Rational Protocol Design: Cryptography Against Incentive-driven Adversaries

Juan Garay*, Jonathan Katz†, Ueli Maurer‡, Björn Tackmann‡, and Vassilis Zikas§
*AT&T Labs – Research
Email: garay@research.att.com
†University of Maryland
Email: jkatz@cs.umd.edu
‡ETH Zurich
Email: {maurer,bjoernt}@inf.ethz.ch
§UCLA
Email: vzikas@cs.ucla.edu

***Abstract*—Existing work on "rational cryptographic protocols" treats each party (or coalition of parties) running the protocol as a selfish agent trying to maximize its utility. In this work we propose a fundamentally different approach that is better suited to modeling a protocol under attack from an external entity. Specifically, we consider a two-party game between an *protocol designer* and an external *attacker*. The goal of the attacker is to break security properties such as correctness or privacy, possibly by corrupting protocol participants; the goal of the protocol designer is to prevent the attacker from succeeding.**

We lay the theoretical groundwork for a study of cryptographic protocol design in this setting by providing a methodology for defining the problem within the traditional simulation paradigm. Our framework provides ways of reasoning about important cryptographic concepts (e.g., adaptive corruptions or attacks on communication resources) not handled by previous game-theoretic treatments of cryptography. We also prove composition theorems that—for the first time—provide a sound way to design rational protocols assuming "ideal communication resources" (such as broadcast or authenticated channels) and then instantiate these resources using standard cryptographic tools.

Finally, we investigate the problem of secure function evaluation in our framework, where the attacker has to pay for each party it corrupts. Our results demonstrate how knowledge of the attacker's incentives can be used to circumvent known impossibility results in this setting.

***Keywords*-Cryptographic Protocols, Game Theory, Secure Computation, Composition**

Research supported in part by NSF awards #0830464, #1111599, and #1223623, the Army Research Laboratory under Cooperative Agreement Number W911NF-11-2-0086, the Swiss National Science Foundation (SNF), project no. 200020-132794, the NSF grants CCF-0916574; IIS-1065276; CCF-1016540; CNS-1118126; CNS-1136174; and Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014-11-1-0392. The views and conclusions contained in this document are those of the authors and should not be viewed as representing the official policies, either expressed or implies, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

I. Introduction

Consider a cryptographic protocol carrying out some task. In traditional security definitions, threats to the protocol are modeled by an explicit external entity—the *adversary*—who can corrupt some bounded number of protocol participants and make them behave in an arbitrary fashion. A protocol is "secure" if it realizes the ideal specification of the task against *any* adversarial strategy.

While this approach yields strong security guarantees, it has been criticized as being overly pessimistic since it neglects the *incentives* that lead parties to deviate from their prescribed behavior; this may result in protocols designed to defend against highly unlikely attacks. Motivated by this, a recent line of work on "rational cryptography" has focused on using ideas from game theory to analyze cryptographic protocols run by a set of rational parties [1]–[7].[1] There, parties are no longer viewed as being "good" (honest) or "bad" (corrupt); all parties are simply *rational*, motivated by some utility function. The goal of this line of work is to design protocols for which following the protocol is a game-theoretic equilibrium for the parties. It has been shown that by incorporating incentives one can circumvent impossibility results (e.g., for fairness in the two-party

[1]Our focus here is on applications of game theory to cryptography, where the protocol defines the game; another line of work initiated by Dodis, Rabin, and Halevi [8] focuses on applications of cryptography to game theory, with the aims to design cryptographic protocols for playing a pre-existing game [9]–[12].

setting [6], [7]), or design protocols with better efficiency (e.g., using the notion of covert security [13]).

Game-theoretic models of the above sort are useful for modeling incentive-driven misbehavior of mutually distrustful protocol participants, but they are not directly applicable to settings in which a set of mutually trusting parties willing to follow a protocol are concerned about an *external* attacker with its own set of preferences. In that context, it is more accurate to speak of a two-party game between the attacker and a defender (i.e., the protocol designer and the participants themselves). Note that, similarly to traditional cryptographic definitions, the notion of a rational external attacker is also suitable for capturing coalitions of incentive-driven misbehaving protocol participants.

In this work we propose a framework—which we term *rational protocol design*—that is intended to model exactly the setting just described. We prove a composition theorem for our framework which allows us to analyze protocols assuming idealized communication resources (e.g., secure point-to-point channels or a broadcast channel) are available, and then instantiate those idealized assumptions using standard cryptographic primitives. (A composition theorem of this sort is not known to hold in many existing rational-cryptography frameworks. See Section III for further discussion.) Finally, we showcase our framework by using it to model a scenario in which a protocol for multi-party computation is run in the presence of an external attacker who gains utility by violating privacy or correctness but must pay for corruption of protocol participants. In that setting we show how full security is attainable even with no *a priori* upper bound on the number of parties that can be compromised by the attacker. We explain our results in greater detail in the sections that follow.

A. Overview of our Framework

We provide a high-level overview of our framework, allowing ourselves to be somewhat informal. Formal details of the model are given in Section II.

At the most basic level, we propose a new way of modeling incentive-driven attacks via a two-party game between a *protocol designer* $\mathtt{D}$, who specifies a protocol Π for the (honest) participants to run, and a *protocol attacker* $\mathtt{A}$, who specifies a polynomial-time attack strategy $\mathcal{A}$ by which it may corrupt parties and try to subvert execution of the protocol (uncorrupted parties follow Π as prescribed). Both $\mathtt{D}$ and $\mathtt{A}$ are unbounded, and so this is a zero-sum extensive game with perfect information and observable actions, or more concretely a *Stackelberg game* (cf. [14, Section 6.2]).

The goal of the attacker is to choose a strategy that maximizes its utility; see below for how utility is defined. Since this is a zero-sum game, the goal of the defender is simply to minimize the attacker's utility.[2] We are interested in *ϵ-subgame-perfect equilibria* in this game, a refinement of subgame-perfect equilibria (the natural solution concept for Stackelberg games) that allows for negligible deviation in the choice of the players' strategies. We remark that although the notion of subgame-perfect equilibria is the most natural notion of stability for game-theoretic protocols, this notion is notoriously difficult to define in a computational setting [3], [4], [15]–[18]. We can use this notion cleanly here because the players in our game (namely, $\mathtt{D}$ and $\mathtt{A}$) are unbounded, and because our game tree has (fixed) depth two.

An additional novelty of our framework lies in the way we define the utility of the attacker. Fix some desired functionality $\mathcal{F}$ to be realized by a protocol. Given some moves $\Pi, \mathcal{A}$ by the players of our game, the utility of the attacker is defined using the traditional simulation paradigm in which a real-world execution of protocol Π in the presence of attack strategy $\mathcal{A}$ is compared to an ideal-world execution involving an ideal-world attack strategy (that is, a simulator) interacting with an ideal functionality for the task at hand. In our ideal world, however, we allow the simulator to make explicit queries to a "defective" ideal functionality $\langle\mathcal{F}\rangle$ that allow the simulator to cause specific "security breaches." The utility of the attacker depends on the queries made by the simulator,[3] and our initial game is specified by fixing a utility function that assigns values to events that occur in the ideal world. Roughly speaking, then, the goal of the attacker is to generate an adversarial strategy $\mathcal{A}$ that will "force" the simulator to cause certain security breaches in the ideal world in order to complete a successful simulation; Π is "secure" (with respect to some utility function) if the protocol can be simulated for any choice of $\mathcal{A}$ while generating utility 0 for the attacker.

