, contribute legal information into the game (e.g., if they are playing a card game, a player cannot use a card that was not dealt to him), and additionally that the players cannot collude while computing the outcome of the game, communicating to each other the information that they would not be able to communicate if the game were played using a trusted mediator. In order to address these challenges, Izmalkov *et al.*, as well as the related works in [10, 11], place severe restrictions on the communication model (no channels of communication except those explicitly available), and utilize physical primitives such as a ballot box and a physical envelope. In contrast, we use standard communication channels, allow existence of covert channels and steganography, and make assumptions instead about the utility functions of the parties: we assume that the parties already have an incentive to participate in the protocol and to contribute their true inputs. These assumptions enable us to come up with protocols where rational parties will not be motivated to deviate in any way.

Intuition for the Construction. In a nutshell, the impossibility result of Halpern and Teague goes as follows. Suppose that this is the last round of the reconstruction protocol of a secret sharing scheme; then sending out information cannot increase utility, but may decrease it by allowing another party to learn x. How can it be, then, that a randomized protocol exists? In any given round, the players do not know whether this is supposed to be the last round (and so they would do better by keeping their information to themselves) or whether this is a test round in which no meaningful information can be revealed, but instead the parties are just being tested, and deviating from the protocol will have the consequence that others will abort the protocol. This is the key idea of in Halpern and Teague's protocol that we will use also.

Our protocol assumes that we have a *synchronous* broadcast channel. That means that, in computing message for round i, no one can take into account other parties' message for this round: waiting for those messages to arrive will mean missing your chance to speak in round i. Further, we assume that all parties, both rational and adversarial, are computationally bounded. As a subroutine, we will invoke a traditional SMPC protocol over the broadcast channel, specifically the GMW protocol [6] as analyzed in the UC model by Canetti *et al.* [3]. At each step, each party will provide a ZK proof that the data it published on the channel was computed according to the protocol as a function of the data it previously received, as well as of its input and random tape. We call a protocol where at each step each party provides such a proof, a *verifiable* MPC protocol.

Our main protocol for MMPC works roughly as follows. In a setup step, we set up the parameters of the computation, such as the common random string needed for secure multi-party computation and non-interactive zero-knowledge proofs. Next, as is usual in MPC, we run a preprocessing step as a result of which each party P_i is committed to its input x_i. Next is the key step: using a verifiable MPC protocol, the parties come up with an m-out-n secret sharing (e.g., polynomial secret sharing [12]) either of $y = 0$ (with probability $1/2$) or of the value $y = f(x_1, \dots, x_n)$ (we assume that f never returns a 0). Here, n is the number of participants, and we will specify m later. If at any point of this protocol, any party P_j deviates (note that since we are using verifiable MPC, a deviation is detectable with high probability), the protocol tells P_i to abort.

So far, no party has an incentive to deviate from the protocol: deviation will be detected, causing everyone following the protocol to abort the computation. On the other hand, at least computationally, no information about $f(x_1, \dots, x_n)$ has been communicated so far by the security properties of the regular SMPC. In the next step, each party P_i broadcasts its share of y and a non-interactive ZK proof of correctness. If m correct shares are broadcasted, then the parties combine them to obtain y, and if it is the case that $y \neq 0$, then they obtain $f(x_1, \dots, x_n)$. If combining the correct shares present does not yield $f(x_1, \dots, x_n)$, and fewer than n correct shares are present, then the protocol aborts. Otherwise, go back to the key step above.

Consider what damage an adversary controlling $m-1$ participants can inflict. It can approach $n-2m+2$ rational parties, give each of them $m-1$ secret shares of the outcome, thus guaranteeing that each party will be able to *locally* (as opposed to in collaboration with other rational parties) compute the outcome and therefore will not participate in the protocol any longer. Then, the remaining $m-1$ parties, no matter what strategy they adopt between themselves, will be unable to compute anything meaningful (they do not have enough shares). This is undesirable, since we want a protocol in which, no matter what the adversary does, either everyone computes their output, or no one does. Thus, the best we can hope to do is to allow the adversary to control up to $t=m-2$ participants, which, we will show, our protocol will achieve. A key step in the proof is to show that rational parties will have a disincentive to collude with others.

Consider what happens if only m parties are rational. Suppose that $m-1$ of them are following the protocol, and the party P_i is considering deviating. Note that, depending on P_i's utility function, it may be a good idea for him to not broadcast his share of y. He risks causing everyone to abort (with probability $1/2$), but on the other hand, his utility might skyrocket in case he is lucky and $y \neq 0$. On the other hand, if $m+1$ participants are rational, and we know that all of them except P_i are following the protocol, then P_i can only lose in utility if he does not broadcast his share of y_i, since in case $y=0$, he will force everyone to abort without ever computing the function, while if $y \neq 0$, everyone will learn the outcome whether or not P_i broadcasts. Therefore, we see that the protocol is a Nash equilibrium if $m+1$ or more of the participants are rational. Accordingly, we set the value of m to be $m=\lceil \frac{n}{2} \rceil$, thus we derive a protocol that tolerates adversarial behavior for any adversary controlling up to $\lceil \frac{n}{2} \rceil - 2$ of the parties.

Note that the above observation solves an open problem posed by Halpern and Teague who asked whether it was possible to have a protocol that works independently on how much utility is attached to being the only party privy to the output of the function. Also, note that the only communication channel that our protocol needs is a synchronous broadcast channel, while Halpern and Teague assume private channels in addition to synchronous broadcast. Unlike Izmalkov *et al.*, our protocol tolerates the existence of additional communication channels between the participants.

Assumptions on Utilities and on Rationality. Halpern and Teague observe that rational parties are not going to participate in a protocol unless they have an incentive to compute the function and, moreover, they must have an incentive to submit their true individual inputs. Therefore, the function must be non-cooperatively computable (NCC) [13]. We need to extend this assumption to work in the mixed-behavior model. For example, if the function in question is computable based on inputs of a small number of parties, and covert channels between participants are present, then the parties have no incentive to participate in the protocol to begin with, since they may learn the output of the function from the adversary. Therefore, in order to tolerate t adversarial participants, we must require that it is impossible to compute the function in question based on $t+1$ of the inputs. We define such functions, called t-NCC, in Definition 5.

Also following Halpern and Teague, we make assumptions that the more a party learns about the output, the happier he is. On the other hand, the less others learn about it, the happier he is as well. Since the adversary is not rational and is not trying to learn anything, the utility functions should ignore whether or not the adversary learns anything. We explore this in Definitions 3 and 4.

Suppose that two different strategies of a party yield indistinguishable views. Then the difference in how much this party learned is negligible, assuming polynomial-time algorithms. Our final assumption on utilities is that negligible difference in how much P_i or others have learned yield negligible differences in utilities. We call this assumption the *computational satisfaction assumption*, and state it more precisely in Definition 9.

Following Halpern and Teague, rationality is captured by postulating that a party will choose to follow a given protocol if (1) it is a Nash equilibrium, i.e., provided that other non-adversarial players follow the protocol, this party's utility is maximized by following it as well; (2) it survives the process of iterated deletion of weakly dominated strategies—i.e., those strategies for which another strategy is always at least as good and sometimes even better. However, what do these mean in the mixed-behavior model? We adapt these game-theoretic definitions to the scenario where a malicious adversary exhibiting arbitrary behavior is present. In a nutshell, a set of strategies is a Nash equilibrium in the mixed-behavior model (Definition 6) if it is a Nash equilibrium for all adversaries. A strategy σ weakly dominates another strategy τ if for all adversaries and no matter what strategy other rational parties are following, σ gives as much or better utility than τ, *and* for some adversary and some set of strategies for the remaining rational parties, σ gives strictly more utility (Definition 7).

Our protocol is not a preferred equilibrium in the strict sense because it relies on the computational intractability of certain problems that can still be solved with negligble, but positive, probability. What we show is that no deviation can increase the utility by more than a negligible amount (Definitions 6 and 7).

Paper Outline. In Section 2 we present the mixed-behavior model for multi-party computation, introduce related new concepts and formally define the notion of unconditionally or computationally secure preferred protocols. In Section 3, we describe a protocol that implements any function in the mixed-behavior model, using a special channel, and prove that it is an unconditionally secure preferred protocol. In Section 4 we describe our main protocol over a synchronous broadcast channel, and prove that it is a computationally secure preferred protocol that is ϵ-Nash and survives iterated deletion of ϵ-weakly dominated strategies, where ϵ is negligible. We refer the reader to the full version of the paper for all the details this extended abstract omits.

