Towards a Game Theoretic View of Secure Computation

Gilad Asharov[1,⋆], Ran Canetti[2,⋆⋆], and Carmit Hazay[3]

[1] Department of Computer Science, Bar-Ilan University, Israel
asharog@cs.biu.ac.il
[2] Department of Computer Science, Tel-Aviv University, Israel
canetti@tau.ac.il
[3] Department of Computer Science, Aarhus University, Denmark
carmit@cs.au.dk

Abstract. We demonstrate how Game Theoretic concepts and formalism can be used to capture cryptographic notions of security. In the restricted but indicative case of two-party protocols in the face of malicious fail-stop faults, we first show how the traditional notions of secrecy and correctness of protocols can be captured as properties of Nash equilibria in games for rational players. Next, we concentrate on fairness. Here we demonstrate a Game Theoretic notion and two different cryptographic notions that turn out to all be equivalent. In addition, we provide a simulation based notion that implies the previous three. All four notions are weaker than existing cryptographic notions of fairness. In particular, we show that they can be met in some natural setting where existing notions of fairness are provably impossible to achieve.

1 Introduction

Both Game Theory and the discipline of cryptographic protocols are dedicated to understanding the intricacies of collaborative interactions among parties with conflicting interests. Furthermore, the focal point of both disciplines is the same, and is algorithmic at nature: designing and analyzing algorithms for parties in such collaborative situations. However, the two disciplines developed very different sets of goals and formalisms. Cryptography focuses on designing algorithms that allow those who follow them to interact in a way that guarantees some basic concrete properties, such as secrecy, correctness or fairness, in face of adversarial, malicious behavior. Game Theory is more open-ended, concerning itself with understanding algorithmic behaviors of "rational" parties with well-defined goals in a given situation, and on designing rules of interaction that will "naturally" lead to behaviors with desirable properties.

Still, in spite of these differences, some very fruitful cross fertilization between the two disciplines has taken place (see e.g. [26,8]). One very natural

⋆ Supported by the European Research Council as part of the ERC project LAST.
⋆⋆ Supported by the Check Point Institute for Information Security, BSF, ISF, and Marie Curie grants.

K.G. Paterson (Ed.): Eurocrypt 2011, LNCS 6632, pp. 426–445, 2011.

direction is to use cryptographic techniques to solve traditional Game Theoretic problems. In particular, the works of Dodis et al. [7], Ismalkov et al. [25,24], Abraham et al. [1] and Halpern and Pass [23] take this path and demonstrate how a mutli-party protocol using cryptographic techniques can be used to replace a trusted correlation device or a mediator in mechanism design.

Another line of research is to extend the traditional Game Theoretic formalisms to capture, within the context of Game Theory, cryptographic concerns and ideas that take into account the fact that protocol participants are computationally bounded, and that computational resources are costly [7,23,21].

Yet another line of work is aimed at using Game Theoretic concepts and approach to amend traditional cryptographic goals such as secure and fair computation. A focal point in this direction has been the concept of rational fair exchange of secrets (also known as *rational secret sharing*) [22,19,29,27,28,30,9,3]. Here the goal is to design a protocol for exchanging secrets in a way that "rational players" will be "interested" in following the protocol, where it is assumed that players are interested in learning the secret inputs of the other players while preventing others from learning their own secrets. In fact, it is assumed that the participants have specific preferences and some quantitative prior knowledge on these preferences of the participants is known to the protocol designer. Furthermore, such prior knowledge turns out to be essential in order to get around basic impossibility results [3,6].

These ingenious works demonstrate the benefit in having a joint theory of protocols for collaborative but competing parties; but at the same time they underline the basic incompatibility in the two formalisms. For instance, the (primarily Game Theoretic) formalisms used in the works on rational secret sharing do not seem to naturally capture basic cryptographic concepts, such as semantic security of the secrets. Instead, these works opt for more simplistic notions that are not always compatible with traditional cryptographic formalisms. In particular, existing modeling (that is used both by constructions and by impossibility results) treats the secret as an atomic unit and consider only the case where the parties either learnt or did not learn the secret *entirely.* Unlike traditional cryptographic modeling, the option where *partial information* about the secret is leaked through the execution is disregarded.

This work. We relate the two *formalisms.* In particular we show how Game Theoretic formalism and concepts can be used to capture traditional cryptographic security properties of protocols. We concentrate on the setting of two-party protocols and fail-stop adversaries. While this setting is admittedly limited, it does incorporate the core aspects of secrecy, correctness and fairness in face of malicious (i.e., not necessarily "rational") aborts.

In this setting, we first show Game Theoretic notions of secrecy and correctness that are *equivalent,* respectively, to the standard cryptographic notions of secret and correct evaluation of deterministic functions in the fail-stop setting (see e.g [12]). We then turn to capturing fairness. Here the situation turns out to be more intricate. We formulate a natural Game Theoretic notion of fairness, and observe that it is *strictly weaker* than existing cryptographic notions of fair two-party function evaluation. We then formulate new cryptographic

notions of fairness that are *equivalent* to this Game Theoretic notion, and a simulation-based notion of fairness that implies the above three. Furthermore, we show that these new notions can indeed be realized in some potentially meaningful settings where traditional cryptographic notions are provably unrealizable.

The results in more detail. The basic idea proceeds as follows. We translate a given protocol into a set of games, in such a way that the protocol satisfies the cryptographic property in question *if and only if* a certain pair of strategies (derived from the protocol) are in a (computational) Nash equilibrium in each one of the games. This allows the cryptographic question to be posed (and answered) in Game Theoretic language. More precisely, given a protocol, we consider the (extensive form with incomplete information) game where in each step the relevant party can decide to either continue running the protocol as prescribed, or alteratively abort the execution. We then ask whether the pair of strategies that instruct the players to continue the protocol to completion is in a (computational) Nash equilibrium. Each cryptographic property is then captured by an appropriate set of utilities and input distributions (namely, distributions over the types). In particular:

Secrecy. A given protocol is secret (as in, e.g. [12]) if and only if the strategy that never aborts the protocol is in a computational Nash equilibrium with respect to the following set of utilities and distributions over the types. For each pair of values in the domain, we define a distribution that chooses an input for one party at random from the pair. The party gets low payoff if the two values lead to the same output value and yet the other party managed to guess which of the two inputs was used. It is stressed that this is the first time where a traditional cryptographic notion of secrecy (in the style of [16]) is captured in Game Theoretic terms. In particular, the works on rational secret sharing do not provide this level of secrecy for the secret. (Indeed, the solution approaches taken there need the secret to be taken from a large domain.)

Correctness. A protocol correctly computes a deterministic function if and only if the strategy that never aborts the protocol is in a computational Nash equilibrium with respect to the set of utilities where the parties get high payoff only if they output the correct function value on the given inputs (types), or abort before the protocol starts; in addition, the players get no payoff for incorrect output.

Fairness. Here we make the bulk of our contributions. We first recall the basic setting: Two parties interact by exchanging messages in order to valuate a function f on their inputs. The only allowed deviation from the protocol is abortion, in which event both parties learn that the protocol was aborted. Consequently, a protocol in this model should specify, in addition to the next message to be sent, also a prediction of the output value in case the execution is aborted. (Although the setting makes sense for any function, it may be helpful to keep in mind the *fair exchange* function, where the output of each party is the input of the other.)