Our choice to model utility by the occurrence of ideal-world events shares the same motivations that lead to adoption of the simulation paradigm in the standard cryptographic setting: it allows us, for example, to capture the fact that *some* information has been leaked without having to specify precisely what that information corresponds to, or having to worry about the fact that the information may correspond to different things

[2]There is some practical justification for assuming a zero-sum game. We leave consideration of nonzero-sum games for future work.

[3]Utility need not be positive; some events, such as corrupting parties, might incur negative utility for the attacker.

in different executions. We remark that the "security breaches" considered in this paper may be viewed as overly restrictive (making our definitions overly conservative), but our framework is flexible enough to allow one to specify more fine-grained security breaches and assign different utilities to each of them, if desired.

An important feature of our framework is that it also allows us to meaningfully compare two *insecure* protocols by comparing the maximum achievable utility of the attacker when attacking each of them. This, in turn, means we can speak of an "optimal" protocol (with respect to some utility function) even when a "secure" protocol is impossible. Coming back to our original two-party game, we show that a protocol Π is optimal if and only if having $\mathtt{D}$ choose that protocol yields an ϵ-subgame-perfect equilibrium in that game.

B. Results in our Framework

The fact that our framework can be cast as a cryptographic maximization problem enables us to prove a composition (subroutine replacement) theorem. Roughly speaking, this allows us to design and analyze protocols in a hybrid world where certain ideal functionalities are available (such as secure point-to-point channels or broadcast) and then draw conclusions about the resulting real-world protocol when those ideal functionalities are instantiated with secure implementations.

In Section IV we illustrate the benefits of our framework by investigating the problem of (multi-party) secure function evaluation (SFE) in the presence of an attacker whose goal is to violate the privacy of the uncorrupted parties' inputs, by learning more information on them than allowed by the inputs and outputs of corrupted parties, and/or correctness of their outputs. The attacker may specify an adversarial strategy in which arbitrarily many parties may be adaptively corrupted, though a cost is charged to the attacker for each corrupted party. We show the following results (here, "secure" is meant in the rational sense outlined above):

1. We show a secure protocol for computing arbitrary functions assuming the cost of corrupting parties is high compared to the attacker's utility for violating privacy. (This assumes static corruption only.) Conversely, we show that there are functions that cannot be computed securely when the cost of corruption is low compared to the attacker's utility for violating privacy or correctness. We also show a secure protocol (even under adaptive corruption) for computing arbitrary functions even when the utility for breaking privacy is high, but assuming that the utility for violating correctness is relatively low (both compared to the corruption cost).
2. Perhaps more interestingly, we provide a generic *two-party* SFE protocol, i.e., a protocol for evaluating any given function, when the utilities for both breaking privacy and/or correctness are higher than the cost of corruption (this again assumes static corruption) and prove that for a natural class of functions our protocol is in fact *optimal*, in the sense that it optimally tames the adversary. Note that our impossibility result excludes the existence of a *secure* protocol for this case.
3. Finally, for any function f in the class of functions for which $1/p$-secure protocols exist [19], [20] (for some polynomial p), we provide an attack-payoff secure protocol for evaluating f for *any* choice of the attacker's utility.

We remark that only the last result requires the protocol designer to have exact knowledge of the attacker's utilities; the rest only require known bounds on the attacker's utilities.

C. Comparison to Prior Work on Rational Cryptography

The main difference of our approach to rational cryptography compared to the traditional approach is that, instead of defining security in a game among rational protocol participants, we define it in a "meta"-game between the protocol designer—who decides the protocol to be executed by the (non-rational) honest parties—and the attacker. Our approach provides a simpler, more intuitive, and composable handling of incentive-driven attacks against cryptographic protocols. Furthermore, it allows to make optimality statements for cases for which security is impossible both in the traditional and in the rational setting. In the following, we give an abridged comparison of our results to existing results in rational cryptography. We refer to the full version [27] for a more detailed comparison.

To the best of our knowledge, our protocols are the *first* to consider incentives to deviate in SFE while relying on the minimal assumptions of insecure channels and a PKI. In particular, existing rational protocols assume ideal communication resources such as access to a broadcast channel or to secure channels. However, as we show, such implementations would fail if the communication resource were instantiated with a secure broadcast protocol. Similarly, the models from [9], [11], [12], [21]–[23] make even stronger communication assumptions which are *provably* unrealizable from standard cryptographic primitives [10], [23].

Recently, Halpern and Pass [5] suggested a game-theoretic framework which is suitable for modeling a protocol being run among a set of computationally bounded parties as a game. They investigated the relationship between equilibria in such games and traditional cryptographic notions of security. The motivation of our work is different, and in particular our aim is to analyze protocols that are not secure in the standard cryptographic sense but still offer protection against a rational attacker.

Aumann and Lindell [13] demonstrated how to take advantage of the adversary's "fear" of getting caught cheating to build more efficient protocols.[4] Their model can be readily captured in our framework by assigning a negative payoff to the event of (identifiable) abort. In work closer in spirit to ours, Groce *et al.* [24] investigated the feasibility of Byzantine agreement (BA) in a setting where the parties are rational but are split into two categories: the "selfish corrupt" parties that have some known utility function representing potential attacks to BA, and the "honest" parties who wish to follow the protocol. Our model can be tuned (by appropriately instantiating the utility function) to formalize the results of [24] in a simulation-based manner.

D. Preliminaries and Notation

Our model is based on the simulation paradigm and the formalization follows Canetti's UC framework [33]. We specify our (synchronous) protocols and ideal functionalities in a way similar to Canetti's synchronous model [25], knowing that (most) security proofs in that model carry over to the UC model, given that certain functionalities are available to the protocols [26]. Before proceeding to the details of our model and our results, we recall the basics of this model and specify some terminology and notation.

All entities involved in a protocol execution are described by *interactive Turing machines* (*ITMs*). The set of all *efficient*, i.e., composable polynomial time, ITMs is denoted by `ITM`. The *view* of an ITM in an execution is the list of all "external write" requests (that is, input- and output operations of the ITM) that involve the respective ITM—identified by the contents of its identity tape.

Most statements in this paper are actually asymptotic with respect to an (often implicit) security parameter $k \in \mathbb{N}$ and we use the standard definitions of *negligible* and *noticeable* (i.e, non-negligible). Furthermore, for functions f and g we introduce the symbols $f \overset{\text{negl}}{\approx} g$ to denote that for some negligible function $\mu, |f - g| \leq \mu$, and $f \overset{\text{negl}}{\geq} g$ to denote that $\exists$ negligible μ such that $f \geq g - \mu$ ("$\overset{\text{negl}}{\leq}$" is defined analogously). Unless stated otherwise, whenever we use *strict* inequalities we imply that the two sides differ by a noticeable (i.e., non-negligible) portion; that is, we write $a < b$ (resp., $a > b$) to denote that a is strictly smaller than $b - \mu$ (resp., a is strictly greater than $b + \mu$), for some noticeable function μ.