2 Mixed-Behavior Model for Multi-Party Computation

We introduce a new model for multi-party computation, the *mixed-behavior model* (MMPC). In the standard multi-party computation setting, consider any protocol that implements a function f when executed by n parties $\mathcal{P}$; f is either unary ($f(\mathbf{x})$ is the common output) or n-ary, denoted as $\mathbf{f}$ ($\mathbf{f}(\mathbf{x}) = (y_1, \ldots, y_n)$ are the private outputs). In the mixed-behavior model, parties $\mathcal{P}$ are partitioned into *rational* parties $\mathcal{R}$ and *adversarial* parties $\mathcal{A}$. Rational parties consider the joint computation to be a game, during which they act selfishly, deviating from a prescribed protocol and exhibiting strategies aiming to increase their gained utility after the termination of the computation. At the same time, an adversary $\mathcal{A}$ controls up to t of the parties, which jointly exhibit arbitrary strategies, sharing information and acting adversarially during the computation.

We next define concepts related to computations performed in the mixed-behavior model and also new properties that protocols must satisfy, both unconditionally and computationally. We use some standard notation: $[n]$ denotes set $\{1,\ldots,n\}$; bold letters denote vectors, where $\mathbf{v}=(v_1,\ldots,v_n)$ can be written as $(v_i,\mathbf{v}_{-i})$ for any $i\in[n]$; $(v_i',\mathbf{v}_{-i})$ denotes $\mathbf{v}'=(v_1,\ldots,v_{i-1},v_i',v_{i+1},\ldots,v_n)$; $\mathbf{v}_I$ denotes vector projection to set $I\subseteq[n]$. We write $\mathcal{P}=\mathcal{R}\cup\mathcal{A}$ to denote that a set of parties $\mathcal{P}$ is partitioned into rational parties $\mathcal{R}$ and adversarial parties $\mathcal{A}$, where $\mathcal{A}$ also denotes the actual programs run by the adversary. If A is a randomized algorithm, $a\leftarrow A$ denotes that a was obtained by running A.

Definition 1 (Communication channels and structures). *Let $\mathcal{P}$ be a set of interactive TMs. A* channel $C=\{\mathcal{P}_I,F_C,\mathsf{state},\mathcal{P}_O\}$ available to $\mathcal{P}$ *is a (possibly, stateful and randomized) TM with $\mathcal{P}_I,\mathcal{P}_O\subseteq\mathcal{P}$ which operates as follows. C shares a dedicated tape t_i^I with each $P_i\in\mathcal{P}_I$ and a dedicated tape t_i^O with each $P_i\in\mathcal{P}_O$. C gets activated when a party $P_i\in\mathcal{P}_I$ writes a string I_i on t_i^I. If C is a* synchronous *channel, then it waits for all other parties in $\mathcal{P}_I$ to submit an input, or times out with a default value $\perp$. If activated, C computes the values $(\mathbf{O},\mathsf{state}')\leftarrow F_C(\mathbf{I},\mathsf{state})$, writing value O_i on tape t_i^O, $\forall i\in[n]$, and updating its state information. A* channel structure $\mathcal{C}(\mathcal{P})$ *is a set of channels available to $\mathcal{P}$, always containing channel $\{\{P_i\},I,\perp,\{P_i\}\}$, $\forall i\in[n]$ (I: identity function).*

It is immediate how the above formulation of a communication channel as an interactive Turing machine captures the conventional notion of any channel (e.g., point-to-point or broadcast channels). Moreover, it constitutes a generalization of a communication channel: by allowing a channel to keep state and by appropriately setting its operation function F_C, we can essentially define any ideal functionality over inputs submitted by the parties in $\mathcal{P}$, thus expressing useful additional properties in computations (e.g., perfect privacy).

We consider joint computations among individual parties $\mathcal{P}$ performed over some underlying communication channel C available to $\mathcal{P}$. A computation consists of a (possibly infinite) series of rounds, each round corresponding to an activation of C. Each participating party $P_i\in\mathcal{P}$ interacts with C through its corresponding tapes: P_i runs a (possibly randomized) program that processes strings appearing on t_i^O (incoming messages) and computes a string to be written on t_i^I (outgoing messages), also updating an internal state. Also, additional channels may be used among parties in $\mathcal{P}$ in the course of a computation.

Definition 2 (Runs, views, outputs). *Let $\mathcal{P}=\mathcal{R}\cup\mathcal{A}$ be a set of parties jointly computing over channel structure $\mathcal{C}(\mathcal{P})$. A* run R *of the system $(\mathcal{P},\mathcal{C}_\mathcal{P})$ is tuple $(\mathbf{p},\mathbf{r},\mathcal{A})$, where $\mathbf{p}_i$ is the (randomized) program that party $P_i\in\mathcal{R}$ runs, $\mathcal{A}$ also denotes the program that parties in $\mathcal{A}$ jointly run, and $\mathbf{r}$ is the vector of random tapes needed by parties in $\mathcal{P}$. The* view *of party $P_i\in\mathcal{R}$ in run R, denoted*

$VIEW_i(R)$, is tuple $(r_i, \mathbf{M}_i)$, where $\mathbf{M}_i$ is the set of all messages received by P_i; the view of the adversary $\mathcal{A}$, denoted $VIEW_{\mathcal{A}}(R)$, is tuple $(\mathbf{r}_{\mathcal{A}}, \mathbf{M}_{\mathcal{A}})$, where $\mathbf{M}_{\mathcal{A}}$ is the set of all messages received by the parties in $\mathcal{A}$. Each message in $\mathbf{M}_i$ or $\mathbf{M}_{\mathcal{A}}$ specifies both the channel over which it was received, and its contents (and possibly its sender). The output *of a party $P_i \in \mathcal{R}$ in run R, denoted $OUT_i(R)$, is the last message received (and possibly sent) by P_i in R, if R terminates.*

Definition 3 (Protocol utility and satisfaction functions). *Let $\mathcal{P} = \mathcal{R} \cup \mathcal{A}$ be a set of parties. We say that $\mathbf{u}$ is a* protocol utility function with satisfaction *$\boldsymbol{\mu}$ if for any run R: (1) for $P_i \in \mathcal{P}$, $\mu_i : \{0,1\}^* \mapsto [0,1] \cup \perp$ is any function, such that $\mu_i(\cdot) \in [0,1]$ if $P_i \in \mathcal{R}$ and $\mu_i(\cdot) = \perp$ if $P_i \in \mathcal{A}$ and (2) there exists a set of functions $\mathbf{u}'$ for parties $\mathcal{R}$, where $u'_i : [0,1]^n \mapsto [0,\infty]$ and $u_i(R) = u'_i(\boldsymbol{\mu}(R))$.*

The above definition captures our formulation of a joint computation as a game. Rational parties are each associated with a utility function, mapping (consequences of) outcomes of the computation—which is fully described by protocol runs—to positive reals as follows. A personal satisfaction function μ_i maps the view that $P_i \in \mathcal{R}$ gets out of a computation to a value in $[0,1]$ according to certain criteria for P_i about what constitutes a desired outcome; intuitively, on input $VIEW_i(R)$, μ_i measures how well P_i succeeded in computing whatever it wished to compute. Then P_i evaluates the outcome of a computation by getting (through u'_i) a utility that depends on the evaluations of all parties' satisfaction functions on the protocol run. In essence, u_i is what characterizes P_i rationality, meaning that any deviation from a proposed protocol by P_i corresponds to a strategy that can be preferable or not for P_i depending solely on utility function u_i. In contrast, the adversary $\mathcal{A}$ has no specific desired outcomes; if $P_i \in \mathcal{A}$ then $\mu_i = \perp$ and there is no associated utility. This is in accordance with our intuition: $\mathcal{A}$ acts in an arbitrary, possibly irrational way. Note that it is not necessarily the case that $P_i \in \mathcal{R}$ can infer his utility u_i from his view, (since, e.g., he does not know who the adversary is); however, P_i can still act so as to maximize u_i.

We next present some minimal assumptions on the above definition, providing a concrete notion of rationality, i.e., incentives for protocol deviation, in the MMPC model. We do this by defining target functions. Intuitively, given a vector of utility functions $\mathbf{u}$, a vector of functions $\mathbf{f}$ is its corresponding target function if the utilities capture the fact that each rational P_i wishes to compute f_i, and prefers that as few as possible other rational P_j's compute f_j.

We assume that inputs $\mathbf{x}$ for computations among parties in $\mathcal{P}$ are chosen by first drawing inputs $\mathbf{x}'$ from distribution $\mathbf{X}$, then setting $\mathbf{x}_{\mathcal{R}} = \mathbf{x}'_{\mathcal{R}}$ and, finally letting $\mathbf{x}_{\mathcal{A}} \leftarrow \mathcal{A}(\mathbf{x}'_{\mathcal{A}})$. That is, $\mathbf{x}$ is formed by generating a set of values according to $\mathbf{X}$, and then replacing the entries corresponding to adversarial parties with those chosen by the adversary. We denote the above process as $\mathbf{x} \leftarrow_{\mathcal{A}} \mathbf{X}$.