Current notions of fairness for two party protocols in this model (e.g., [20,18]) require there to be a point in the computation where both parties move from a state of no knowledge of the output to a full knowledge of it. This is a strong notion, which is impossible to realize in many situations. Instead, we would like

to investigate more relaxed notions of fairness, which allow parties to gradually learn partial information on their desired outputs - but do so in a way that is "fair". Indeed, such an approach seems reasonable both from a game theoretic point of view (as a zero-sum game) and from a cryptographic point of view via the paradigm of gradual release (see e.g. [4,15,11,20,3] and the references within).

A first thing to note about such a notion of fairness is that it is sensitive to the potential prior knowledge that the parties may have on each other's inputs. Indeed, a "gradual release" protocol that is "fair" without prior knowledge may become "unfair" in a situation where one of the parties has far more knowledge about the possible values of the inputs of the second party than vice versa.

We thus explicitly model in our security notions the knowledge that each party has on the input of the other party. That is, we let each party has, in addition to its own input, some additional information on the input of the other party. Furthermore, to simplify matters and put the two parties on equal footing, we assume that the information that the parties have on the input of the other consists of two possible values for that input. That is, each party receives three values: its own input, and two possible values for the input of the other party. Indeed, such information naturally captures situations where the domain of possible inputs is small (say, binary). The formalism can also be naturally extended to deal with domains of small size which is larger than two.

We first sketch our Game Theoretic notion. We consider the following set of distributions over inputs (types): Say that a quadruple of elements (a_0, a_1, b_0, b_1) in the domain of function f is *valid* if for all $i \in \{0, 1\}$, $f(a_0, b_i) \neq f(a_1, b_i)$ and $f(a_i, b_0) \neq f(a_i, b_1)$. For each valid quadruple of values in the domain, we define a distribution that chooses an input for one party at random from the first two values, and an input for other party at random from the other two values. The utility function for a party is the following: When the party aborts the protocol, each party predicts its output. If the party predicts correctly and the other one does not, then it gets payoff +1. If it predicts incorrectly and the other party predicts correctly then it gets payoff -1. Else, it gets payoff 0. We say that a protocol is Game Theoretically Fair if the strategy that never aborts the protocol is in a computational Nash equilibrium with respect to the above utility, applied to both parties, and any distribution from the above family.

We then consider three different cryptographic notions of fairness and investigate their relationships with the above Game Theoretic notion.

- First, we formulate a simple "game based" notion of fairness that limits the gain of an arbitrary (i.e., not necessarily "rational") fail-stop adversary in a game that closely mimics the above Game Theoretic interaction. The main difference between the notions is that in the cryptographic setting the adversary is arbitrary, rather than rational. Still, we show that the notions are *equivalent.*
- Next, we show that this notion in fact corresponds to the natural concept of *gradual release.* That is, say that a protocol satisfies the *gradual release* property if at any round the probability of any party to predict its output increases only by a negligible amount. We show that a protocol is fair (as in

the above notions) *if and only if* it satisfies the gradual release property. We note that the notion of gradual release is in essence the basis of the classic protocols of Beaver, Goldwasser and Levin [4,15]. It has also been a key tool in the work of Asharov and Lindell [3]. Due to lack of space we do not present the definition of gradual release here; see full version [2] for a formal description.

– Then, we formulate an ideal-model based notion of fairness that allows for gradual release of secrets. In this notion the ideal functionality accepts a "sampling algorithm" M from the ideal-model adversary. The functionality then obtains the inputs from the parties and runs M on these inputs, and obtains from M the outputs that should be given to the two parties. The functionality then makes the respective outputs available to the two parties. (Ie, once the outputs are available, the parties can access them at any time.) The correctness and fairness guarantees of this interaction clearly depend on the properties of M. We thus require that M be both "fair" and "correct", in the sense that both parties get correct output with roughly equal (and substantial) probability. We then show that the new simulation based definition implies the gradual release notion. (We note that the converse does not necessarily hold with respect to secure computation in the fail-stop model, even disregarding fairness).

A positive result. Finally, we consider the realizability of this notion. Here, we first assert that the impossibility results of Cleve and Asharov and Lindell [6,3] hold even with respect to the new notions, as long as *both parties are required to receive an output.* We then observe that our notion is meaningful even in the case where the parties are not guaranteed to always learn the correct output when played honestly. Surprisingly, in cases where the parties learn the correct output with probability one half or smaller (i.e., correctness holds with probability between 0 and 1/2), our simulation-based notion of fairness is in fact *achievable* with no set-up or trusted third parties. We demonstrate a family of two-party protocols, parameterized by this correctness probability, that realize the new notion of fairness. For instance, for the case that correctness is guaranteed with probability one half, we design a fair protocol where with probability one half both parties obtain the correct output, and with probability one half both parties obtain an incorrect value. An alternative protocol makes sure that each party obtains a correct output with probability one half, and at each execution exactly one party obtains the correct output value. These scenarios were not known to be achievable before (not even by [18]), and may prove to be useful.

On the definitional choices. One question that comes to mind when considering our modeling is why use plain Nash equilibria to exhibit correspondence between cryptographic notions and Game Theoretic ones. Why not use, for instance, stronger notions such as Dominant Strategy, Survival Under Iterated Deletions, or Subgame Perfect equilibria. It turns out that in our setting of two party computation with fail-stop faults, Nash equilibria do seem to be the concept that most naturally corresponds to cryptographic secure protocols. In particular, in

the fail-stop case any Nash equilibrium is sub-game perfect, or in other words empty threats do not hold (see more discussion on this point in the next section).

Future work. One interesting challenge is extending the results of this work to the Byzantine case. For one, in the Byzantine case there are multiple cryptographic notions of security, including various variants of simulation based notions. Capturing these notions using Game Theoretic tools might shed light on the differences between these cryptographic notions. In particular, it seems that here the Game Theoretic formalism will have to be extended to capture arbitrary polynomial time strategies at each decision point.In particular, it seems likely that more sophisticated Game Theoretic solution concepts such as sub-game perfect equilibria and computational relaxations thereof [21,31] will be needed.

Another challenge is to extend the notions of fairness presented here to address also situations where the parties have more general, asymmetric a priori knowledge on each other's inputs, and to find solutions that use minimal trust assumptions on the system. Dealing with the multi-party case is another interesting challenge.

Organization. Section 2 presents cryptographic and Game Theoretic "solution concepts". Section 4 presents results regarding fairness: (i) the game theoretic notion, (ii) the equivalent cryptographic definition, (iii) a new simulation based definition and, (iv) the study of the fairness definition. The gradual release property, its relation to fairness and some more details are found in the full version [2].

2 The Model and Solution Concepts

We review some basic definitions that capture the way we model protocols (or, equivalently, strategies), as well as the solution concepts we will consider — both the cryptographic and the game theoretic ones. While most of these definitions are known, some are new to this work.

2.1 Cryptographic Definitions

We review some standard cryptographic definitions of security for protocols.

The Fail-Stop setting. The setting that we consider in this paper is that of two-party interaction in the presence of *fail-stop* faults. In this setting both parties follow the protocol specification exactly, with the exception that any one of the parties may, at any time during the computation, decide to stop, or *abort* the computation. Specifically, it means that fail-stop adversaries do not change their initial input for the execution, yet, they may arbitrarily decide on their output.

Cryptographic Security. We present game based definitions that capture the notions of privacy and correctness. We restrict attention to deterministic functions. By definition [12], the view of the ith party ($i \in \{0,1\}$) during an execution of π on (x_0, x_1) is denoted $\mathsf{view}_{\pi,i}(x_0, x_1, n)$ and equals $(x_i, r^i, m_1^i, ..., m_t^i)$, where r^i equals the contents of the ith party's *internal* random tape, and m_j^i represents the jth message that it received.