[4] Secure computation with honestly looking parties was also considered in [34].

II. Cryptographic Security as a Game

In this section, we introduce our definition of protocol optimality in terms of a game—which we term the *attack game*—between a protocol designer and an attacker, where the choice of the protocol designer is the protocol code (to be executed by uncorrupted parties) and the choice of the attacker is a concrete adversarial strategy for attacking the chosen protocol.

A. The Attack Game $\mathcal{G}_{\mathcal{M}}$

The attack game is a two-player zero-sum extensive game of perfect information with a horizon of length two (i.e., two sequential moves) and is formulated as a so-called *Stackelberg game* [14]. Informally, such a game involves two players, called the *leader* and the *follower*, and proceeds in two steps. In the first step, (only) the leader plays and the follower is (perfectly) informed about the leader's move; in the second step (only) the follower plays and the game terminates. (We refer to [14, Section 6.2] for a formal definition of extensive games of perfect information and a description of Stackelberg games.)

In our attack game, we refer to the leader and the follower as the *protocol designer* `D` and the *attacker* `A`, respectively. The game is parametrized by the (multi-party) functionality $\mathcal{F}$ to be computed—the number n of parties is implicit in any such description, which is known to both `D` and `A`. The designer `D` chooses a protocol for computing the functionality $\mathcal{F}$ from the set of all n-party (probabilistic and polynomial-time computable) protocols[5], the protocol consists of the code that the (honest) parties are supposed to execute. `D` sends to `A` the description $\Pi \subseteq \{0,1\}^*$ of this protocol in some predetermined encoding (in the form of interactive Turing machines (ITMs), for example). Upon receiving Π, it is `A`'s turn to play its strategy. `A` chooses a polynomial-time ITM $\mathcal{A}$ to attack protocol Π. The set Z of possible terminal histories is then the set

[5] As usual, to ensure that the running time of such a protocol is also polynomial in the respective security parameter, we assume that inputs to protocol machines include the security parameter in unary notation.

of sequences $(\Pi, \mathcal{A})$, where Π is an n-party protocol, and $\mathcal{A}$ is a central adversary (strategy) for attacking Π in the traditional cryptographic definition. We denote the corresponding game by $\mathcal{G}_{\mathcal{M}}$, where the subscript $\mathcal{M}$ in the above notation is referred to as the *attack model*, which specifies all the public parameters of the game, namely, the functionality and the description of the action sets as well as the utilities (formal definition in Section II-B below).

We use the standard solution concept for extensive games with perfect information, namely *subgame-perfect equilibrium*, where, informally, the actions of each party *at any point in the game* (i.e., after any history) form a best response to this history (cf. [14, Definition 97.2]). However, as we are interested in cryptographic security definitions with negligible error terms, we need a refinement of this notion which we call ϵ-*subgame perfect equilibrium*, and which considers as solutions all profiles in which the parties' utilities are ϵ-close to their best-response utilities. The formal definition of ϵ-subgame perfect equilibrium in our attack game follows trivially by adapting, as above, the definition from [14, Definition 97.2].

B. Defining the Utilities

In order to define the attacker's utility we proceed in three steps, as follows.

The first step is carried out by "relaxing" the ideal functionality $\mathcal{F}$ to obtain a (possibly) weaker ideal functionality $\langle \mathcal{F} \rangle$, which explicitly allows the attacks we wish to model. For example, $\langle \mathcal{F} \rangle$ could give the simulator access to the parties' inputs, or let it modify their outputs.

For the second step, we define a function v mapping the joint view of the relaxed functionality $\langle \mathcal{F} \rangle$ and the environment $\mathcal{Z}$ to a real-valued *payoff*. We then define the random variable ensemble $v_{\langle \mathcal{F} \rangle, \mathcal{S}, \mathcal{Z}}$ as the result of applying v to the views of $\langle \mathcal{F} \rangle$ and $\mathcal{Z}$ in a random experiment describing an ideal evaluation with ideal-world adversary $\mathcal{S}$. In other words, $v_{\langle \mathcal{F} \rangle, \mathcal{S}, \mathcal{Z}}$ describes (as a random variable) the payoff of $\mathcal{S}$ in an execution using directly the functionality $\langle \mathcal{F} \rangle$. The *(ideal) expected payoff* of $\mathcal{S}$ with respect to the environment $\mathcal{Z}$ is defined to be the expected value of $v_{\langle \mathcal{F} \rangle, \mathcal{S}, \mathcal{Z}}$, i.e.,

$$U_I^{\langle \mathcal{F} \rangle}(\mathcal{S}, \mathcal{Z}) = E(v_{\langle \mathcal{F} \rangle, \mathcal{S}, \mathcal{Z}}).$$

We refer to the triple $\mathcal{M} = (\mathcal{F}, \langle \mathcal{F} \rangle, v)$ as the *attack model*.

Finally (step 3), we define the payoff of an adversarial strategy for a certain protocol Π based on the above (ideal) expected payoff: The *(real) expected payoff* of a pair $(\mathcal{A}, \mathcal{Z})$ with respect to Π, where $\mathcal{Z}$ is the environment, $\mathcal{A}$ is the adversarial strategy,[6] and Π realizes $\langle \mathcal{F} \rangle$, is taken to be the payoff of the "best" simulator for $\mathcal{A}$, that is, the simulator that successfully emulates $\mathcal{A}$ while achieving the *minimum* score. The reason for considering a minimizing simulator that some events (for example, obtaining a party's input) can be provoked by the simulator without any effect in the remaining execution: the simulator can provoke an event even if it could simulate all necessary messages without doing so. The adversary, on the other hand, should be "rewarded" only if it *forces* the simulator $\mathcal{S}$ to provoke the event; hence, we minimize over the set of all "good" simulators.

Formally, for a functionality $\langle \mathcal{F} \rangle$ and a protocol Π, denote by $\mathcal{C}_{\mathcal{A}}$ the class of simulators that are "good" for $\mathcal{A}$, i.e, $\mathcal{C}_{\mathcal{A}} = \{\mathcal{S} \in \mathtt{ITM} \mid \forall \mathcal{Z} : \mathrm{EXEC}_{\Pi, \mathcal{A}, \mathcal{Z}} \approx \mathrm{EXEC}_{\langle \mathcal{F} \rangle, \mathcal{S}, \mathcal{Z}}\}$. The *real expected payoff* of the pair $(\mathcal{A}, \mathcal{Z})$ is then defined as

$$U^{\Pi, \langle \mathcal{F} \rangle}(\mathcal{A}, \mathcal{Z}) = \inf_{\mathcal{S} \in \mathcal{C}_{\mathcal{A}}} \{U_I^{\langle \mathcal{F} \rangle}(\mathcal{S}, \mathcal{Z})\}.$$

In other words, $U^{\Pi, \langle \mathcal{F} \rangle}$ assigns to each pair $(\mathcal{A}, \mathcal{Z}) \in \mathtt{ITM} \times \mathtt{ITM}$ (and each value of the security parameter k) a real number corresponding to the expected payoff obtained by $\mathcal{A}$ in attacking Π within environment $\mathcal{Z}$. For completeness, for adversaries $\mathcal{A}$ with $\mathcal{C}_{\mathcal{A}} = \emptyset$, i.e., adversaries that cannot be simulated in the $\langle \mathcal{F} \rangle$-ideal world, we define the score to be ∞, as these adversaries might break the security of the protocol in ways that are not even incorporated in the model.