How does μ_i captures what P_i wants to learn? The measure of how well P_i learns some function $f_i(\mathbf{x})$ is the likelihood p_i that he outputs the correct $f_i(\mathbf{x})$, where the probability is taken over $\mathbf{X}$ as well as the randomness of the particular run of the protocol. This is relatively standard in the cryptographic definitional literature, starting with semantic security. The only adjustment we want to make is to scale μ_i so that it is 0 if p_i is the same as the *a-priori* probability that P_i outputs $f_i(\mathbf{x})$, and 1 if $p_i = 1$. More formally:

Definition 4 ((Computational) target function). *For $\mathcal{P} = \mathcal{R} \cup \mathcal{A}$, let $\mathbf{u}$ be a protocol utility function with satisfaction measure $\boldsymbol{\mu}$ and let $\mathbf{x} \leftarrow_{\mathcal{A}} \mathbf{X}$. Then $\mathbf{f} : (\{0,1\}^*)^n \mapsto (\{0,1\}^*)^n$ is an n-ary* (computational) target function *for input distribution $\mathbf{X}$ and adversary $\mathcal{A}$ if the following conditions are met:*

$P_i \in \mathcal{R}$ **wants to learn** f_i**:** *μ_i is a (computational) measure of how much P_i has learnt about $f_i(\mathbf{x})$ based on the run R. Let some algorithm F_i be the best (polynomial-time) estimator of $f_i(\mathbf{x})$ both based on x_i alone (i.e., $\mu_i(x_i)$) and based on the view $VIEW_i(R)$ obtained by P_i in run R (i.e., $\mu_i(R) \triangleq \mu_i(VIEW_i(R))$). Assume that $\Pr[F_i(x_i) = f_i(\mathbf{x})] \neq 1$ (i.e., it is impossible to always correctly output $f_i(\mathbf{x})$ based on x_i alone). Then*

$$\mu_i(R) = (\Pr[F_i(R) = f_i(\mathbf{x})] - \Pr[F_i(x_i) = f_i(\mathbf{x})])/(1 - \Pr[F_i(x_i) = f_i(\mathbf{x})]).$$

The scaling and normalizing in the formula above is done so that if for a run R, $\Pr[F_i(R) = f_i(\mathbf{x})] = 1$, then $\mu_i(R) = 1$; and if for a run R, $\Pr[F_i(R) = f_i(\mathbf{x})] = \Pr[F_i(x_i) = f_i(\mathbf{x})]$ (i.e., run R did not increase P_i's chances of correctly computing $f_i(\mathbf{x})$), then $\mu_i(R) = 0$. By convention, if in run R, $\mu_i(R) = 1$, we write that $OUT_i(R) = f_i(\mathbf{x})$. Moreover, if for some $P_i \in \mathcal{R}$, and runs R and R' it is the case that $\mu_i(R) > \mu_i(R')$ and for all $j \neq i$, $\mu_j(R) = \mu_j(R')$, then $u_i(R) > u_i(R')$.

P_i **does not want others to learn:** *If for some $P_j \in \mathcal{R}$ and runs R and R' it is the case that $\mu_j(R) > \mu_j(R')$, and for all $P_i \in \mathcal{R}$, $i \neq j$, $\mu_i(R) = \mu_i(R')$, then for all $P_i \in \mathcal{R}$, $u_i(R) < u_i(R')$.*

Worst outcome: *Let R be a run such that for some $P_j \in \mathcal{R}$, $OUT_j(R) = f_j(\mathbf{x})$. Then for all $P_i \in \mathcal{R}$, if $OUT_i(R) \neq f_i(\mathbf{x})$, then $u_i(R) = 0$.*

Thus, we consider functions which for rational parties and with respect to their utility functions satisfy properties that express selfishness and antagonism in multi-party computation: $u_i(R)$ strictly increases when P_i gets closer to the target value $f_i(\mathbf{x})$ or computes $f_i(\mathbf{x})$, or when some other party P_j gets further away from its target value $f_j(\mathbf{x})$. This formulation is a generalization of the rationality in RMPC [8]. In contrast, adversarial parties behave totally arbitrarily and unpredictably: the adversary $\mathcal{A}$ acts independently of parties' rationality and its behavior may affect rational parties' utilities; e.g., $\mathcal{A}$ may consistently try to minimize or even maximize the utility of a specific rational party.

In the mixed-behavior model, we are interested in computing (target) functions for which rational parties have incentive to submit their true values as input to the joint computation. This class of functions, introduced by Shoham and Tennenholtz as *non-cooperatively computable* functions, were studied for the binary case in [13]. We next extend this concept to general n-ary functions defined on strings (rather than single bits) for computations in MMPC. In the mixed-behavior setting, we also need to capture the case where a malicious adversary changes some of the inputs: we wish to ensure that we still express honesty in submitting inputs, but capture the possible scenario where a rational party does worse by substituting his true input.

Definition 5 (t-NCC functions). *Let $\mathbf{f} : \mathbf{X} \mapsto \mathbf{Y}$ be an n-ary function, let $\mathcal{P} = \mathcal{R} \cup \mathcal{A}$, $|\mathcal{P}| = n$, $\mathbf{u}$ a protocol utility function with satisfaction $\boldsymbol{\mu}$. Let the channel structure of $\mathcal{P}$ consist just of synchronous channel $C = (\mathcal{P}, \mathbf{f}, \mathsf{state}, \mathcal{P})$ as well as a channel to and from the adversary $\mathcal{A}$; C is stateful and can only be used once. Let $Eqv_{\mathbf{f}}(i, x_i)$ be the maximal set of strings such that $\forall \mathbf{x}_{-i}$ and $\forall x \in Eqv_{\mathbf{f}}(i, x_i)$, $\mathbf{f}(x_i, \mathbf{x}_{-i}) = \mathbf{f}(x, \mathbf{x}_{-i})$. Let $\Sigma'_i(x_i)$ be the set of strategies for P_i that submit some $x \in Eqv_{\mathbf{f}}(i, x_i)$ to C, and output the value y_i received from C. Let $\sigma^*_i(x_i)$ be the strategy that submits x_i to C and outputs y_i. Suppose that input $\mathbf{x}$ is chosen with the process: (1) $\mathbf{x}' \leftarrow \mathbf{X}$; (2) $\mathbf{x}_{\mathcal{R}} = \mathbf{x}'_{\mathcal{R}}$; (3) $\mathbf{x}_{\mathcal{A}} \leftarrow \mathcal{A}(\mathbf{x}'_{\mathcal{A}})$. We say that $\mathbf{f}$ is* t-non-cooperatively computable (t-NCC) *under distribution $\mathbf{X}$ if for all $P_i \in \mathcal{R}$, for all adversaries $\mathcal{A}$, $|\mathcal{A}| \le t$: (1) $\mu_i(x_i, \mathbf{x}_{\mathcal{A}}) = 0$, i.e., nothing can be learned about $f_i(\mathbf{x})$ based on x_i and $\mathbf{x}_{\mathcal{A}}$ alone; (2) for all $\mathcal{A}$, i, $f_i(\mathbf{x}') = f_i(\mathbf{x})$, i.e., $\mathcal{A}$ cannot change the value of the function by changing his inputs; (3) for all $\mathcal{A}$, P_i gets at least as much utility from submitting his true input x_i to C as from substituting any other different input value: for all $1 \le i \le n$, all $\sigma_i \notin \Sigma'_i$,*

$$E_{\mathbf{X}}[u_i((\sigma_i(x_i), (\boldsymbol{\sigma}^*_{\mathcal{R}})_{-i}((\mathbf{x}_{\mathcal{R}})_{-i})), \perp, \mathcal{A}(\mathbf{x}_{\mathcal{A}}))] < E_{\mathbf{X}}[u_i(\boldsymbol{\sigma}^*_{\mathcal{R}}(\mathbf{x}_{\mathcal{R}}), \perp, \mathcal{A}(\mathbf{x}_{\mathcal{A}}))].$$

For $t > 0$ the above definition is restrictive: it requires that distribution $\mathbf{X}$ outputs codewords of an error-correcting code. For instance, requiring that $\mu_i(x_i, x_{\mathcal{A}}) = 0$ is needed to exclude the following scenario: $\mathcal{A}$ gives its inputs $\mathbf{x}_{\mathcal{A}}$ to party P_i; then if $\mu_i(x_i, x_{\mathcal{A}}) > 0$, P_i and $\mathcal{A}$ abort the computation and P_i gains positive utility, whereas other rational parties gain zero utility. This demonstrates an inherent difficulty for realizing "fair" joint computations. However, the resulting class of functions is still of interest if we consider computations over values for which a secret sharing has first been computed.