Definition 1 (Privacy). *Let f and π be as above. We say that π* privately computes *f if the following holds:*

1. *For every non-uniform* PPT *adversary $\mathcal{A}$ that controls party P_0*

$$\left\{\mathsf{view}_{\pi,\mathcal{A}(z),0}(x_0,x_1,n)\right\}_{x_0,x_1,x_1',y,z\in\{0,1\}^*,n\in\mathbb{N}} \stackrel{c}{\equiv} \left\{\mathsf{view}_{\pi,\mathcal{A}(z),0}(x_0,x_1',n)\right\}_{x_0,x_1,x_1',z\in\{0,1\}^*,n\in\mathbb{N}}$$

   where $|x_0|=|x_1|=|x_1'|$ and $f(x_0,x_1)=f(x_0,x_1')$.

2. *For every non-uniform* PPT *adversary $\mathcal{A}$ that controls party P_1*

$$\left\{\mathsf{view}_{\pi,\mathcal{A}(z),1}(x_0,x_1,n)\right\}_{x_0,x_0',x_1,z\in\{0,1\}^*,n\in\mathbb{N}} \stackrel{c}{\equiv} \left\{\mathsf{view}_{\pi,\mathcal{A}(z),1}(x_0',x_1,n)\right\}_{x_0,x_0',x_1,z\in\{0,1\}^*,n\in\mathbb{N}}$$

   where $|x_0|=|x_0'|=|x_1|$ and $f(x_0,x_1)=f(x_0',x_1)$.

Definition 2 (Correctness). *Let f and π be as above. We say that π* correctly computes *f if for all sufficiently large inputs x_0 and x_1 such that $|x_0|=|x_1|=n$, we have that $\Pr[\mathsf{output}_{\pi,i}\in\{\perp\circ\{0,1\}^*, f(x_0,x_1)\}]\geq 1-\mu(n)$, where $\mathsf{output}_{\pi,i}\neq\perp$ denotes the output returned by P_i upon the completion of π whenever the strategy of the parties is* `continue`*, and μ is a negligible function.*

2.2 Game Theoretic Definitions

We review the relevant concepts from Game Theory, and the extensions needed to put these concepts on equal footing as the cryptographic concepts. Traditionally, a 2-player *(normal form, full information)* game $\Gamma=(\{A_0,A_1\},\{u_0,u_1\})$ is determined by specifying, for each player P_i, a set A_i of possible actions and a utility function $u_i: A_0\times A_1\mapsto R$. Letting $A\stackrel{\text{def}}{=}A_0\times A_1$, we refer to a tuple of actions $a=(a_0,a_1)\in A$ as an outcome. The utility function u_i of party P_i expresses this player's preferences over outcomes: P_i prefers outcome a to outcome a' if and only if $u_i(a)>u_i(a')$. A *strategy* σ_i for P_i is a distribution on actions in A_i. Given a strategy vector $\sigma=\sigma_0,\sigma_1$, we let $u_i(\sigma)$ be the expected utility of P_i given that all the parties play according to σ. We continue with a definition of Nash equilibria:

Definition 3 (Nash equilibria for normal form, complete information games). *Let $\Gamma=(\{A_0,A_1\},\{u_0,u_1\})$ be as above, and let $\sigma=\sigma_0,\sigma_1$ be a pair of strategies as above. Then σ is in a* Nash equilibrium *if for all i and any strategy σ_i' it holds that $u_i(\sigma_0'',\sigma_1'')\leq u_i(\sigma)$, where $\sigma_i''=\sigma_i'$ and $\sigma_{1-i}''=\sigma_{1-i}$.*

The above formalism is also naturally extended to the case of *extensive form* games, where the parties take turns when taking actions. Another natural extension is to games with *incomplete information.* Here each player has an additional piece of information, called *type,* that is known only to itself. That is, the strategy σ_i now takes as input an additional value x_i. To extend the notion of Nash equilibria to deal with this case, it is assumed that an a priori distribution on the inputs (types) is known and fixed.

Definition 4 (Nash equilibria for extensive form, incomplete information games). *Let $\Gamma = (\{A_0, A_1\}, \{u_0, u_1\})$ be as above, and let D be a distribution over $(\{0,1\}^*)^2$. Also, let $\sigma = \sigma_0, \sigma_1$ be a pair of extensive-form strategies as described above. Then σ is in a* Nash equilibrium for D *if for all i and any strategy σ_i' it holds that $u_i(x_0, x_1, \sigma_0''(x_0), \sigma_1''(x_1)) \leq u_i(x_0, x_1, \sigma_0(x_0), \sigma_1(x_1))$, where (x_0, x_1) is taken from distribution D, $\sigma_i(x)$ denotes the strategy of P_i with type x, $\sigma_i'' = \sigma_i'$ and $\sigma_{1-i}'' = \sigma_{1-i}$.*

Extensions for the cryptographic model. We review the (by now standard) extensions of the above notions to the case of computationally bounded players. See e.g. [7,26] for more details. The first step is to model a strategy as an (interactive) probabilistic Turing machine that algorithmically generates the next move given the type and a sequence of moves so far. Next, in order to capture computationally bounded behavior (both by the acting party and, more importantly, by the other party), we move to an asymptotic treatment. That is, we consider an infinite sequence of games. The third and last step is to relax the notion of "greater or equal to" to "not significantly less than". This is intended to compensate for the small inevitable imperfections of cryptographic constructs. That is, we have:

Definition 5 (Computational Nash equilibria for extensive form, incomplete inf. games). *Let $\Gamma = (\{A_0, A_1\}, \{u_0, u_1\})$ be as above, and let $D = \{D_n\}_{n \in \mathbf{N}}$ be a family of distributions over $(\{0,1\}^*)^2$. Let $\sigma = \sigma_0, \sigma_1$ be a pair of* PPT *extensive-form strategies as described above. Then σ is in a* Nash equilibrium for D *if for all sufficiently large n's, all i and any* PPT *strategy σ_i' it holds that $u_i(n, x_0, x_1, \sigma_0''(n, x_0), \sigma_1''(n, x_1)) \leq u_i(n, x_0, x_1, \sigma_0(n, x_0), \sigma_1(n, x_1)) + \mu(n)$, where (x_0, x_1) is taken from distribution D_n, $\sigma_i(x, n)$ denotes the strategy of P_i with type x, $\sigma_i'' = \sigma_i'$ and $\sigma_{1-i}'' = \sigma_{1-1}$, and μ is a negligible function.*

Our setting. We consider the following setting: At each step, the relevant party can make a binary decision: Either *abort* the computation, in which case the other party is notified that an abort action has been taken, or else *continue running the protocol π scrupulously.* The traditional Game Theoretic modeling of games involving such "exogenous" random choices that are not controllable by the players involves, introduces additional players (e.g., "Nature") to the game. In our case, however, the situation is somewhat different, since the random choices may be secret, and in addition each player also has access to local state that is preserved throughout the interaction and may affect the choices. Specifically, an action may specify a (potentially randomized) algorithm and a configuration. The outcome of taking this action is that an output of running the said algorithm from the said configuration, is appended to the history of the execution, and the new configuration of the algorithm is added to the local history of the player. More formally:

Definition 6. *Let $\pi = (P_0, P_1)$ be a two-party protocol (i.e., a pair of Interactive Turing Machines). Then, the* local history *of P_i (for $i \in \{0,1\}$), during*

an execution of π *on input* (x_0, x_1) *and* internal *random tape* r^i*, is denoted by* $\mathsf{history}_{\pi,i}(x_0, x_1, n)$ *and equals* $(x_i, r^i, m_1^i, \ldots, m_t^i)$*, where* m_j^i *represents its* j*th message. The* history *of* π *during this execution is captured by* $(m_1^0, m_1^1), ..., (m_t^0, m_t^1)$ *and is denoted by* $\mathsf{history}_\pi$*. The configuration of* π *at some point during the interaction consists of the local configurations of* P_0, P_1*.*

Fail-stop games. We consider games of the form $\Gamma_{\pi,u} = (\{A_0, A_1\}, \{u_0, u_1\})$, where $A_0 = A_1 = \{\texttt{continue,abort}\}$. The decision is taken before the sending of each message. That is, first the program π_i is run from its current configuration, generating an outgoing message. Next, the party makes a strategic decision whether to continue or to abort. A `continue` action by player i means that the outgoing message generated by π_i is added to the history, and the new configuration is added to the local history. An `abort` action means that a special `abort` symbol is added to the configurations of both parties and then both π_0 and π_1 are run to completion, generating local outputs, and the game ends. We call such games *fail-stop games.*

The utility functions in fail-stop games may depend on all the histories: the joint one, as well as the local histories of both players. In the following sections, it will be convenient to define utility functions that consider a special field of the local history, called the *local output* of a player P_i. We denote this field by $\mathsf{output}_{\pi,i}$. Denote by $\sigma^{\texttt{continue}}$ the strategy that always returns `continue`. The basic Game Theoretic property of protocols that we will be investigating is whether the pair of strategies $(\sigma^{\texttt{continue}}, \sigma^{\texttt{continue}})$ is in a (computational) Nash equilibrium in fail-stop games, with respect to a given set of utilities and input distributions. That is:

Definition 7 (Nash protocols). *Let* $\mathcal{D}$ *be a set of distribution ensembles over pairs of strings, and let* $\mathcal{U}$ *be a set of extensive-form binary utility functions. A two-party protocol* π *is called* Nash Protocol with respect to $\mathcal{U}, \mathcal{D}$ *if, for any* $u \in \mathcal{U}$ *and* $D \in \mathcal{D}$*, the pair of strategies* $\sigma = (\sigma^{\texttt{continue}}, \sigma^{\texttt{continue}})$ *is in a computational Nash equilibrium for the fail-stop game* $\Gamma_{\pi,u}$ *and distribution ensemble* D*.*

On subgame perfect equilibria and related solution concepts. An attractive solution concept for extensive form games (namely, interactive protocols) is subgame perfect equilibria, which allow for analytical treatment which is not encumbered by "empty threats". Furthermore, some variants of this notion that are better suited to our computational setting have been recently proposed (see [21,31]). However, we note that in our limited case of fail-stop games any Nash equilibrium is subgame perfect. Indeed, once one of the parties aborts the computation there is no chance for the other party to "retaliate", hence empty threats are meaningless. (Recall that the output generation algorithms are not strategic, only the decision whether to abort is.)

3 Privacy and Correctness in Game Theoretic View

In this section we capture the traditional cryptographic privacy and correctness properties of protocols using Game Theoretic notions. We restrict attention to

the fail-stop setting and deterministic functions with a single output. (Fairness aside, private computation of functions with two distinct outputs can be reduced to this simpler case; see [12] for more details.)

Privacy in Game Theoretic view. Our starting point is the notion of private computation. A protocol is private if no (fail-stop) PPT adversary is able to distinguish any two executions where the adversary's inputs and outputs are the same, even when the honest party uses different inputs in the two executions. Our goal, then, is to define a set of utility functions that preserve this property for Nash protocols. We therefore restrict ourselves to input distributions over triples of inputs, where the input given to one of the parties is fixed, whereas the input of the other party is uniformly chosen from the remaining pair. This restriction captures the strength of cryptographic (semantic) security: *even* if a party knows that the input of the other party can only be one out of two possible values, the game does not give it the ability to tell which is the case. We then have a distribution for each such triple.

We turn to defining the utility functions. At first glance it may seem that one should define privacy by having each party *gain* whenever it learns something meaningful on the other party's private input. Nevertheless, it seems that it is better to make a party *lose* if the other party learns anything about its secret information. Intuitively, the reason is that it must be worthwhile for the party who holds the data to maintain it a secret. In other words, having the other party gain any profit when breaking secrecy is irrelevant, since it does not introduce any incentive for the former party to prevent this leakage. (Note however that here the utility of a party depends on events that are not visible to it during the execution.) The following definition formalizes the above.

Definition 8 (Distribution ensemble for privacy). *The* distribution ensemble for privacy for P_0 *for a two-party function f is the ensemble $\mathcal{D}^{\mathsf{p}}_f = \{D^{\mathsf{p}}_{f,n}\}_{n\in\mathbf{N}}$ where $D^{\mathsf{p}}_{f,n} = \{D_{a_0,a_1,b}\}_{a_0,a_1,b\in\{0,1\}^n, f(a_0,b)=f(a_1,b)}$, and $D_{a_0,a_1,b}$ outputs (x,b), where $x \stackrel{R}{\leftarrow} (a_0,a_1)$.*

Distribution ensembles for privacy for P_1 are defined analogously.

Let π be a two-party protocol computing a function f. Then, for every n, a, b, c as above and for every PPT algorithm $\mathcal{B}$, let the *augmented protocol for privacy* for π, with guess algorithm $\mathcal{B}$, be the protocol that first runs π, and then runs $\mathcal{B}$ on the local state of π and two additional auxiliary values. We assume that $\mathcal{B}$ outputs a binary value. This value is interpreted as a guess for which of the two auxiliary values is the input value of the other party.

Definition 9 (Utility function for privacy). *Let π be a two-party protocol and f be a two party function. Then, for every a_0, a_1, b such that $f(a_0,b) = f(a_1,b)$, and for every guessing algorithm $\mathcal{B}$, the* utility function for privacy *for party P_0, on input $x \in \{a_0,a_1\}$, is defined by:*

$$u^{\mathsf{p}}_0(\mathsf{history}_{\pi^{\mathsf{p}}_{\mathsf{Aug},\mathcal{B}},1}(x,b,n), a_0, a_1, b) \mapsto \begin{cases} -1 \text{ if } \mathsf{guess}_{\pi^{\mathsf{p}}_{\mathsf{Aug},\mathcal{B}},1} = g \text{ and } x = a_g \\ 0 \text{ otherwise} \end{cases}$$

The utility function for party P_1 is defined analogously. Note that if the history of the execution is empty, ie, no message has been exchanged between the parties, and the inputs of the parties are taken from a distribution ensemble for privacy, then u_0^{p} equals at least $-1/2$. This is due to the fact that P_1 can only guess x with probability at most $1/2$. Therefore, intuitively, it will be rational for P_0 to participate in the protocol (rather than to abort at the beginning) only if (and only if) the other party cannot guess the input of P_0 with probability significantly greater than $1/2$. The definition of Game-Theoretic privately is as follows:

Definition 10 (Game-Theoretic private protocols). *Let f and π be as above. Then, we say that π is* Game-Theoretic private for party P_0 *if $\pi^{\mathsf{p}}_{\mathsf{Aug},\mathcal{B}}$ is a Nash protocol with respect to $u_0^{\mathsf{p}}, u_1^{\mathsf{p}}$ and $\mathcal{D}_f^{\mathsf{p}}$ and all valid* PPT *$\mathcal{B}$.*

Game-Theoretic private protocol for P_1 is defined analogously. A protocol is Game-Theoretic private if it is Game-Theoretic private both for P_0 and for P_1.