Having defined the notion of real expected payoff, $U^{\Pi, \langle \mathcal{F} \rangle}(\mathcal{A}, \mathcal{Z})$, we proceed to define our main quantity of interest, namely, the (maximal) payoff of an adversarial strategy $\mathcal{A}$, which, intuitively, corresponds to its expected payoff when executing the protocol with $\mathcal{A}$'s "preferred" environment, i.e., the one that maximizes its score. More formally, the *(maximal) payoff* of an adversary $\mathcal{A}$ attacking the execution of protocol Π for realizing $\langle \mathcal{F} \rangle$ with respect to a certain payoff function v is defined as

$$\hat{U}^{\Pi, \langle \mathcal{F} \rangle}(\mathcal{A}) = \sup_{\mathcal{Z} \in \mathtt{ITM}} \{U^{\Pi, \langle \mathcal{F} \rangle}(\mathcal{A}, \mathcal{Z})\}.$$

Finally, having defined the (maximal) payoff of any given adversarial strategy $\mathcal{A}$, we can now define *the utility function $u_{\mathtt{A}}$ of attacker $\mathtt{A}$ in the attack game $\mathcal{G}_{\mathcal{M}}$* as follows. Let $(\Pi, \mathcal{A})$ be a terminal history in $\mathcal{G}_{\mathcal{M}}$, i.e., $\mathcal{A} = \mathtt{A}(\Pi)$. Then

$$u_{\mathtt{A}}(\Pi, \mathcal{A}) \coloneqq \hat{U}^{\Pi, \langle \mathcal{F} \rangle}(\mathcal{A}).$$

[6] We write $\mathcal{A}$ instead of $\mathtt{A}(\Pi)$ whenever the protocol Π is implicit by the context.

As $\mathcal{G}_{\mathcal{M}}$ is a zero-sum game, the designer's utility is defined as $u_{\mathtt{D}}(\cdot) \coloneqq -u_{\mathtt{A}}(\cdot)$.

We remark that we take the attacker's utility to be the maximal utility over the class of *all* environments, as (1) this is a natural worst-case assumption, and (2) it ensures that the achieved solution is stable even when the cheating parties have prior information about other parties' inputs.

A natural class of utility functions. As defined, the payoff function v can be arbitrary. In many applications, however, including those in Section IV, meaningful payoff functions have the following, simple representation: Let $(E_1, \ldots, E_n)$ denote a vector of (disjoint) events defined on the views (of $\mathcal{S}$ and $\mathcal{Z}$) in the ideal experiment corresponding to the security breaches that contribute to the attacker's utility. Each event E_i is assigned a real number γ_i, and the payoff function $v^{\vec{\gamma}}$ assigns, to each ideal execution, the sum of γ_i's for which E_i occurred. The ideal expected payoff of a simulator is computed according to our definition as

$$U_I^{\langle\mathcal{F}\rangle}(\mathcal{S}, \mathcal{Z}) = \sum_{E_i \in \vec{E}, \gamma_i \in \vec{\gamma}} \gamma_i \Pr[E_i],$$

where the probabilities are taken over the random coins of $\mathcal{S}$, $\mathcal{Z}$, and $\langle\mathcal{F}\rangle$. At times, to make the payoff vector $\vec{\gamma}$ explicit in the specification of the payoff function, we write $U_I^{\langle\mathcal{F}\rangle,\vec{\gamma}}(\mathcal{S}, \mathcal{Z})$.

III. The Attack Game as a Cryptographic (Maximization) Problem

We give a characterization of protocol optimality as a maximization problem using only cryptographic language, which emerges from our formulation of the problem as a zero-sum game. Furthermore, we provide a notion of security of protocols which is suitable for incentive-driven attackers. Let $\mathcal{M} = (\mathcal{F}, \langle\mathcal{F}\rangle, v)$ be an attack-model; for any given n-party protocol Π, we refer to the adversarial strategy $\mathcal{A}$ that achieves a (maximal) payoff $\hat{U}^{\Pi,\langle\mathcal{F}\rangle}(\mathcal{A})$ as a *$\mathcal{M}$-maximizing adversary for* Π (note that in $\mathcal{G}_{\mathcal{M}}$, $\mathcal{A}$ is a best response of $\mathtt{A}$ to protocol Π). Formally:

Definition 1. Let $\mathcal{M} = (\mathcal{F}, \langle\mathcal{F}\rangle, v)$ be an attack model and Π a protocol that realizes $\langle\mathcal{F}\rangle$. We say that an ITM $\mathcal{A}$ is a *$\mathcal{M}$-maximizing adversary for* Π if

$$\hat{U}^{\Pi,\langle\mathcal{F}\rangle}(\mathcal{A}) \overset{\text{negl}}{\approx} \sup_{\mathcal{A}' \in \mathtt{ITM}} \hat{U}^{\Pi,\langle\mathcal{F}\rangle}(\mathcal{A}') =: \hat{U}^{\Pi,\langle\mathcal{F}\rangle}.$$

The notion of maximizing adversaries naturally induces a notion of optimality for protocols:

Definition 2. Let $\mathcal{M} = (\mathcal{F}, \langle\mathcal{F}\rangle, v)$ be an attack model and Π be a protocol that realizes functionality $\langle\mathcal{F}\rangle$. We say that protocol Π is *attack-payoff optimal* in $\mathcal{M}$ if for any other protocol Π', $\hat{U}^{\Pi,\langle\mathcal{F}\rangle} \overset{\text{negl}}{\leq} \hat{U}^{\Pi',\langle\mathcal{F}\rangle}$.

We note that the above notion of optimality is only meaningful for comparing protocols that use the same "hybrid" functionalities (otherwise the trivial protocol using the functionality $\mathcal{F}$ would always be optimal, and the definition would coincide with Definition 4 below). We chose to nevertheless state the definition in the above simplified form and refer to [27] for further discussion.