We next define properties that protocols in MMPC should satisfy, either unconditionally or computationally. For brevity we do not explicitly denote that the strategies are a function of input $\mathbf{x} \leftarrow_{\mathcal{A}} \mathbf{X}$; instead we say that parties in $\mathcal{P}$ receive inputs from $\mathbf{X}$ and write run $(\boldsymbol{\sigma}_{\mathcal{R}}(\mathbf{x}_{\mathcal{R}}), \mathbf{r}, \mathcal{A}(\mathbf{x}_{\mathcal{A}}))$ as $(\boldsymbol{\sigma}_{\mathcal{R}}, \mathbf{r}, \mathcal{A})$. Also, we simplify the cumbersome $(\sigma_i, (\boldsymbol{\sigma}_{\mathcal{R}})_{-i})$ to $(\sigma_i, \boldsymbol{\sigma}_{-i})$. Finally, for the computational counterparts of the following definitions, any strategy or the adversary receives 1^k as an additional input—where k is a (security) parameter for probabilistic, polynomial-time (PPT) algorithms.

Definition 6 ((ϵ-)Nash equilibrium in the mixed-behavior model). *A set of (PPT) strategies $\boldsymbol{\sigma}^*$ is a* (ϵ-)Nash equilibrium *for $\mathcal{P} = \mathcal{R} \cup \mathcal{A}$ receiving inputs from $\mathbf{X}$, with a given channel structure $\mathcal{C}(\mathcal{P})$ and protocol utility function $\mathbf{u}$ if for all $P_i \in \mathcal{R}$, for all (PPT) strategies σ_i, and all (PPT) adversaries $\mathcal{A}$,*

$$E_{\mathbf{X},\mathbf{r}}[u_i((\sigma_i, \boldsymbol{\sigma}^*_{-i}), \mathbf{r}, \mathcal{A})] \le E_{\mathbf{X},\mathbf{r}}[u_i((\sigma_i, \boldsymbol{\sigma}^*_{-i}), \mathbf{r}, \mathcal{A})] \ \ (+ \ \epsilon(k)).$$

The above definition extends the (canonical in game theory) concept of Nash equilibrium in the mixed-behavior and bounded computational models.

In MMPC, a strategy is a Nash equilibrium, if no deviation is strictly preferable for a rational party when all others do not deviate, and independently of what the adversary does; for an ϵ-Nash equilibrium in a computational setting it holds that, in the same environment, no deviation is preferable by more than $\epsilon(k)$.

We next consider a refinement of the Nash equilibrium, using the notion of *weakly dominated strategies* and the process of *iterated deletion* of such strategies which define a better, preferred, notion of equilibrium. We use analogues of these game-theoretic concepts in the mixed-behavior and bounded computational models to define the corresponding refined Nash equilibria. Intuitively, a strategy survives the process of iterated deletion of weakly dominated strategies if no other strategy is (always) preferable to it. This last notion of preferable strategies is expressed as a partial order over individual strategies. Strategy σ_i' of P_i is preferable to σ_i if there exists a specific environment (strategies of others and of $\mathcal{A}$) under which σ_i gives strictly less utility than σ_i', whereas at the same time, in no other environment does σ_i give strictly larger utility.

Definition 7 (Iterated deletion of (ϵ-)weakly dominated strategies). *Given $\mathcal{P} = \mathcal{R} \cup \mathcal{A}$ receiving inputs from $\mathbf{X}$, with communication structure $\mathcal{C}$ and protocol utility functions $\mathbf{u}$, let $\boldsymbol{\Sigma}'$ be a set of (PPT) strategies for $\mathcal{R}$. Strategy σ_i is* (computationally ϵ-)weakly dominated *by (PPT) strategy σ_i' restricted to $\boldsymbol{\Sigma}'$ (denoted $[\sigma_i < \sigma_i' \mid_{(\epsilon)} \boldsymbol{\Sigma}']$) if (1) for all (PPT) strategies $\boldsymbol{\sigma}_{-i} \in \boldsymbol{\Sigma}'_{-i}$, for all (PPT) adversaries $\mathcal{A}$,*

$$E_{X,\mathbf{r}}[u_i((\sigma_i, \boldsymbol{\sigma}_{-i}), \mathbf{r}, \mathcal{A})] \le E_{X,\mathbf{r}}[u_i((\sigma_i', \boldsymbol{\sigma}_{-i}), \mathbf{r}, \mathcal{A})]$$

and (2) for some (PPT) $\boldsymbol{\sigma}'_{-i} \in \boldsymbol{\Sigma}'_{-i}$, for some (PPT) adversary $\mathcal{A}$,

$$E_{\mathbf{X},\mathbf{r}}[u_i((\sigma_i, \boldsymbol{\sigma}'_{-i}), \mathbf{r}, \mathcal{A})] \;\;(\;\; +\epsilon(k) \;\;) \;\; < E_{\mathbf{X},\mathbf{r}}[u_i((\sigma_i', \boldsymbol{\sigma}'_{-i}), \mathbf{r}, \mathcal{A})].$$

Let $\boldsymbol{\Sigma}^0$ be the set of strategies for $\mathcal{R}$. For $\ell \ge 1$, let $\boldsymbol{\Delta}^\ell$, $\boldsymbol{\Sigma}^\ell$ be defined as follows:

$$\Delta_i^\ell = \{\sigma_i : \exists \sigma_i' \in \Sigma_i^{\ell-1} \text{ such that } [\sigma_i < \sigma_i'|_{(\epsilon)} \boldsymbol{\Sigma}^{\ell-1}]\} \text{ and } \Sigma_i^\ell = \Sigma_i^{\ell-1} - \Delta_i^\ell.$$

We say that a strategy σ for party P_i survives iterated deletion of (computationally ϵ-)weakly dominated strategies *if for all ℓ, $\sigma \notin \Delta_i^\ell$.*

We finally define conditions that we wish a protocol to satisfy.

Definition 8 ((Computationally) t-secure preferred protocol for function f). *Given $\mathcal{P}$ receiving inputs from $\mathbf{X}$, with channel structure $\mathcal{C}(\mathcal{P})$ and protocol utility functions $\mathbf{u}$, a vector of (PPT) strategies $\boldsymbol{\sigma}^*$ constitutes a* (computationally) t-secure preferred protocol *for (polynomial-time computable)* $\mathbf{f}$ *under input distribution* $\mathbf{X}$ *if it has the following properties:*

Correctness and security: *As is standard for multi-party computation [5], (static) correctness and security is defined by requiring that for every (PPT) adversary $\mathcal{A}$, $\mathcal{A} \subset \mathcal{P}$, $|\mathcal{A}| \le t$, there exists a (PPT) simulator S of comparable computational ability such that for all $\mathbf{x} \in \mathbf{X}$, the joint distribution of $(\boldsymbol{OUT}_{\mathcal{R}}(\boldsymbol{\sigma}^*_{\mathcal{R}}(\mathbf{x}_{\mathcal{R}}), \mathbf{r}, \mathcal{A})$, $VIEW_{\mathcal{A}}(\boldsymbol{\sigma}^*_{\mathcal{R}}(\mathbf{x}_{\mathcal{R}}), \mathbf{r}, \mathcal{A}))$ is indistinguishable* *from the distribution sampled from as follows. S is interactive and obtains the following inputs in the following order: First, $\mathbf{x} \leftarrow \mathcal{A}$, and S is given $\mathbf{x}_{\mathcal{A}}$. Then S may output y, in which case the sample of the distribution is $(\perp_{\mathcal{R}}, y)$, or may produce query $\mathbf{x}'_{\mathcal{A}}$. In the latter case, in response to the query, S is given $\mathbf{f}_{\mathcal{A}}(\mathbf{x}')$, where $\mathbf{x}'_{\mathcal{R}} = \mathbf{x}_{\mathcal{R}}$. Finally, S outputs y. Then the sample of the distribution is $(\mathbf{f}_{\mathcal{R}}(\mathbf{x}'), y)$. Moreover, we require that there exist an adversary $\mathcal{A}$ (namely, one that follows $\sigma^*_{\mathcal{A}}$ other than for possible substitution of $\mathbf{x}'_{\mathcal{A}}$ for the inputs $\mathbf{x}_{\mathcal{A}}$), such that $\boldsymbol{OUT}_{\mathcal{R}}(\boldsymbol{\sigma}^*_{\mathcal{R}}(\mathbf{x}_{\mathcal{R}}), \mathbf{r}, \mathcal{A}) \neq \perp_{\mathcal{R}}$.*

Nash equilibrium: *(For all constants c,) $\boldsymbol{\sigma}^*(\mathbf{x})$ is a (computational k^{-c}-)Nash equilibrium for any partitioning $\mathcal{P} = \mathcal{R} \cup \mathcal{A}$.*

Survival condition: *(For all constants c,) for any partitioning $\mathcal{P} = \mathcal{R} \cup \mathcal{A}$, for any $P_i \in \mathcal{R}$, the strategy $\sigma^*_i(x_i)$ survives iterated deletion of (computational k^{-c}-)weakly dominated strategies.*

Discussion of Definitional Framework. The definitions above are a first step in reconciling the definitional framework of SMPC with that of game theory. Our starting point in this work was our main protocol of Section 4 and the definitions presented above are meant to capture the properties of our protocol. Further definitional study of MMPC may lead to interesting insights.