Theorem 11. *Let f be a deterministic two-party function, and let π be a two-party protocol that computes f correctly (cf. Definition 2). Then, π is Game-Theoretic private if and only if π privately computes f in the presence of fail-stop adversaries.*

The proof can be found in the full version [2].

Correctness in Game Theoretic view. We continue with a formulation of a utility function that captures the notion of correctness as formalized in Definition 12. That is, we show that a protocol correctly computes a deterministic function if and only if the strategy that never aborts the protocol is in a computational Nash equilibrium with respect to the set of utilities specified as follows. The parties get high payoff only if they output the correct function value on the given inputs (types), or abort before the protocol starts; in addition, the players get no payoff for incorrect output. More formally, we introduce the set of distributions for which we will prove the Nash theorem. The distribution ensemble for correctness is simply the collection of all point distributions on pairs of inputs:

Definition 12 (Distribution ensemble for correctness). *Let f be a deterministic two-party function. Then, the* distribution ensemble for correctness *is the ensemble $\mathcal{D}_f^{\mathsf{c}} = \{D_n^{\mathsf{c}}\}_{n\in\mathbf{N}}$ where $D_n^{\mathsf{c}} = \{D_{a,b}\}_{a,b\in\{0,1\}^n}$, and $D_{a,b}$ outputs (a,b) w.p. 1.*

Note that a fail-stop adversary cannot affect the correctness of the protocol as it plays honestly with the exception that it may abort. Then, upon receiving an abort message we have the following: (i) either the honest party already learnt its output and so, correctness should be guaranteed, or, (ii) the honest party did not learn the output yet, for which it outputs $\perp$ together with its guess for the output (which corresponds to a legal output by Definition 2). Note that this guess is different than the guess appended in Definition 9 of utility definition for privacy, as here, we assume that the protocol instructs the honest party how to

behave in case of an abort. Furthermore, an incorrect protocol in the presence of fail-stop adversary implies that the protocol is incorrect regardless of the parties' actions (where the actions are continue or abort).

This suggests the following natural way of modeling a utility function for correctness: The parties gain a higher utility if they output the correct output, and lose if they output an incorrect output. Therefore, the `continue` strategy would not induce a Nash Equilibrium in case of an incorrect protocol, as the parties gain a higher utility by not participating in the execution. More formally:

Definition 13 (Utility function for correctness). *Let π be a two-party fail-stop game as above. Then, for every a, b as above the* utility function for correctness *for party P_0, denoted u_0^{c}, is defined by:*

- $u_0^{\mathsf{c}}(\mathsf{history}_{\pi,0}^{\phi}) = 1$.
- $u_0^{\mathsf{c}}(\mathsf{output}_{\pi,0}, a, b) \mapsto \begin{cases} 1 \text{ if } \mathsf{output}_{\pi,0} = f(a,b) \\ 0 \text{ otherwise} \end{cases}$

where $\mathsf{history}_{\pi,0}^{\phi}$ denotes the case that the local history of P_0 is empty. (Namely, P_0 does not participate in the protocol).

Intuitively, this implies that the protocol is a fail-stop Game if it is correct and vice versa. A formal statement follows below. u_1^{c} is defined analogously, with respect to P_1.

Theorem 14. *Let f be a deterministic two-party function, and let π a two-party protocol. Then, π is a Nash protocol with respect to $u_0^{\mathsf{c}}, u_1^{\mathsf{c}}$ and $\mathcal{D}_f^{\mathsf{c}}$ if and only if π correctly computes f in the presence of fail-stop adversaries.*

The proof can be found in the full version [2].

4 Exploring Fairness in the Two-Party Setting

Having established the notions of privacy and correctness using Game Theoretic formalism, our next goal is to capture fairness in this view. However, this turns out to be tricky, mainly due to the highly "reciprocal" and thus delicate nature of this notion. To illustrate, consider the simplistic definition for fairness that requires that one party learns its output if and only if the second party does. However, as natural as it seems, this definition is lacking since it captures each party's output as an atomic unit. As a result, it only considers the cases where the parties either learnt or did not learn their output *entirely*, and disregards the option in which partial information about the output may be gathered through the execution. So, instead, we would like to have a definition that calls a protocol fair if at any point in the execution both parties gather, essentially, the same partial information about their respective outputs.

Motivated by this discussion, we turn to the Game Theoretic setting with the aim to design a meaningful definition for fairness, as we did for privacy and correctness. This would, for instance, allow investigating known impossibility results under a new light. Our starting point is a definition that examines the

information the parties gain about their outputs during the game, where each party loses nominatively to the success probability of the other party guessing its output. (This is motivated by the same reasoning as in privacy). In order to obtain this, we first define a new set of utility functions for fairness for which we require that the game would be Nash; see Section 4.1 for the complete details.

Having defined fairness for rational parties, we wish to examine its strength against cryptographic attacks. We therefore introduce a new game-based definition that formalizes fairness for two-party protocols and is, in fact, equivalent to the Game Theoretic definition; see Theorem 21.

We then introduce in Section 4.3 a new notion of simulation based definition for capturing security of protocols that follow our game-based notion of fairness, specified above. This new notion is necessary as (gamed-based) fair protocols most likely cannot be simulatable according to the traditional simulation based definition [12]. We consider the notion of "partial information" in the ideal world alongside preserving some notion of privacy. We then prove that protocols that satisfy this new definition are also fair with respect to game-based definition.

Finally, we consider the realizability of our notion of fairness. We then observe that our notion is meaningful even in the case where parties are not guaranteed to always learn the output when both parties never abort. Somewhat surprisingly, in cases where the parties learn the output with probability one half or smaller, our notion of fairness is in fact *achievable* with no set-up or trusted third parties. We demonstrate two-party protocols that realize the new notion in this settings. We also show that whenever this probability raises above one half, our notion of fairness cannot be realized at all.

4.1 Fairness in Game Theoretic View

In this section we present our first definition for fairness that captures this notion from a Game Theoretic view. As for privacy and correctness, this involves definitions for utility functions, input distributions and a concrete fail-stop game (or the sequence of games). We begin with the description of the input distributions. As specified above, the input of each party is picked from a domain of size two, where all the outputs are made up of distinct outputs. More formally,

Definition 15 (Collection of distribution ensembles for fairness). *Let f be a two-party function. Let $(x_0^0, x_0^1, x_1^0, x_1^1, n)$ be an input tuple such that: $|x_0^0| = |x_0^1| = |x_1^0| = |x_1^1| = n$, and for every $b \in \{0,1\}$ it holds that:*

- $f_0(x_0^0, x_1^b) \neq f_0(x_0^1, x_1^b)$ *(in each run there are two possible outputs for P_0).*
- $f_1(x_0^b, x_1^0) \neq f_1(x_0^b, x_1^1)$*, (in each run there are two possible outputs for P_1).*

Then, a collection of distribution ensembles for fairness $\mathcal{D}_f^{\mathsf{f}}$ *is a collection of distributions $\mathcal{D}_f^{\mathsf{f}} = \{D_{x_0^0,x_0^1,x_1^0,x_1^1,n}\}_{x_0^0,x_0^1,x_1^0,x_1^1,n}$ such that for every $(x_0^0, x_0^1, x_1^0, x_1^1, n)$ as above, $D_{x_0^0,x_0^1,x_1^0,x_1^1,n}$ is defined by*

$$(x_0, x_1) \leftarrow D_{x_0^0,x_0^1,x_1^0,x_1^1,n}(1^n), \quad \text{where} \quad x_0 \stackrel{R}{\leftarrow} (x_0^0, x_0^1) \quad \text{and} \quad x_1 \stackrel{R}{\leftarrow} (x_1^0, x_1^1).$$