The quantities $\hat{U}^{\Pi,\langle\mathcal{F}\rangle}$ and $\hat{U}^{\Pi',\langle\mathcal{F}\rangle}$ in Definition 2 denote the maximal payoffs of (different) adversarial strategies that attack protocols Π and Π', respectively. In the following, we prove an equivalence theorem linking the above notion of protocol optimality to the equilibrium in the attack game $\mathcal{G}_{\mathcal{M}}$. The equivalence is stated in the following theorem which follows from a simple backwards-induction argument:

Theorem 3. *Protocol Π is attack-payoff optimal in the attack model $\mathcal{M} = (\mathcal{F}, \langle\mathcal{F}\rangle, v)$ if and only if the strategy profile $(\Pi, \mathtt{A})$ is a λ-subgame-perfect equilibrium in the corresponding attack-game $\mathcal{G}_{\mathcal{M}}$ for some negligible λ, where for each $\Pi \in \mathtt{ITM}^n$, $\mathtt{A}(\Pi)$ is a $\mathcal{M}$-maximizing adversary attacking Π.*

At times protocols can "tame" the attacker completely, in the sense that the simulator is able to perform the simulation without ever provoking the events that give the chosen adversarial strategy a positive payoff. This is for example the case for protocols which securely realize the functionality $\mathcal{F}$ in the standard cryptographic sense. Still, depending on the attack model $\mathcal{M} = (\mathcal{F}, \langle\mathcal{F}\rangle, v)$—recall that this defines the attacker's incentive—a protocol tailored to a particular attacker's preference can achieve this "best possible" security even for functionalities that cannot be implemented in the traditional cryptographic sense. Intuitively, a protocol achieving this in some attack model $\mathcal{M}$ enjoys full security in the presence of a $\mathcal{M}$-maximizing adversary. We will call such protocols *attack-payoff secure*. In fact, most of our feasibility results are for attack-payoff security. Nonetheless, attack-payoff optimality is still a very interesting notion as it is achievable in settings where attack-payoff security cannot be obtained (see Section IV-B).

Definition 4. Protocol Π is *attack-payoff secure* in $\mathcal{M} = (\mathcal{F}, \langle\mathcal{F}\rangle, v)$ if $\hat{U}^{\Pi,\langle\mathcal{F}\rangle} \overset{\text{negl}}{\leq} \hat{U}^{\Phi^{\mathcal{F}},\langle\mathcal{F}\rangle}$, where $\Phi^{\mathcal{F}}$ is the "dummy" $\mathcal{F}$-hybrid protocol [33].

As protocol $\Phi^{\mathcal{F}}$ also implements $\langle\mathcal{F}\rangle$—in the $\mathcal{F}$-hybrid model—an attack-payoff secure protocol Π is at least as useful for the honest parties as the ideal

functionality $\mathcal{F}$.

Composition. Our model allows for protocol composition (in the sense of subroutine replacement). Interestingly, by using a backwards-induction argument (similar in spirit to the argument from [3]) we can show that support of such subroutine replacement has so far been lacking in existing rational cryptography models. The formal statement of the composition theorem can be found in [27].

Theorem 5 (Informal)**.** *Let Π be a $\mathcal{H}$-hybrid protocol, and Ψ be a protocol that securely realizes $\mathcal{H}$ (in the traditional simulation-based notion of security). Then for any $\mathcal{M} = (\mathcal{F}, \langle\mathcal{F}\rangle, v)$ replacing in Π calls to $\mathcal{H}$ by invocations of protocol Ψ does not (noticeably) increase the utility of a $\mathcal{M}$-maximizing adversary.*

The above theorem shows that the subroutine replacement operation can be applied to protocols that (fully, e.g., UC-)implement a given functionality without affecting optimality statements. If the underlying (subroutine) protocol Ψ is only attack-payoff secure/optimal for $\mathcal{H}$, we obtain only a weaker composition statement which, roughly speaking, establishes an upper bound on the utility loss of the designer when using attack-payoff secure/optimal protocols for replacing a subroutine. The formal statement and its proof can be found in [27].

IV. Scoring Privacy, Correctness, and the Cost of Corruption

In this section, we use our framework to study the feasibility of secure function evaluation (SFE) [28] with respect to a natural class of rational attackers. Recall that in SFE, a set of n distrustful parties with indices from the set $\mathcal{P} = [n]$ are to *correctly* compute a common function on their inputs, and in such a way that (to the extent possible) their individual inputs remain *private*. We consider an attacker whose specific incentive is to violate those very basic two properties. We additionally make the natural assumption that the act of corrupting parties is not for free (cf. [29]) and that there is a cost (negative payoff) associated with it, which further motivates the attacker to achieve its goals with as few corruptions as possible. Our protocols aim at realizing the fully secure version of SFE (i.e., including *robustness*) in which the parties always obtain their outputs.

Next, we specify how attacks with the above goals are modeled in our framework. Following the methodology described in Section II, we first describe a "relaxed" version of the ideal SFE functionality, denoted as $\langle\mathcal{F}_{\text{SFE}}\rangle$, to allow us to define events in the ideal experiment corresponding to the adversary achieving those goals. More precisely, in addition to $\mathcal{F}_{\text{SFE}}$'s standard communication, $\langle\mathcal{F}_{\text{SFE}}\rangle$ accepts the following commands from the simulator:

Breaking correctness: Upon receiving message $(\texttt{out}, y)$ from the simulator, replace the output of the function by y. We let E_{c} denote the event that the simulator sends to $\langle\mathcal{F}_{\text{SFE}}\rangle$ a message $(\texttt{out}, \cdot)$ and assign payoff γ_{c} to it.

Breaking privacy: Upon receiving message $(\texttt{inp}, \vec{x}_{\mathcal{I}})$, where $\vec{x}_{\mathcal{I}}$ is a vector containing inputs of corrupted parties (with adaptive corruptions, the set $\mathcal{I}$ of corrupted parties grows between queries), return to $\mathcal{S}$ the output y of the function evaluated on $(\vec{x}_{-\mathcal{I}}, \vec{x}_{\mathcal{I}})$, where $\vec{x}_{-\mathcal{I}}$ denotes the inputs given by honest parties (if these inputs are not received yet, $y \coloneqq \bot$). To capture a minimal privacy-breaking event (c.f. [27] for a discussion), the functionality restricts the class of queries it accepts from the simulator. In particular, for a query $(\texttt{inp}, \vec{x}_{\mathcal{I}})$, each $p_i \in \mathcal{I}$ must fulfill one of the following conditions: (1) this is the first time a query with $p_i \in \mathcal{I}$ is made, or (2) a query with input x_i' for p_i has been already made, and $x_i \in \{x_i', \bot\}$. Note, however, that $\langle\mathcal{F}_{\text{SFE}}\rangle$ does not register the vector $\vec{x}_{\mathcal{I}}$ as the actual inputs of corrupted parties; i.e., the command $(\texttt{inp}, \cdot)$ does not result in $\langle\mathcal{F}_{\text{SFE}}\rangle$ computing the honest parties' output. We let E_{p} denote the event that the simulator sends to $\langle\mathcal{F}_{\text{SFE}}\rangle$ a $(\texttt{inp}, \vec{x}_{\mathcal{I}})$ message and assign payoff γ_{p} to it.

Costly Corruption: To model costly corruption, we define for each set $\mathcal{I} \subseteq \mathcal{P}$ of (potentially) corrupted parties the event $E_{\mathcal{I}}$, which occurs when the adversary corrupts *exactly* the parties in $\mathcal{I}$ and assign payoff $\gamma_{\mathcal{I}}$ to it.