3 The Ideal-World Protocol: Using a Special Channel

In this section, we assume that all participants have access to a special (ideal) communication channel C that essentially does all the computational work: the function F_C that describes the operation of C is a randomized process that computes the target functions of the parties. Using this channel, the only way that the participants influence a protocol is by contributing an input to C at the beginning, and then at every round, notifying the channel whether they wish to participate at this round or not. Additionally, the channel C notifies the parties about the other parties' participation, and enables each P_i to broadcast any string v_i.

The Channel $C_{m,\mathbf{f},c} = (\mathcal{P}_I, F_C, \mathsf{state}, \mathcal{P}_O)$

Setting: The channel $C_{m,\mathbf{f},c}$ is a synchronous channel used by n participants $\mathcal{P} = \{P_1, \ldots, P_n\}$ (where up to t of them may be malicious). For channel $C_{m,\mathbf{f},c}$, $\mathcal{P}_I = \mathcal{P}_O = \mathcal{P}$, state is initially $\perp$, m and c are integer parameters and $\mathbf{f} = (f_1, \ldots, f_n)$ is a set of functions where each f_i takes as input n binary strings and outputs a binary string; operation function F_C is described below.

Initial round: In the initial round $r = 0$, the channel is activated and each participant P_i submits to $C_{m,\mathbf{f},c}$ an input x_i; if some P_i does not, then by convention, $x_i = \perp$. Define $\mathcal{P}_0 = \{P_i : x_i \neq \perp\}$, i.e., the set of participants that contributed inputs. $C_{m,\mathbf{f},c}$ announces the set $\mathcal{P}_0$ to all participants. $C_{m,\mathbf{f},c}$ stores $state = \{\mathbf{x}, r\}$. This signals the end of the initial round.

Round $r > 0$**:** Each round includes the following.

- The channel is activated and each participant P_i gives $C_{m,\mathbf{f},c}$ a value $z_i^r \in \{\texttt{Compute}, \texttt{Defect}\}$; if some P_i does not, then by convention, $z_i^r = \texttt{Defect}$. Let $\mathcal{P}_r = \{P_i : z_i^r = \texttt{Compute}\}$. Each participant may also contribute an additional value v_i; if P_i does not contribute, then $v_i = \perp$.
- If $c = 0$ or $r \bmod c \equiv 0$, then $C_{m,\mathbf{f},c}$ flips a coin to obtain a random bit b. Otherwise, $b = 0$.
- If $|\mathcal{P}_r| \geq m$, i.e., at least m participants wish to have the functions computed at this round, and $b = 1$, then $C_{m,\mathbf{f},c}$ sets $y_i = f_i(x_1, \ldots, x_n)$ for $1 \leq i \leq n$. Otherwise, $y_i = \perp$ for all i.
- $C_{m,\mathbf{f},c}$ sends to each participant P_i the value y_i, vector $\mathbf{v}$ and set $\mathcal{P}_r$, and stores r. This signifies the end of round r. Proceed to round $r + 1$.

Let $\mathcal{P} = \mathcal{R} \cup \mathcal{A}$, $|\mathcal{A}| = t$. We describe a protocol, a suggested strategy σ^C, for computing any NCC target function $\mathbf{f}$ over channel $C_{m,\mathbf{f},c}$ and next show that σ^C is an unconditionally t-secure preferred strategy according to Definition 8.

The Strategy $\sigma_i^C(x_i^*)$ for party $P_i \in \mathcal{R}$

Setting: For all $i \in [n]$, party P_i has input x_i^* and has access to channel $C_{m,\mathbf{f},c}$. Parties are also connected with each other via arbitrary communication links.
Initial round: Send $x_i = x_i^*$ to C. Receive the set $\mathcal{P}_0$ from C.
Round $r > 0$: In each round, P_i acts as follows.

- If any message has ever arrived over any communication channel other than $C_{m,\mathbf{f},c}$, never send any messages again (in any round), unless you know that the message is from the adversary $\mathcal{A}$, in which case ignore it.
- If $|\mathcal{P}_{r-1}| = n$, send $z_i^r = \texttt{Compute}$ to $C_{m,\mathbf{f},c}$, unless it was previously decided not to send any messages again. Otherwise, never send any messages again (in any round).
- Receive value y_i, vector $\mathbf{v}$ and set $\mathcal{P}_r$ from $C_{m,\mathbf{f},c}$.
- If $y_i \neq \perp$, output y_i and halt. Otherwise, proceed to round $r + 1$.

Lemma 1. *Let $\mathcal{P} = \mathcal{R} \cup \mathcal{A}$ and suppose that $\mathcal{A}$ is an entity that can be contacted. The following* puppet strategy *$\sigma_i^p(x_i)$ survives iterated deletion for all i: on input x_i, send x_i to $\mathcal{A}$; then follow instructions from $\mathcal{A}$ regarding messages to send to other parties and when a message from party P_j is received over any channel, forward it to $\mathcal{A}$; halt when receive $f_i(\mathbf{x})$ from $\mathcal{A}$ or when can compute it based on available information.*

Proof. (Idea) Consider the case when all P_ℓ, $\ell \neq i$, each follows $\sigma_\ell^p(x_\ell)$, and $\mathcal{A}$ acts such that either rewards or punishes P_i depending on whether or not P_i conforms with $\sigma_i^p(x_i)$, which can be detected by $\mathcal{A}$. It is within $\mathcal{A}$'s power to reward or punish P_i, because assuming that all $P_j \in \mathcal{R}$ follow $\sigma_j^p(x_j)$, $\mathcal{A}$ knows everyone's inputs. This scenario creates the situation where no other strategy can be strictly preferable to $\sigma_i^p(x_i)$ for P_i. □

We present sufficient conditions for a general class of strategies that use channel $C_{m,\mathbf{f},c}$ to survive iterated deletion even in the presence of side-channels. Let $\mathcal{P} = \mathcal{R} \cup \mathcal{A}$ be parties receiving inputs from $\mathbf{X}$ with protocol utility functions $\mathbf{u}$ and channel structure $\mathcal{C}(\mathcal{P})$ that includes $C_{m,\mathbf{f},c}$ and $\mathbf{f}$ be t-NCC target functions. We say that $\boldsymbol{\sigma}$ are *indefinite all-or-nothing strategies with visible deviations* if the following conditions are met: (1) $\boldsymbol{\sigma}(\mathbf{x})$ are strategies which, when executed jointly by $\mathcal{P}$, have the invariant that at every step of the computation, either each $P_i \in \mathcal{R}$ has already learned $f_i(\mathbf{x})$ and halted, or none of them have learned $f_i(\mathbf{x})$, and, moreover, all the messages they received so far are distributed independently of the value $f_i(\mathbf{x})$; (2) let E_s be the event that, under $\boldsymbol{\sigma}(\mathbf{x})$, after s steps of the protocol, no $P_i \in \mathcal{R}$ knows its $f_i(\mathbf{x})$; then there exists some $c(s) > 0$ such that for all views that P_i received by running $\sigma_i(x_i)$, it holds $0 < \Pr[E_{s+c(s)} \mid E_s, \mathit{VIEW}_i] < 1$, where the probability is over the random choices of the channels used; and (3) let the *signature* of a strategy given the inputs $\mathbf{x}$ and strategies of other parties and $\mathcal{A}$, consist of all the messages other parties received from P_i; then there exists a detection procedure that given a signature of $\sigma_i(x_i)$ can determine whether all of the actions of P_i were computed the same as if P_i was following $\sigma_i(x)$ for some x.