Next, let $\pi_{\mathcal{B}}$ be the protocol, where $\mathcal{B} = (\mathcal{B}_0, \mathcal{B}_1)$. By this notation, we artificially separate between the protocol and the predicting algorithms in case of prematurely abort. More precisely, in the case that P_0 prematurely aborts, P_1 invokes algorithm $\mathcal{B}_1$ on its input, its auxiliary information and the history of the execution, and outputs whatever $\mathcal{B}_1$ does. $\mathcal{B}_0$ is defined in a similar manner. In fact, we can refer to these two algorithms by the instructions of the parties regarding the values they need to output after each round, capturing the event of an early abort. We stress these algorithms are embedded within the protocol. However, this presentation enables us to capture scenarios where one of the parties follow the guessing algorithm as specified by the protocol, whereas, the other party follows an arbitrary algorithm. That is, we can consider protocols $\pi_{\mathcal{B}'}$ (with $\mathcal{B}' = (\widetilde{\mathcal{B}}_0, \mathcal{B}_1)$) that are equivalent to the original protocol $\pi_{\mathcal{B}}$ except for the fact that P_0 guesses its output according to $\widetilde{\mathcal{B}}_0$ instead of $\mathcal{B}_0$.

We describe the fairness game $\Gamma_{\pi_{\mathcal{B}},u^{\mathsf{f}}}$ for some $\mathcal{B} = (\mathcal{B}_0, \mathcal{B}_1)$. The inputs of the parties, x_0, x_1, are selected according to some distribution ensemble $D_{x_0^0,x_0^1,x_1^0,x_1^1}$ as defined in Definition 15. Then, the parties run the fail-stop game, where their strategies instruct them in each step whether to `abort` or `continue`. In case that a party P_i aborts, the outputs of both parties are determined by the algorithms $(\mathcal{B}_0, \mathcal{B}_1)$. Let $\mathsf{output}_{\pi_{\mathcal{B}'},i}$ denote the output of P_i in game $\pi_{\mathcal{B}'}$, then a utility function for fairness is defined by:

Definition 16 (Utility function for fairness). *Let f be a deterministic two-party function, and let π be a two-party protocol. Then, for every $x_0^0, x_0^1, x_1^0, x_1^1, n$ as above (cf. Definition 15), for every pair of strategies (σ_0, σ_1) and for every* PPT $\widetilde{\mathcal{B}}_0$*, the* utility function for fairness *for party P_0, denoted by u_0^{f}, is defined by:*

$$u_0^{\mathsf{f}}(\sigma_0,\sigma_1) \mapsto \begin{cases} 1 \text{ if } \mathsf{output}_{\pi^{\mathsf{f}}_{\mathcal{B}'},0} = f_0(x_0,x_1) \wedge \mathsf{output}_{\pi^{\mathsf{f}}_{\mathcal{B}'},1} \neq f_1(x_0,x_1) \\ -1 \text{ if } \mathsf{output}_{\pi^{\mathsf{f}}_{\mathcal{B}'},0} \neq f_0(x_0,x_1) \wedge \mathsf{output}_{\pi^{\mathsf{f}}_{\mathcal{B}'},1} = f_1(x_0,x_1) \\ 0 \text{ otherwise} \end{cases}$$

where x_0, x_1 are as in Definition 15 and $\mathcal{B}' = (\widetilde{\mathcal{B}}_0, \mathcal{B}_1)$. Moreover, the utility for P_1, $u_1^{\mathsf{f}} = 0$.

Since the utility function of P_1 is fixed, only P_1 has no incentive to change its strategy. Moreover, we consider here the sequence of games where P_1 always guesses its output according to $\mathcal{B}_1$, the "original" protocol. This actually means that P_1 always plays as honest, in the cryptographic point of view. We are now ready to define a protocol that is Game-Theoretic fair for P_1 as:

Definition 17 (Game-Theoretic fairness for P_1). *Let f and $\pi_{\mathcal{B}}$ be as above. Then, we say that $\pi_{\mathcal{B}}$ is* Game-Theoretic fair for party P_0 *if $\Gamma_{\pi_{\mathcal{B}'},(u_0^{\mathsf{f}},u_1^{\mathsf{f}})}$ is a Nash protocol with respect to $(u_0^{\mathsf{f}}, u_1^{\mathsf{f}})$ and $\mathcal{D}_f^{\mathsf{f}}$ and all* PPT $\widetilde{\mathcal{B}}_0$*, where $\mathcal{B}' = (\widetilde{\mathcal{B}}_0, \mathcal{B}_1)$.*

Similarly, we define Game-Theoretic fair for party P_0, where here we consider all the protocols $\pi_{\mathcal{B}'}$, for all PPT $\widetilde{\mathcal{B}}_1$ and $\mathcal{B}' = (\mathcal{B}_0, \widetilde{\mathcal{B}}_1)$, and the utilities functions are opposite (that is, the utility for P_0 is fixed into zero, whereas the utility of P_1 is modified according to its guess). We conclude with the definition for Game-Theoretic protocol:

Definition 18 (Game-Theoretic fair protocol). *Let f and π be as above. Them. we say that π is* Game-Theoretic fair protocol *if it is Game-Theoretic fair for both P_0 and P_1.*

4.2 A New Indistinguishability-Based Definition of Fairness

We now define a game-based definition (or, an indistinguishability definition) for fairness in cryptographic settings. Again, as in the game-theoretic settings, we assume that the protocol instructs the party what to output in case of abortion. Our definition tests the protocol in a "fail" environment, where each party has two possible inputs and its effective input is chosen uniformly at random from this set. Moreover, both parties know the input tuple and the distribution over the inputs. Before introducing the game-based definition, we first introduce non-trivial functionalities, to avoid functionalities that one of the parties may know the correct output without participating.

Definition 19 (Non-trivial functionalities.). *Let f be a two-party function. Then, f is non trivial if for all sufficiently large n's, there exists an input tuple $(x_0^0, x_0^1, x_1^0, x_1^1, n)$ such that $|x_0^0| = |x_0^1| = |x_1^0| = |x_1^1| = n$ and $\{f_0(x_0, x_1^b), f_0(x_0, x_1^b)\}_{b\in\{0,1\}}$, $\{f_1(x_0^b, x_1^0), f_1(x_0^b, x_1^1)\}_{b\in\{0,1\}}$ are distinct values.*
We are now ready to introduce our formal definition for fairness:

Definition 20 (Game-based definition for fairness.). *Let f be a non-trivial two-party function, and let π be a two-party protocol. Then, for every input tuple (cf. Definition 19) and any* PPT *fail-stop adversary $\mathcal{A}$, we define the following game:*
Game $\mathsf{Fair}_{\pi,\mathcal{A}}(x_0^0, x_0^1, x_1^0, x_1^1, n)$*:*

1. Two bits b_0, b_1 are picked at random.
2. Protocol π is run on inputs $x_0^{b_0}$ for P_0 and $x_1^{b_1}$ for P_1, where $\mathcal{A}$ sees the view of P_{i^*}.
3. Whenever $\mathcal{A}$ outputs a value y, P_{1-i^*} is given an `abort` message. (At this point, P_{1-i^*} would write its guess for $f_{1-i^*}(x_0^{b_0}, x_1^{b_1}, n)$ on its output tape.)
4. The output of the game is:
   - 1 if (i) $y = f_0(x_0^{b_0}, x_1^{b_1}, n)$ and (ii) P_{1-i^*} does not output $f_1(x_0^{b_0}, x_1^{b_1}, n)$.
   - -1 if (i) $y \neq f_0(x_0^{b_0}, x_1^{b_1}, n)$ and (ii) P_{1-i^*} outputs $f_1(x_0^{b_0}, x_1^{b_1}, n)$.
   - 0 in any other possibility (both parties output correct outputs, or both parties output incorrect outputs).