For the above defined events, the adversary's payoff is specified by the vector $\vec{\gamma} = (\gamma_{\mathsf{c}}, \gamma_{\mathsf{p}}, -\{\gamma_{\mathcal{I}}\}_{\mathcal{I}\subseteq\mathcal{P}})$. For convenience, we generally let the corruption costs $\gamma_{\mathcal{I}}$ be non-negative. We denote the corresponding payoff function as $v^{\vec{\gamma}}$. Following our methodology, the ideal expected utility for a simulator in the above described attack model $\mathcal{M} = (\mathcal{F}_{\text{SFE}}, \langle\mathcal{F}_{\text{SFE}}\rangle, v^{\vec{\gamma}})$ is defined as:

$$U_I^{\langle\mathcal{F}_{\text{SFE}}\rangle}(\mathcal{S}, \mathcal{Z}) \coloneqq \gamma_{\mathsf{c}} p_{\mathsf{c}} + \gamma_{\mathsf{p}} p_{\mathsf{p}} - \sum_{\mathcal{I}\subseteq\mathcal{P}} \gamma_{\mathcal{I}} p_{\mathcal{I}},$$

where $p_{\mathsf{c}} = \Pr[E_{\mathsf{c}}]$, $p_{\mathsf{p}} = \Pr[E_{\mathsf{p}}]$, and $p_{\mathcal{I}} = \Pr[E_{\mathcal{I}}]$ ($\mathcal{I} \subseteq \mathcal{P}$) are probabilities taken over the random coins of $\mathcal{S}$, $\mathcal{Z}$, and $\langle\mathcal{F}_{\text{SFE}}\rangle$.

In the remainder of this section, we study feasibility of SFE in the above attack model. To simplify our treatment, we restrict ourselves to the setting where the

same cost $\gamma_\$$ is associated with corrupting each party—i.e., for any set $\mathcal{I}$ of size t, $\gamma_\mathcal{I} = t\gamma_\$$. For simplicity and in slight abuse of notation, we shall denote the corresponding payoff vector as $\vec{\gamma} = (\gamma_\mathsf{c}, \gamma_\mathsf{p}, -\gamma_\$)$. Our protocols assume the availability of a public-key infrastructure.

A. Feasibility of Attack-Payoff SFE

We study attack-payoff SFE in the attack model described above, for various choices of the payoff vector $\vec{\gamma} = (\gamma_\mathsf{c}, \gamma_\mathsf{p}, -\gamma_\$)$. We start with a possibility result for *static corruptions*[7] and small payoff for breaking privacy, i.e., $\gamma_\mathsf{p} < \lceil \frac{n}{2} \rceil \gamma_\$$ (Theorem 6). Subsequently, we prove impossibility of attack-payoff security when, for some $t \geq \frac{n}{2}$, $\gamma_\mathsf{p} > t\gamma_\$$ and $\gamma_\mathsf{c} > (n-t)\gamma_\$$ (Theorem 7) and complete the picture with an attack-payoff secure protocol for a large class of $\vec{\gamma}$'s that are not excluded by the above impossibility, i.e., for $\gamma_\mathsf{p} + \gamma_\mathsf{c} < t\gamma_\$$ and $\gamma_\mathsf{c} < (n-t)\gamma_\$$ (Theorem 8); the latter protocol is secure even with *adaptive* corruptions (where no erasures are assumed). Due to lack of space, our feasibility statements are given in an existential form, i.e., we only claim existence of a protocol satisfying them; we remark however, that the corresponding proofs are constructive and refer to [27] for detailed descriptions of the corresponding protocols.

Theorem 6 (Informal)**.** *Let $\gamma_\mathsf{p} < \lceil \frac{n}{2} \rceil \gamma_\$$. Assuming the existence of enhanced trapdoor permutations, there exists a protocol $\Pi_{\text{St-SFE}}$ which is attack-payoff secure in the attack model $(\mathcal{F}_{\text{SFE}}, \langle \mathcal{F}_{\text{SFE}} \rangle, v^{\vec{\gamma}})$ with static but arbitrarily many corruptions.*

Proof (idea): The idea is to design a protocol $\Pi^f_{\text{St-SFE}}$ which is secure for an honest majority, and remains correct even if more than half of the parties are corrupted (i.e., the simulator might only need to provoke the event E_p in this case). Such a protocol can be obtained by appropriately modifying the protocol from [35] (we refer to [27] for a description). Given such a protocol $\Pi^f_{\text{St-SFE}}$ it is straightforward to verify that the adversary has no incentive to corrupt any party. Indeed, let t be the number of corrupted parties. If $t = 0$ then the adversary's utility equals 0. However, if $0 < t < n/2$ then the adversary's utility is $-t\gamma_\$ < 0$. Similarly, if $t > n/2$ then the adversary's utility is at most $\gamma_\mathsf{p} - t\gamma_\$ < \gamma_\mathsf{p} - \frac{n}{2}\gamma_\$ < 0$. ■

[7]Note that this implies that the corresponding class of adversaries that can be chosen by $\mathtt{A}$ in the attack-game $\mathcal{G}_\mathcal{M}$ is restricted to adversaries that statically choose the set of corrupted parties at the beginning of the protocol.

Theorem 7. *Let $t \geq n/2$. If $\gamma_\mathsf{p} > \gamma_\$ t$ and $\gamma_\mathsf{c} > \gamma_\$(n-t)$, then there exists a function f such that there exists no (polynomial-round) attack-payoff secure protocol Π in the attack model $(\mathcal{F}^f_{\text{SFE}}, \langle \mathcal{F}^f_{\text{SFE}} \rangle, v^{\vec{\gamma}})$.*

Proof (idea): In [30], it was shown that there is a finite, deterministic function f for which there is no polynomial-round protocol Π that simultaneously satisfies the following two properties: (1) Π securely realizes $\mathcal{F}^f_{\text{SFE}}$ in the presence of t corrupted parties, and (2) Π is $(n-t)$-private. By appropriately translating the privacy definition from [30] to our terminology, one can show that existence of a protocol which is attack-payoff secure in the attack model described in Theorem 7 contradicts the above impossibility result. ■

Theorem 8 (Informal)**.** *Let $t \geq n/2$ and let $\mathcal{M} = (\mathcal{F}_{\text{SFE}}, \langle \mathcal{F}_{\text{SFE}} \rangle, v^{\vec{\gamma}})$ be an attack model. Assuming enhanced trapdoor permutations, if $\gamma_\mathsf{c} + \gamma_\mathsf{p} < t\gamma_\$$ and $\gamma_\mathsf{c} < (n-t)\gamma_\$$ then there exists a protocol $\Pi_{\textit{Ad-SFE}}$ which is attack-payoff secure in $\mathcal{M}$ with arbitrarily many (adaptive) corruptions.*

The protocol construction uses an idea similar to the one Theorem 6. We refer to [27] for a detailed description and security proof.