Lemma 2. *If $\boldsymbol{\sigma}(\mathbf{x})$ are indefinite all-or-nothing strategies with visible deviation, then any $\boldsymbol{\sigma}_i(\mathbf{x}_i)$, $i \in [n]$, survives iterated deletion of weakly dominated strategies.*

Proof. (Sketch) Consider a strategy $\sigma'(x_i)$. Suppose that there exist strategies $\boldsymbol{\tau}_{-i}$ and an adversary $\mathcal{A}$ such that, if P_i follows $\sigma'(x_i)$, and $\mathcal{P}_{-i}$ follow $\boldsymbol{\tau}_{-i}$, then P_i's utility is higher when following σ'_i than σ_i. Then there also exist strategies $\boldsymbol{\tau}'_{-i}$ that survive iterated deletion and an adversary $\mathcal{A}$ where P_i gets strictly more utility when following σ_i than when following σ'_i. Let strategies $\boldsymbol{\tau}'_{-i}$ be just $\boldsymbol{\sigma}^p_{-i}(\mathbf{x}_{-i})$, which by Lemma 1 survive iterated deletion. Suppose that $\mathcal{A}$ directs each $P_j \in \mathcal{R}_{-i}$ to send messages according to $\boldsymbol{\tau}_j$. Let d be the first deviation (from $\sigma_i(x_i)$) point in the protocol: at step d, if P_i is following σ'_i rather than σ_i, then with positive probability this will be detected. If at step d, P_i acts in a way that agrees with σ_i, then he is rewarded by $\mathcal{A}$ with $\mathbf{x}_{-i}$ and thus can compute the desired value $f_i(\mathbf{x})$. If at step d, P_i deviates, then $\mathcal{A}$ punishes P_i by directing all parties P_j not to send any more messages. Thus, in this setting and given that σ_i is an indefinite all-or-nothing strategy, following some $\sigma_i(x)$ gives better utility than $\sigma'_i(x_i)$ and no strategy σ' can weakly dominate all $\sigma_i(x)$ after any number of rounds of iterated deletion. Finally, following $\sigma_i(x_i)$ cannot be weakly dominated by following any $\sigma_i(x')$ where $x' \neq x_i$, because $\mathbf{f}$ is t-NCC. □

Theorem 1. *Given $\mathcal{P} = \mathcal{R} \cup \mathcal{A}$ receiving inputs from $\mathbf{X}$, with channel structure $\mathcal{C}(\mathcal{P}) \supseteq C_{m,\mathbf{f},c}$ and protocol utility functions $\mathbf{u}$, strategies $\boldsymbol{\sigma}^C(\mathbf{x})$ constitute an unconditionally t-secure preferred protocol for any t-NCC target function $\mathbf{f}$.*

Proof. (Sketch) Correctness and security hold for strategy $\boldsymbol{\sigma}^C(\mathbf{x})$ in a straightforward way, since our working channel $C_{m,\mathbf{f},c}$ is an ideal one. The Nash equilibrium condition is satisfied: given that all other parties follow $\boldsymbol{\sigma}^C(\mathbf{x})$, P_i has no incentive to deviate ($\mathbf{f}$ is t-NCC). Finally, strategy $\boldsymbol{\sigma}^C(\mathbf{x})$ survives iterated deletion, by Lemma 2 and because it is a vector of indefinite all-or-nothing strategies with visible deviations: at any round, either each $P_i \in \mathcal{R}$ already knows $f_i(\mathbf{x})$ and has halted or no party knows $f_i(\mathbf{x})$; also, no party has any information related to the termination of the current run; finally, all possible deviations from

$\boldsymbol{\sigma}_i^C(\mathbf{x})$ are visible given some strategy τ of the other parties that appropriately uses $C_{m,\mathbf{f},c}$ or some other channel. □

Remark. Strategy $\boldsymbol{\sigma}^C$ aborts in case of side-channel communication, so that any subset of rational parties has a disincentive to collude and thus exclude other parties. The price is that even one malicious party can cause the entire protocol to abort. In the full paper, we discuss a modified protocol and the issue of adversarial abort as it is related to covert channels and the survival condition.

4 The Real-World Protocol: Using Secure MPC

In this section we present and analyze our main protocol for computing any function in MMPC. Our protocol $\boldsymbol{\sigma}$ requires only a synchronous broadcast channel. The main idea is to implement the special channel $C_{m,\mathbf{f},c}$ using secure multi-party computation, where in each round the parties compute secret shares of either a useless value or the outputs $\mathbf{f}(\mathbf{x})$, and then they compute the target function by performing a global broadcast of such shares. At the same time, the parties conform to the protocol $\boldsymbol{\sigma}(\mathbf{x}) \triangleq \boldsymbol{\sigma}^C(\mathbf{x})$: if something goes not as expected (essentially, a `Defect` was ordered), this is always detected and parties abort the computation. We then use the results on $C_{m,\mathbf{f},c}$ and $\boldsymbol{\sigma}^C(\mathbf{x})$, for $t+1 < m = \lceil \frac{n}{2} \rceil$.

The Protocol $\boldsymbol{\sigma}$ – Strategy $\sigma_i(x_i)$ for party $P_i \in \mathcal{R}$

Inputs to the protocol, and general rules: $\mathbf{x} \leftarrow \mathbf{X}$, each P_i receives x_i, and security parameter 1^k. If P_i receives any message on any channel other than the broadcast channel at any point in the computation, it aborts unless it knows that the message is from the adversary, in which case it ignores it.

Setup phase: In the setup phase, all the parties jointly agree on a random string CRS of length $\ell(k)$ (to be defined later), and a public-key infrastructure with PK_i for each P_i. This is done as follows:

1. Each P_i chooses a random string CRS_i of length $\ell(k)$ and broadcasts a commitment $Com_i \leftarrow Commit(1^k, CRS_i)$ (where we use a computationally hiding, unconditionally binding cryptographic commitment) and a public key PK_i. If P_i detects that P_j failed to broadcast, then P_i aborts.
2. In turn, each P_i proves to each P_j over the broadcast channel that he knows the opening of the commitment Com_i, and a secret key corresponding to his public key PK_i, using a zero-knowledge proof of knowledge protocol with security parameter k. P_j broadcasts whether he accepts or rejects the proof. If P_l sees that some P_j does not accept the proof of some P_i (or fails to broadcast his decision), then P_l aborts.
3. Each P_i broadcasts the value CRS_i. If P_i sees that some P_j failed to broadcast, then P_i aborts.
4. In turn, each P_i proves to each P_j, using a zero-knowledge proof protocol over the broadcast channel, that the value he broadcasted corresponds to his commitment Com_i. After this, each P_j broadcasts whether he accepts or rejects the proof. If P_l sees that some P_j does not accept the proof of some P_i (or fails to broadcast his decision), then P_l aborts.

5. Obtain $CRS = \oplus_{i=1}^{n} CRS_i$, and $PKI = (PK_1, \ldots, PK_n)$.

The input phase: Parse $CRS = CRS_{COM} \circ CRS_{MPC} \circ CRS_{NIZK}$. Let $SECom$ be a simulatable and extractable commitment scheme [2] (which requires CRS_{COM} as input). Each P_i broadcasts $z_i = SECom(CRS_{COM}, x_i, r_i^{SEC})$, where r_i^{SEC} is the randomness needed to form the commitment. If P_j sees that some P_i failed to broadcast a valid commitment, P_j aborts.

The MPC phase: As a result of this phase, we want P_i to obtain m-out-n secret shares of either the strings $y_i = 0^{p_i}$ for $1 \leq i \leq n$ or the strings $y_i = out_i = f_i(\mathbf{x})$, where $p_i = |out_i|$ is known ahead of time. We want each type of output to be equally likely. This is done as follows: P_i chooses a random r_i^{MPC} (of appropriate length, to be specified later) and contributes input $(x_i, r_i^{SEC}, r_i^{MPC})$ to a secure multi-party protocol for computing an n-input function $g_{PKI,\mathbf{z}}$ over the broadcast channel [3], using CRS_{MPC} as a common random string.

Function $g_{PKI,\mathbf{z}}$ operates as follows on input $\{(x_i, r_i^{SEC}, r_i^{MPC}) : 1 \leq i \leq n\}$:

- Check that for all $1 \leq i \leq n$, $z_i = SECom(CRS_{COM}, x_i, r_i^{SEC})$. If for some i, the check fails, output the n-bit string y where $y_i = 0$ iff the check failed for z_i.
- Compute $r = \oplus_{i=1}^{n} r_i^{MPC}$. Parse $r = R_1 \circ R_2 \circ R_3$, where $|R_1| = 1$, and the lengths of R_2 and R_3 will be clear from the sequel.
- If $R_1 = 0$, then for each $1 \leq i \leq n$, come up with an m-out-n secret sharing of 0, and let $y_{i,j}$ be the j'th share. Otherwise, for each $1 \leq i \leq n$, come up with an m-out-n secret sharing of $1 \circ f_i(\mathbf{x})$, and let $y_{i,j}$ be the j'th share. Secret sharing requires randomness: use R_2 as the random tape for this step.
- Use R_3 as the random tape for forming n^2 ciphertexts as follows: $c_{i,j} = Enc(PK_j, y_{i,j})$; output these ciphertexts.