We say that π fairly computes *f if for every* PPT *adversary $\mathcal{A}$, there exists a negligible function $\mu(\cdot)$ such that for all sufficiently large inputs it holds that,*

$$\mathrm{E}(\mathsf{Fair}_{\pi,\mathcal{A}}(x_0^0, x_0^1, x_1^0, x_1^1, n)) \leq \mu(n)$$

At first sight it may seem that Definition 20 is tailored for the fair exchange function, i.e., when the parties trade their inputs. This is due to the fact that the parties' output completely reveal their inputs. Nevertheless, we note that the definition does not put any restriction on f in this sense, and is aimed to capture fairness with respect any nontrivial function. We continue with the following theorem:

Theorem 21. *Let f be a two-party function and let π be a protocol that computes f correctly. Then, π is Game-Theoretic fair (in the sense of Definition 18), if and only if π fairly computes f in the presence of fail-stop adversaries, (in the sense of Definition 20).*
The proof for this Theorem can be found in the full version [2].

4.3 A New Notion of Simulation Based Fairness

We formulate a new simulation-based notion of fair two party computation. The goal is to capture in a simulation-based way the same concept captured by the previous notions in this section. That is, we wish to allow the parties to obtain "partial information" on each other's secrets, as long as the gain of information is "essentially the same" for both parties.

The basic idea is to consider an ideal functionality (namely, a trusted party) which, in addition to obtaining from the parties their own inputs and a priori information on the input of the other party, also obtains from the ideal adversary (i.e., the simulator), a sampling PPT machine M. The functionality then runs M on the inputs of the parties and sends the parties the outputs that M returns. In order for our definition to make sense in the fair setting, we require that M should be "fair" in the sense that the values obtained by the parties are correlated with their respective outputs in essentially the same way.

For lack of space, we only sketch the highlights of this definition. See full details in [2]. We make the following requirements from the machine M:

1. CORRECTNESS: We require that $y'_0 = f_0(x_0, x)$, $y'_1 = f_1(x', x_1)$ for some x, x', where x_0, x_1 are the inputs of the parties that were sent to the trusted party and y'_0, y_1 are M's outputs. Namely, the sampling machine can never output a value for some party that is uncorrelated with its input.
2. FAIRNESS: we require that there exists a negligible function $\mu(\cdot)$ such that for all sufficiently large n's it holds that:

$$\left|\Pr\left[y_{i^*} = f_{i^*}(x_0, x_1)\right] - \frac{1}{2}\right| \leq \Pr\left[y_{1-i^*} = f_{1-i^*}(x_0, x_1)\right] - \frac{1}{2} + \mu(n) \quad (1)$$

where $(y_0, y_1) = M(x_0, x_1, z_0, z_1, r)$, the adversary controls party i^* and the probability is taken over the random coins of M.

The definition of security is now the standard one:

Definition 22. *Protocol π is said to* securely compute *f if for every* PPT *adversary $\mathcal{A}$ in the real model, there exists a* PPT *simulator $\mathcal{S}$ in the ideal model, such that:*

$$\{\text{NIDEAL}_{f,\mathcal{S}}(\boldsymbol{x})\}_{\boldsymbol{x}\in\{0,1\}^*} \equiv \{\text{REAL}_{\pi,\mathcal{A}}(\boldsymbol{x})\}_{\boldsymbol{x}\in\{0,1\}^*}$$

where the ideal execution uses any (adversatively chosen) sampling machine M that satisfies the above requirements.

Theorem 23. *Let f, π be as above. Then, if π is simulatable (in the sense of Definition 22), then π is fair with respect to the Game-Based (in the sense of Definition 20).*

4.4 The Feasibility of Our Definition

In this section, we study our new game-based cryptographic definition of fairness in a cryptographic context. Our starting point is any correct protocol, where both parties learn their output if playing honestly. We then show, that by relaxing the (negligibly close to) perfect completeness requirement, which implies that the parties should (almost) always learn their output if playing honestly, we can fully characterize the set of two-party protocols according partial correctness. Informally,

1. In case correctness holds with probability that is non-negligibly greater than 1/2, we present an impossibility result, saying that there does not exists a fair protocol with this probability of correctness. This implies that the difficulties in designing fair protocols are already embedded within the fail-stop setting. Stating differently, these difficulties already emerge whenever early abort is permitted.
2. On the positive side, in case correctness holds with probability that is smaller equals to 1/2, we show how to design a fair protocol that meets our notion of fairness. Specifically, we present a family of such protocols, parameterized by this probability of correctness. The implications of this is that there may be still hope for the fail-stop setting with respect to designing fair protocols.

An impossibility result. In this section we demonstrate that our game-based definition for fairness cannot be achieved for protocols that guarantee correctness with probability greater than 1/2. Before turning to our main theorem we present a definition of an α-correct protocol.

Definition 24. *Let f be a non-trivial two-party function, and let π be a two-party protocol. We say that the protocol π is a α-*correct for *f if there exists a negligible function $\mu(\cdot)$ such that for all sufficiently large x_0, x_1, n such that $|x_0| = |x_1| = n$,*

$$\left|\Pr[\mathsf{output}_{\pi,0}(x_0, x_1) = f_0(x_0, x_1) \wedge \mathsf{output}_{\pi,1}(x_0, x_1) = f_1(x_0, x_1, n)] - \alpha\right| \leq \mu(n)$$

where $\mathsf{output}_{\pi,i}(x_0, x_1)$ denote the output of party P_i when invoked on input x_i, while P_{1-i} is invoked on x_{1-i}, and both parties are honest.

Our theorem of impossibility:

Theorem 25. *Let f be a non-trivial two-party function. Then, for every non-negligible function $\epsilon(\cdot)$ and every $\alpha > 1/2 + \epsilon(n)$, there does not exist an α-*correct *protocol which is also fair (in the sense of Definition 20), with a polynomial round complexity.*

A positive result. Interestingly, we show that for relaxed correctness (i.e., lower equal than 1/2), there *do* exist non-trivial functionalities that can be computed fairly in this setting. In the following, we present a fair protocol in which either both parties learn the correct output together, or alternatively neither

party obtains a correct result. The case where in each execution exactly one party learns its correct output can also be achieved with fairness. More generally, denote by α the probability in which both parties should learn their outputs. Then, we show that for every $\alpha \leq 1/2$, there exists an α-correct protocol that is also fair, even in the non-simultaneous channel model. This relaxation is necessary to obtain fairness, as higher α values set a threshold for achieving this property (as shown in Section 4.4). Intuitively, the fact that each party does not know whether it has the correct output implies that a corrupted party would not have any incentive to abort after learning its output, since it does not give the honest party any new information anyway.