B. A Positive Result for an Impossible Case

So far we have given positive and negative results for attack-payoff *secure* function evaluation. Here, we investigate attack-payoff *optimality* (cf. Definition 2) in settings where attack-payoff security is provably unachievable. More precisely, consider the case of secure two-party computation, where the values γ_p and γ_c are both greater than the cost $\gamma_\$$ of corrupting a party. Theorem 7 shows that in this setting it is impossible to get an attack-payoff secure protocol implementing functionality $\mathcal{F}_{\text{SFE}}$. We now describe a protocol that is attack-payoff optimal in $(\mathcal{F}_{\text{SFE}}, \langle \mathcal{F}_{\text{SFE}} \rangle, v^{\vec{\gamma}})$, assuming the adversary *statically* chooses the set of corrupted parties.

Our protocol, called $\Pi_{\mathtt{Op-SFE}}$, consists of two phases; let f denote the function to be computed:

1. $\Pi_{\mathtt{Op-SFE}}$ uses a protocol for SFE with abort (e.g., [31], [32]) to compute the following function f': f' takes as input the inputs of the parties to f and outputs an authenticated two-out-of-two sharing of the output of f along with an index $i \in_R \{1, 2\}$ chosen uniformly at random. In case of an unfair abort, the honest party takes a default value as the input of the corrupted party and locally computes the function f—and the protocol ends here.

2. If the evaluation of f' did not abort, the second phase consists of two rounds. In the first round, the output (sharing) is reconstructed towards p_i, and in the second round it is reconstructed towards $p_{\bar{i}}$.

Theorem 9. *Let $\mathcal{M} = (\mathcal{F}_{\text{SFE}}, \langle \mathcal{F}_{\text{SFE}} \rangle, v^{\vec{\gamma}})$ be an attack-model where $\min\{\gamma_{\mathsf{p}}, \gamma_{\mathsf{c}}\} > \gamma_{\$}$. Then for any static adversary attacking $\Pi_{\mathtt{Op-SFE}}$ in $\mathcal{M}$, $\hat{U}^{\Pi_{\mathtt{Op-SFE}}, \langle \mathcal{F}_{\text{SFE}} \rangle} \overset{\text{negl}}{\leq} \frac{\min\{\gamma_{\mathsf{p}}, \gamma_{\mathsf{c}}\}}{2} - \gamma_{\$}$.*

Proof (idea): If the adversary corrupts party p_i, which is the first to receive (information about) the output, then he can force the simulator to provoke one of the events E_{c} and E_{p}. However, the simulator can choose which of the events to provoke (and the best simulator will choose the one that pays less to the adversary) as at the point when the index i is announced, the adversary has only seen a share of the output (hence, the simulator can wait and query $\mathcal{F}_{\text{SFE}}$ at the very end). Because the index i of the party who first learns the output is chosen at random, the simulator needs to provoke one of these events with probability $1/2$; with the remaining $1/2$ probability the honest party receives the output first, in which case the adversary can do nothing. ■

To prove attack-payoff optimality of the above protocol, we show that there are functions f for which $\frac{\min\{\gamma_{\mathsf{p}}, \gamma_{\mathsf{c}}\}}{2} - \gamma_{\$}$ is a lower bound on the adversary's utility for any protocol in the attack model $(\mathcal{F}^f_{\text{SFE}}, \langle \mathcal{F}^f_{\text{SFE}} \rangle, v^{\vec{\gamma}})$. The formal statement and proof can be found in the full version [27].

C. Feasibility for any (Given) Payoff Vector

We finally consider the case where the payoffs for breaking privacy and/or correctness may be (upper-bounded by) arbitrarily large constants. As Theorem 7 suggests, for this case it is impossible to have an attack-payoff secure protocol for arbitrary functions. Nonetheless, we show possibility of such protocols for a large class of functions which, roughly speaking, correspond to the ones for which we can construct a $1/p$-secure (partially fair) protocol [19], [20]. The high-level idea is the following: $1/p$-secure protocols securely realize their specification, within an error smaller than the inverse of an arbitrary polynomial p (in the security parameter). Because $\vec{\gamma}$ is a vector of (constant) real values, we can choose the error $1/p$ to be small enough so that it is not in the adversary's interest to corrupt even a single party.

Theorem 10 (Informal)**.** *Let f be a (two-party or multi-party) function which can be evaluated by a $1/p$-secure protocol [19], [20] and $\mathcal{M} = (\mathcal{F}^f_{\text{SFE}}, \langle \mathcal{F}^f_{\text{SFE}} \rangle, v^{\vec{\gamma}})$ be an attack model where the elements γ_{p}, γ_{c}, and $\gamma_{\$}$ of $\vec{\gamma}$ are arbitrary (known) positive constants. Then, there exists an attack-payoff secure protocol in $\mathcal{M}$.*

REFERENCES

[1] J. Y. Halpern and V. Teague, "Rational secret sharing and multiparty computation: Extended abstract," in *STOC 2004*, L. Babai, Ed. Chicago, Illinois, USA: ACM Press, Jun. 13–16, 2004, pp. 623–632.

[2] I. Abraham, D. Dolev, R. Gonen, and J. Y. Halpern, "Distributed computing meets game theory: robust mechanisms for rational secret sharing and multiparty computation," in *PODC 2006*, E. Ruppert and D. Malkhi, Eds. Denver, Colorado, USA: ACM Press, Jul. 23–26, 2006, pp. 53–62.

[3] G. Kol and M. Naor, "Cryptography and game theory: Designing protocols for exchanging information," in *TCC 2008*, ser. LNCS, R. Canetti, Ed., vol. 4948. San Francisco, CA, USA: Springer, Berlin, Germany, Mar. 19–21, 2008, pp. 320–339.

[4] G. Fuchsbauer, J. Katz, and D. Naccache, "Efficient rational secret sharing in standard communication networks," in *TCC 2010*, ser. LNCS, D. Micciancio, Ed., vol. 5978. Zurich, Switzerland: Springer, Berlin, Germany, Feb. 9–11, 2010, pp. 419–436.

[5] J. Y. Halpern and R. Pass, "Game theory with costly computation: Formulation and application to protocol security," in *ICS 2010*, A. C.-C. Yao, Ed. Tsinghua University Press, 2010, pp. 120–142.

[6] G. Asharov, R. Canetti, and C. Hazay, "Towards a game-theoretic view of secure computation," in *EUROCRYPT 2011*, ser. LNCS, K. G. Paterson, Ed., vol. 6632. Springer, 2011, pp. 426–445.

[7] A. Groce and J. Katz, "Fair computation with rational players," in *EUROCRYPT 2012*, ser. LNCS, D. Pointcheval and T. Johansson, Eds., vol. 7237. Springer, 2012, pp. 81–98.

[8] Y. Dodis, S. Halevi, and T. Rabin, "A cryptographic solution to a game theoretic problem," in *CRYPTO 2000*, ser. LNCS, M. Bellare, Ed., vol. 1880. Santa Barbara, CA, USA: Springer, Berlin, Germany, Aug. 20–24, 2000, pp. 112–130.