Suppose it takes $c - 1$ rounds of computation over the broadcast channel to jointly compute g. If at any round, P_j notices that P_i did not follow the protocol correctly[1] then P_j aborts. If the output of the computation indicates that some party contributed incorrect inputs, then P_j also aborts.

The possible reconstruction phase: Each P_i broadcasts a message of the form $(\{d_{j,i} : 1 \leq j \leq n\}, \pi)$, where π is a non-interactive simulation-sound zero-knowledge proof [4] that each $d_{j,i} = Enc(PK_j, y_{j,i}, r_{j,i})$ where $y_{j,i}$ is the correct decryption of $c_{j,i}$ under the public key PK_i, and $r_{j,i}$ are the cointosses for probabilistic encryption Enc. If P_i receives m messages with valid proofs, then he decrypts all ciphertexts $\{d_{i,j}\}$ addressed to him to obtain m shares either of 0 or of $1 \circ f_i(\mathbf{x})$. In the former case, if P_i received fewer than n messages with valid proofs, abort; otherwise go back to the MPC phase. In the latter case, output $f_i(\mathbf{x})$.

We analyze protocol $\boldsymbol{\sigma}$ in the mixed-behavior model and present our main result in the case where parties $\mathcal{P} = \mathcal{R} \cup \mathcal{A}$ are computationally bounded. We

[1] Recall that, following Goldreich, Micali and Wigderson [6], Canetti *et al.*'s protocol allows every party to detect that another party deviated from the protocol.

impose the following assumptions on protocol utilities in the computational setting: (1) if different runs produce indistinguishable views for P_i, this makes only a negligible difference to satisfaction measure μ_i; (2) negligible differences in satisfaction measures of all $P_j \in \mathcal{R}$ yield negligible differences in P_i's utility u_i.

Definition 9 (Computational satisfaction assumption). *Protocol utility function* $\mathbf{u}$ *with associated satisfaction* $\boldsymbol{\mu}$ *satisfies the* computational satisfaction assumption, *if for all non-negligible functions* ϵ, *there exists some negligible function* ν *such that* $\Pr_s[\mu_i(V_1) - \mu_i(V_2) > \epsilon(k)] = \nu(k)$, *where* $V_1 \leftarrow D_1(1^k)$ *and* $V_2 \leftarrow D_2(1^k)$ *are views that are indistinguishable by probabilistic algorithms with running time polynomial in* k. *Moreover, function* $u_i(R) = u'_i(\mu_1(R), ..., \mu_n(R))$ *has the property that, for any negligible function* $\nu_1(k)$, *there exists a negligible* $\nu_2(k)$ *such that if* R *and* R' *are such that* $\mu_i(R) \leq \mu_i(R') \leq \mu_i(R) + \nu_1(k)$, *and for all* $j \neq i$, $\mu_j(R) = \mu_j(R')$, *then* $u_i(R) \leq u_i(R') \leq u_i(R) + \nu_2(k)$.

Theorem 2 (Main result). *Given* $\mathcal{P} = \mathcal{R} \cup \mathcal{A}$ *receiving inputs from* $\mathbf{X}$, *with channel structure* $\mathcal{C}(\mathcal{P})$ *that includes a synchronous broadcast channel and protocol utility function* $\mathbf{u}$ *satisfying the computational satisfaction assumption, strategies* $\boldsymbol{\sigma}(\mathbf{x})$ *constitute a computational* t*-secure preferred protocol for any* t*-NCC target function* $\mathbf{f}$.

To prove our main result, we: (1) define a notion of computational reducibility among strategies designed for different channel structures, (2) show that computational reducibility between strategies results in computational equivalence in gained utilities, (3) show that $\boldsymbol{\sigma}^C$ executed over $\mathcal{C}(\mathcal{R}) \supseteq C_{m,\mathbf{f},c}$ is reduced to $\boldsymbol{\sigma}$ executed over any channel structure that includes the synchronous broadcast channel, thus proving that $\boldsymbol{\sigma}$ is an ϵ-Nash equilibrium, and (4) show that $\boldsymbol{\sigma}$ survives deletion.

Definition 10 (Computational reducibility). *Suppose* $\mathcal{P} = \mathcal{R} \cup \mathcal{A}$, $P_i \in \mathcal{R}$. *Let* $\mathcal{C} = \mathcal{C}_1 \cup \{C\}$, $\mathcal{C}' = \mathcal{C}_1 \cup \{C'\}$ *be channel structures available to* $\mathcal{P}$, *for some set of channels* $\mathcal{C}_1$ *and some additional channels* C *and* C'. *Let* $(\boldsymbol{\sigma}_{\mathcal{R}})_{-i}$ *and* $(\boldsymbol{\sigma}'_{\mathcal{R}})_{-i}$ *be sets of strategies corresponding to* $\mathcal{C}$ *and* $\mathcal{C}'$, *respectively. Let* $\mathcal{A}$ *be an adversary for channel structure* $\mathcal{C}$. *We say that the probabilistic polytime simulator* $S = (S_0, S_1, S_2, S_3)$ computationally reduces *setting* $(\mathcal{C}', (\boldsymbol{\sigma}'_{\mathcal{R}})_{-i})$ *to setting* $(\mathcal{C}, (\boldsymbol{\sigma}_{\mathcal{R}})_{-i})$ *if for all* $\mathbf{x}$, *PPT strategies* τ_i, $j \in [n]$ *and PPT adversaries* $\mathcal{A}$, $S_3(j, \mathit{VIEW}_j((S_1(\tau_i, x_i, s), (\boldsymbol{\sigma}'_{\mathcal{R}})_{-i}((\mathbf{x}_{\mathcal{R}})_{-i})), \mathbf{r}, S_2(\mathcal{A}, i, \mathbf{x}_{\mathcal{A}}, s)), s)$, *where* $s \leftarrow S_0(1^k)$, *is a distribution that is computationally indistinguishable from* $\mathit{VIEW}_j((\tau_i(1^k, x_i), (\boldsymbol{\sigma}_{\mathcal{R}})_{-i}((\mathbf{x}_{\mathcal{R}})_{-i})), \mathbf{r}, \mathcal{A})$. *In this setting, we say that* S computationally translates *strategy* $\tau_i(x_i)$ *and adversary* $\mathcal{A}(\mathbf{x}_{\mathcal{A}})$ *into strategy* $S_1(\tau_i, x_i, s)$ *and adversary* $S_2(\mathcal{A}, \mathbf{x}_{\mathcal{A}}, s)$.

That is, S_1 transforms strategy τ_i designed for communication structure $\mathcal{C}$ into a strategy designed for communication structure $\mathcal{C}'$, while S_2 does the same with $\mathcal{A}$. It is possible, however, that the resulting strategy and adversary talk to each other. Then S_3 translates the resulting view into a view that is computationally close to the view that P_i could have gotten if it were to run τ_i in $\mathcal{C}$

with adversary $\mathcal{A}$, or into a view that any other rational P_j would have gotten if it were to run σ_i in $\mathcal{C}$ with adversary $\mathcal{A}$ and τ_i, instead of running σ'_i in $\mathcal{C}'$.

Lemma 3. *Under the computational satisfaction assumption, if: (1)* $\mathbf{f}$ *is* t*-NCC, and is the computational target function for the given set of utility functions* $\mathbf{u}$*; (2) for all* $i \in [n]$*, for strategies* $\boldsymbol{\sigma}$ *and* $\boldsymbol{\sigma}'$*, simulator* $S = (S_0, S_1, S_2, S_3)$ *computationally reduces setting* $(\mathcal{C}', (\sigma'_{\mathcal{R}})_{-i})$ *to the setting* $(\mathcal{C}, (\sigma_{\mathcal{R}})_{-i})$*; (3) for all* $i \in [n]$*,* x_i *and* s*,* $S_1(\sigma_i, x_i, s) = \sigma'_i(x_i)$*; (4) for all* $\mathcal{A}$*,* i*, probabilistic poly-time* τ_i*,* $j \neq i$*,* $P_j \in \mathcal{R}$*,* $\mu_j(S_3(j, \mathit{VIEW}_j((S_1(\tau_i, x_i, s), (\boldsymbol{\sigma}'_{\mathcal{R}})_{-i}((\mathbf{x}_{\mathcal{R}})_{-i})), \mathbf{r}, S_2(\mathcal{A}, i, \mathbf{x}_{\mathcal{A}}, s)), s))$ *is equal to* $\mu_j(\mathit{VIEW}_j((S_1(\tau_i, x_i, s), (\boldsymbol{\sigma}'_{\mathcal{R}})_{-i}((\mathbf{x}_{\mathcal{R}})_{-i})), \mathbf{r}, S_2(\mathcal{A}, i, \mathbf{x}_{\mathcal{A}}, s)))$*, that is, no matter what* $\mathcal{A}$ *and* P_i *do, when transforming the view from running* σ'_j *to one for running* σ_j*, the simulator* S_3 *did not throw out any information relevant to* μ_j*. Then, if* $\boldsymbol{\sigma}'$ *is a Nash equilibrium, then* $\boldsymbol{\sigma}$ *is* ϵ*-Nash equilibrium.*