The protocol. The protocol is invoked over tuples of inputs with the distribution of choosing each input randomly out of a known pair. Let $x_0^0, x_0^1, x_1^0, x_1^1$ denote such an input tuple and denote by $x_0^{\mathsf{true}} \stackrel{\text{def}}{=} f_0(x_0^{b_0}, x_1^{b_1}), x_0^{\mathsf{false}} \stackrel{\text{def}}{=} f_0(x_0^{b_0}, x_1^{1-b_1})$, $x_1^{\mathsf{true}} \stackrel{\text{def}}{=} f_1(x_0^{b_0}, x_1^{b_1})$, and $x_1^{\mathsf{false}} \stackrel{\text{def}}{=} f_1(x_0^{1-b_0}, x_1^{b_1})$.

Then, function f^α; formally defined below, sets the output of the parties such that both learn the correct output with probability α, as required from an α-correct protocol. Moreover, the parties realize function f^α via protocol $\pi^\alpha_{\mathsf{abort}}$, which is secure-with-abort.

The Ideal Functionality f^α

- **Input:** P_0 inserts $b_0, x_0^0, x_0^1, x_1^0, x_1^1$. P_1 inserts $b_1, x_0^0, x_0^1, x_1^0, x_1^1$.
- **The function:**
  - Toss a coin σ that equals 0 with probability 2α, and equals 1 with probability $1 - 2\alpha$.
  - If $\sigma = 0$ (*parties learn same output*) do:
    - Toss another coin τ_0 uniformly at random from $\{0, 1\}$.
    - If $\tau_0 = 0$: set the output of P_0, P_1 to be $(x_0^{\mathsf{true}}, x_1^{\mathsf{true}})$, respectively.
    - If $\tau_0 = 1$: set the output of P_0, P_1 to be $(x_0^{\mathsf{false}}, x_1^{\mathsf{false}})$, respectively.
  - If $\sigma = 1$ (*parties learn true and false outputs*) do:
    - Toss another coin τ_1 uniformly at random from $\{0, 1\}$.
    - Set the output of P_{τ_1} to be $x_{\tau_1}^{\mathsf{true}}$.
    - Set the output of $P_{1-\tau_1}$ to be $x_{1-\tau_1}^{\mathsf{false}}$.

In the protocol, the parties compute the function f^α using security-with-abort. At the end of this computation, the adversary is the first to see the output. In case that the adversary deicides to abort, the honest party guesses its output at random from the two optional outputs. Intuitively, fairness is achieved since both parties learn the correct output with the same probability (1/2). On the other hand, in case where both parties play honestly, there is a correlation between the two outputs, as required from an α-correct protocol. For a full description of the protocol, together with a proof for the following theorem, see the full version [2].

Theorem 26. *Let f be a non-trivial two-party function. Then, for every $1/2 \geq \alpha \geq 0$, protocol π^α is an α-correct protocol in the f^α-hybrid model, and is simulatable (in the sense of Definition 22).*

References

1. Abraham, I., Dolev, D., Gonen, R., Halpern, J.Y.: Distributed Computing Meets Game Theory: Robust Mechanisms for Rational Secret Sharing and Multiparty Computation. In: PODC, pp. 53–62 (2006)
2. Asharov, G., Canetti, R., Hazay, C.: Towards a Game Theoretic View of Secure Computation (full version) (in ePrint)
3. Asharov, G., Lindell, Y.: Utility Dependence in Correct and Fair Rational Secret Sharing. Journal of Cryptology 24(1), 157–202 (2011)
4. Beaver, D., Goldwasser, S.: Multiparty computation with faulty majority. In: 30th FOCS, pp. 468–473 (1989)
5. Canetti, R.: Security and Composition of Multiparty Cryptographic Protocols. Journal of Cryptology 13(1), 143–202 (2000)
6. Cleve, R.: Limits on the Security of Coin Flips when Half the Processors are Faulty. In: 18th STOC, pp. 364–369 (1986)
7. Dodis, Y., Halevi, S., Rabin, T.: A Cryptographic Solution to a Game Theoretic Problem. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 112–130. Springer, Heidelberg (2000)
8. Dodis, Y., Rabin, T.: Cryptography and Game Theory. In: Algorithmic Game Theory. Cambridge University Press, Cambridge (2007)
9. Fuchsbauer, G., Katz, J., Naccache, D.: Efficient Rational Secret Sharing in Standard Communication Networks. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 419–436. Springer, Heidelberg (2010)
10. Fudenberg, D., Tirole, J.: Game Theory. The MIT Press, Cambridge (1991)
11. Garay, J.A., MacKenzie, P.D., Prabhakaran, M., Yang, K.: Resource fairness and composability of cryptographic protocols. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 404–428. Springer, Heidelberg (2006)
12. Goldreich, O.: Foundations of Cryptography. Basic Applications, vol. 2. Cambridge University Press, Cambridge (2004)
13. Goldreich, O., Micali, S., Wigderson, A.: How to Play any Mental Game – A Completeness Theorem for Protocols with Honest Majority. In: 19th STOC, pp. 218–229 (1987)
14. Goldreich, O., Kahan, A.: How To Construct Constant-Round Zero-Knowledge Proof Systems for NP. Journal of Cryptology 9(3), 167–190 (1996)
15. Goldwasser, S., Levin, L.A.: Fair computation of general functions in presence of immoral majority. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 77–93. Springer, Heidelberg (1991)
16. Goldwasser, S., Micali, S.: Probabilistic Encryption and How to Play Mental Poker Keeping Secret All Partial Information. J. Comput. Syst. Sci. 28(2), 270–299 (1984)
17. Goldwasser, S., Micali, S., Rachoff, C.: The Knowledge Complexity of Interactive Proof Systems. SIAM J. Computing 18(1), 186–208 (1989)
18. Gordon, S.D., Hazay, C., Katz, J., Lindell, Y.: Complete fairness in secure two-party computation. In: STOC, pp. 413–422 (2008)
19. Gordon, S.D., Katz, J.: Rational Secret Sharing, Revisited. In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 229–241. Springer, Heidelberg (2006)

20. Gordon, S.D., Katz, J.: Partial Fairness in Secure Two-Party Computation. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 157–176. Springer, Heidelberg (2010)
21. Gradwohl, R., Livne, N., Rosen, A.: Sequential Rationality in Cryptographic Protocols. In: FOCS, pp. 623–632 (2010)
22. Halpern, J., Teague, V.: Efficient Rational Secret Sharing in Standard Communication Networks. In: 36th STOC, pp. 623–632 (2004)
23. Halpern, J., Pass, R.: Game Theory with Costly Computation. In: ICS, pp. 120–142 (2010)
24. Izmalkov, S., Lepinski, M., Micali, S.: Verifiably Secure Devices. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 273–301. Springer, Heidelberg (2008)
25. Izmalkov, S., Micali, S., Lepinski, M.: Rational Secure Computation and Ideal Mechanism Design. In: 46th FOCS, pp. 585–595 (2005)
26. Katz, J.: Bridging Game Theory and Cryptography: Recent Results and Future Directions. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 251–272. Springer, Heidelberg (2008)
27. Kol, G., Naor, M.: Games for exchanging information. In: 40th STOC, pp. 423–432 (2008)
28. Kol, G., Naor, M.: Cryptography and Game Theory: Designing Protocols for Exchanging Information. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 320–339. Springer, Heidelberg (2008)
29. Lysyanskaya, A., Triandopoulos, N.: Rationality and Adversarial Behavior in Multi-party Computation. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 180–197. Springer, Heidelberg (2006)
30. Ong, S.J., Parkes, D.C., Rosen, A., Vadhan, S.P.: Fairness with an Honest Minority and a Rational Majority. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 36–53. Springer, Heidelberg (2009)
31. Pass, R., Shelat, A.: Renegotiation-Safe Protocols. In: Innovations in Computer Science, ICS 2011 (2011)