[9] M. Lepinski, S. Micali, C. Peikert, and A. Shelat, "Completely fair SFE and coalition-safe cheap talk," in *PODC 2004*, S. Chaudhuri and S. Kutten, Eds. St. John's, Newfoundland, Canada: ACM Press, Jul. 25–28, 2004, pp. 1–10.

[10] M. Lepinski, S. Micali, and A. Shelat, "Collusion-free protocols," in *STOC 2005*, H. N. Gabow and R. Fagin, Eds. Baltimore, Maryland, USA: ACM Press, May 22–24, 2005, pp. 543–552.

[11] S. Izmalkov, S. Micali, and M. Lepinski, "Rational secure computation and ideal mechanism design," in *FOCS 2005*. Pittsburgh, PA, USA: IEEE Computer Society Press, Oct. 23–25, 2005, pp. 585–595.

[12] S. Izmalkov, M. Lepinski, and S. Micali, "Verifiably secure devices," in *TCC 2008*, ser. LNCS, R. Canetti, Ed., vol. 4948. San Francisco, CA, USA: Springer, Berlin, Germany, Mar. 19–21, 2008, pp. 273–301.

[13] Y. Aumann and Y. Lindell, "Security against covert adversaries: Efficient protocols for realistic adversaries," in *TCC 2007*, ser. LNCS, S. Vadhan, Ed., vol. 4392. Springer, 2007, pp. 137–156.

[14] M. Osborne and A. Rubinstein, *A Course in Game Theory*. MIT Press, 1994.

[15] S. D. Gordon and J. Katz, "Rational secret sharing, revisited," in *SCN 2006*, ser. LNCS, R. D. Prisco and M. Yung, Eds., vol. 4116. Maiori, Italy: Springer, Berlin, Germany, Sep. 6–8, 2006, pp. 229–241.

[16] G. Kol and M. Naor, "Games for exchanging information," in *STOC 2008*, C. Dwork, Ed. Victoria, British Columbia, Canada: ACM Press, May 17–20, 2008, pp. 423–432.

[17] J. Y. Halpern, "Beyond Nash equilibrium: solution concepts for the 21st century," in *PODC 2008*, R. A. Bazzi and B. Patt-Shamir, Eds. Toronto, Ontario, Canada: ACM Press, Aug. 18–21, 2008, pp. 1–10.

[18] R. Gradwohl, N. Livne, and A. Rosen, "Sequential rationality in cryptographic protocols," in *FOCS 2010*. IEEE Computer Society Press, 2010, pp. 623–632.

[19] D. Gordon and J. Katz, "Partial fairness in secure two-party computation," in *EUROCRYPT 2010*, ser. LNCS, H. Gilbert, Ed., vol. 6110. Springer, 2010, pp. 157–176.

[20] A. Beimel, Y. Lindell, E. Omri, and I. Orlov, "$1/p$-secure multiparty computation without honest majority and the best of both worlds," in *CRYPTO 2011*, ser. LNCS, P. Rogaway, Ed., vol. 6841. Springer, 2011, pp. 277–296.

[21] J. Alwen, G. Persiano, and I. Visconti, "Impossibility and feasibility results for zero knowledge with public keys," in *CRYPTO 2005*, LNCS, V. Shoup, Ed., vol. 3621. Santa Barbara, CA, USA: Springer, Berlin, Germany, Aug. 14–18, 2005, pp. 135–151.

[22] J. Alwen, J. Katz, Y. Lindell, G. Persiano, A. Shelat, and I. Visconti, "Collusion-free multiparty computation in the mediated model," in *CRYPTO 2009*, ser. LNCS, S. Halevi, Ed., vol. 5677. Santa Barbara, CA, USA: Springer, Berlin, Germany, Aug. 16–20, 2009, pp. 524–540.

[23] J. Alwen, J. Katz, U. Maurer, and V. Zikas, "Collusion-preserving computation," in *CRYPTO 2012*, ser. LNCS, R. Canetti and J. Garay, Eds. Santa Barbara, CA, USA: Springer, Berlin, Germany, Aug. 2012, pp. 124–143.

[24] A. Groce, J. Katz, A. Thiruvengadam, and V. Zikas, "Byzantine agreement with a rational adversary," in *ICALP 2012*, ser. LNCS. Springer, Berlin, Germany, 2012, pp. 561–572.

[25] R. Canetti, "Security and composition of multiparty cryptographic protocols," *Journal of Cryptology*, vol. 13, pp. 143–202, April 2000.

[26] J. Katz, U. Maurer, B. Tackmann, and V. Zikas, "Universally composable synchronous computation," in *TCC 2013*, ser. LNCS, A. Sahai, Ed., vol. 7785. Springer, March 2013, pp. 477–498.

[27] J. A. Garay, J. Katz, U. Maurer, B. Tackmann, and V. Zikas, "Rational protocol design: Cryptography against incentive-driven adversaries," Cryptology ePrint Archive, Report 2013/496, August 2013, (Full version).

[28] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game, or a completeness theorem for protocols with honest majority," in *STOC '87*, A. Aho, Ed. New York City,, New York, USA: ACM Press, May 25–27, 1987, pp. 218–229.

[29] J. A. Garay, D. Johnson, A. Kiayias, and M. Yung, "Resource-based corruptions and the combinatorics of hidden diversity," in *ITCS 2013*, R. D. Kleinberg, Ed. ACM, 2013, pp. 415–428.

[30] J. Katz, "On achieving the 'best of both worlds' in secure multiparty computation," in *STOC 2007*, U. Feige, Ed. San Diego, California, USA: ACM Press, June 11–13, 2007, pp. 11–20.

[31] R. Canetti, U. Feige, O. Goldreich, and M. Naor, "Adaptively secure multi-party computation," in *STOC '96*, G. L. Miller, Ed. Philadephia, Pennsylvania, USA: ACM Press, May 22–24, 1996, pp. 639–648.

[32] R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai, "Universally composable two-party and multi-party secure computation," in *STOC 2002*, J. H. Reif, Ed. Montréal, Québec, Canada: ACM Press, May 19–21, 2002, pp. 494–503.

[33] R. Canetti, "Universally composable security: A new paradigm for cryptographic protocols," in *FOCS 2001*. Las Vegas, Nevada, USA: IEEE Computer Society Press, Oct. 14–17, 2001, pp. 136–145. Updated full version available at http://eprint.iacr.org/2000/067.

[34] R. Canetti and R. Ostrovsky, "Secure Computation with honest-looking parties: What if nobody is truly honest?" in *STOC '99*, J. S. Vitter, L. L. Larmore, F. Thomson Leighton, Eds. Atlanta, Georgia, USA: ACM Press, May 1–4, 1999, pp. 255–264.

[35] Y. Ishai, E. Kushilevitz, Y. Lindell, and E. Petrank. "On combining privacy with guaranteed output delivery in secure multiparty computation," in *CRYPTO 2006*, ser. LNCS, Cynthia Dwork, Ed., vol. 4117. Santa Barbara, CA, USA: Springer, Berlin, Germany, Aug. 20–24, 2006, pp. 483–500.