Proof. (Sketch) For brevity, let us denote $(\boldsymbol{\sigma}_{\mathcal{R}})_{-i}((\mathbf{x}_{\mathcal{R}})_{-i})$ simply as $\boldsymbol{\sigma}_{-i}$. Consider strategy τ_i, adversary $\mathcal{A}$, inputs $\mathbf{x}$. Then we know that:

$$\begin{aligned}
&\mu_i(\mathit{VIEW}_i((\tau_i(x_i), \boldsymbol{\sigma}_{-i}), \mathbf{r}, \mathcal{A}(\mathbf{x}_{\mathcal{A}})))\\
&\leq \mu_i(S_3(i, \mathit{VIEW}_i((S_1(\tau_i, x_i, s), \boldsymbol{\sigma'}_{-i}), \mathbf{r}', S_2(\mathcal{A}, i, \mathbf{x}_{\mathcal{A}}, s)), s)) + \nu_1(k)\\
&\leq \mu_i(\mathit{VIEW}_i((S_1(\tau_i, x_i, s), \boldsymbol{\sigma'}_{-i})), \mathbf{r}', S_2(\mathcal{A}, i, \mathbf{x}_{\mathcal{A}}, s))) + \nu_1(k).
\end{aligned}$$

The first inequality follows by the definition of computational reducibility and the computational satisfaction assumption; the second inequality holds because S_3 may destroy some relevant information. Next, in channel structure $\mathcal{C}$, we measure μ_j for a party P_j following strategy σ_j while P_i is following τ_i. From the definition of computational reducibility, $\mu_j(\mathit{VIEW}_j((\tau_i(x_i), \boldsymbol{\sigma}_{-i}), \mathbf{r}, \mathcal{A}(\mathbf{x}_{\mathcal{A}})))$ $\approx$ $\mu_j(S_3(j, \mathit{VIEW}_j((S_1(\tau_i, x_i, s), \boldsymbol{\sigma'}_{-i}), \mathbf{r}', S_2(\mathcal{A}, i, \mathbf{x}_{\mathcal{A}}, s)), s))$. This is approximately equal to $\mu_j(\mathit{VIEW}_j((S_1(\tau_i, x_i, s), \boldsymbol{\sigma'}_{-i}), \mathbf{r}', S_2(\mathcal{A}, i, \mathbf{x}_{\mathcal{A}}, s)))$, by the conditions of the lemma. Thus, for every strategy τ_i and adversary $\mathcal{A}$ in channel structure $\mathcal{C}$, there exist a strategy τ'_i (i.e., $S_1(\tau_i, x_i, s)$) and adversary $\mathcal{A}'$ (i.e., $S_2(\mathcal{A}, i, \mathbf{x}_{\mathcal{A}}, s)$), in channel structure $\mathcal{C}'$ such that:

$$\begin{aligned}
&u_i((\tau_i(x_i), \boldsymbol{\sigma}_{-i}), \mathbf{r}, \mathcal{A}(\mathbf{x}_{\mathcal{A}})) \leq u_i((\tau'_i(x_i), \boldsymbol{\sigma'}_{-i}), \mathbf{r}', \mathcal{A}'(\mathbf{x}_{\mathcal{A}})) + \nu_2(k)\\
&\leq u_i(\boldsymbol{\sigma}'_{\mathcal{R}}(\mathbf{x}_{\mathcal{R}}), \mathbf{r}', \mathcal{A}'(\mathbf{x}_{\mathcal{A}})) + \nu_2(k) = u_i(\boldsymbol{\sigma}_{\mathcal{R}}(\mathbf{x}_{\mathcal{R}}), \mathbf{r}', \mathcal{A}(\mathbf{x}_{\mathcal{A}})) + \nu_2(k),
\end{aligned}$$

where the first inequality holds by the computational satisfaction assumption. The second inequality follows because $\boldsymbol{\sigma}'$ is a Nash equilibrium, while the last equality follows by condition (3) of the lemma and Definition 9. □

Lemma 4. *If* $\mathbf{f}$ *is* t*-NCC, then there exists a simulator* $S = (S_0, S_1, S_2, S_3)$ *satisfying the conditions of Lemma 3, reducing the (ideal) strategy* $\boldsymbol{\sigma'}$ *in channel structure* $\mathcal{C}'$ *that includes (ideal) channel* $C_{m,\mathbf{f},c}$ *to the (real) strategy* $\boldsymbol{\sigma}$ *in the channel structure* $\mathcal{C}$ *that includes a synchronous broadcast channel.*

Proof. (Sketch) The main idea is that, for rational P_i following strategy τ_i instead of σ_i, we will have simulator S_1 broadcast his input x_i and his strategy τ_i over the ideal channel $C_{m,\mathbf{f},c}$, while S_2 broadcasts the adversary's inputs and descriptions. This will ensure that the view from the resulting strategy for the ideal channel will enable S_3 to simulate what P_i and $\mathcal{A}$ would do, and create a correctly distributed view for each $P_j \in \mathcal{R}$, $j \neq i$. S_1 and S_2 will then collaboratively simulate the views for P_i and $\mathcal{A}$ using the corresponding simulators and extractors for ZK proofs of knowledge, commitments, and multi-party computation. S_3 will simply output the simulated views created by S_1 and S_2. □

Lemma 5. *Under the computational satisfaction assumption, if* $\mathbf{f}$ *is a t-NCC computational target function, then protocol* $\boldsymbol{\sigma}$ *survives iterated deletion of computational* k^{-c}*-weakly dominated strategies.*

Proof. (Sketch) We appropriately adapt the proof of Lemma 2 for strategy $\boldsymbol{\sigma}$, considering all different ways in which P_i may depart from σ_i. □

Acknowledgements. We thank the anonymous reviewers and Jonathan Katz for their constructive comments, and also Silvio Micali and Moni Naor for helpful discussions. Anna Lysyanskaya is supported by NSF Career Grant CNS–0374661 and Nikos Triandopoulos by NSF Grants CCF–0311510 and IIS–0324846.

References

[1] I. Abraham, D. Dolev, R. Gonen, and J. Halpern. Distributed computing meets game theory: Robust mechanisms for rational secret sharing and multiparty computation. In *Proc. 25th ACM PODC*, 2006. (To appear.)
[2] R. Canetti and M. Fischlin. Universally composable commitments. In *Proc. CRYPTO 2001*, pages 19–40, 2001.
[3] R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. Universally composable two-party and multi-party secure computation. In *Proc. 34th ACM Symposium on Theory of Computing (STOC)*, pages 494–503, 2002.
[4] A. De Santis, G. Di Crescenzo, R. Ostrovsky, G. Persiano, and A. Sahai. Robust non-interactive zero knowledge. In *Proc. CRYPTO 2001*, pages 566–598, 2001.
[5] O. Goldreich. *Foundations of Cryptography.* Cambridge University Press, 2004.
[6] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *Proc. 19th ACM Symposium on Theory of Computing (STOC)*, pages 218–229, 1987.
[7] S. D. Gordon and J. Katz. Rational secret sharing, revisited, 2006. Manuscript available at http://eprint.iacr.org/2006/142. (To appear at *SCN 2006.*)
[8] J. Halpern and V. Teague. Rational secret sharing and multiparty computation: extended abstract. In *Proc. 36th ACM Symposium on Theory of Computing (STOC)*, pages 623–632, 2004.
[9] S. Izmalkov, M. Lepinski, and S. Micali. Rational secure computation and ideal mechanism design. In *Proc. 46th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 585–594, 2005.
[10] M. Lepinksi, S. Micali, and abhi shelat. Collusion-free protocols. In *Proc. 37th ACM Symposium on Theory of Computing*, pages 543–552, 2005.
[11] M. Lepinski, S. Micali, C. Peikert, and A. Shelat. Completely fair SFE and coalition-safe cheap talk. In *Proc. 23rd ACM PODC*, pages 1–10, 2004.
[12] A. Shamir. How to share a secret. *Comm. of the ACM*, 22(11):612–613, 1979.
[13] Y. Shoham and M. Tennenholtz. Non-cooperative computation: Boolean functions with correctness and exclusivity. *Theoretical Comp. Science*, 343(2):97–113, 2005.