Rational Protocol Design: Cryptography Against Incentive-driven Adversaries

Juan Garay
AT&T Labs – Research
garay@research.att.com

Jonathan Katz*
University of Maryland
jkatz@cs.umd.edu

Ueli Maurer†
ETH Zurich
maurer@inf.ethz.ch

Björn Tackmann†
ETH Zurich
bjoernt@inf.ethz.ch

Vassilis Zikas‡
UCLA
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" (e.g., 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.

1 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

*Research supported in part by NSF awards #0830464, #1111599, and #1223623, and the Army Research Laboratory under Cooperative Agreement Number W911NF-11-2-0086. 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.

†Research supported in part by the Swiss National Science Foundation (SNF), project no. 200020-132794.

‡Research supported in part by 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.

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 [HT04, ADGH06, KN08a, FKN10, HP10, ACH11, GK12].[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 setting [ACH11, GK12]), or design protocols with better efficiency (e.g., using the notion of covert security [AL07]).

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, similar to traditional cryptographic definitions, the notion of a rational external attacker is also suitable for capturing coalitions of rationally cheating 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 4 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.

1.1 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 2.

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

[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 [DHR00] focuses on applications of cryptography to game theory, with the aims to design cryptographic protocols for playing a pre-existing game [LMPS04, LMS05, IML05, ILM08]. We refer to Appendix A for a short survey of the literature on rational cryptography.

prescribed). Both D and 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. [OR94, 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 [GK06, KN08b, KN08a, Hal08, FKN10, GLR10]. We can use this notion cleanly here because the players in our game (namely, D and 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 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 D choose that protocol yields an ϵ-subgame-perfect equilibrium in that game.

1.2 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

[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.

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 5 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 under the assumption that 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 [GK10, BLOO11] (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.

1.3 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 to 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. For completeness we include a short survey on these results and a more detailed comparison in Appendix A.

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. However, as demonstrated in Section 4, such implementations would fail if the ideal broadcast channel were instantiated with a secure broadcast protocol.[4] Similarly, the models from [LMPS04, IML05, ILM08, ASV08, AKL+09, AKMZ12, CV12] require even stronger communication resources which are unrealizable from standard cryptographic primitives [LMS05, AKMZ12].

Recently, Halpern and Pass [HP10] 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 [AL07] demonstrated how to take advantage of the adversary's "fear" of getting caught cheating to build more efficient protocols. 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.* [GKTZ12] 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 [GKTZ12] in a simulation-based manner.

1.4 Preliminaries and Notation

Our model is based on the simulation paradigm and the formalization follows Canetti's UC framework [Can05]. We specify our (synchronous) protocols and ideal functionalities in a way similar to Canetti's synchronous model [Can00], knowing that (most) security proofs in that model carry over to the UC model, given that certain functionalities are available to the protocols [KMTZ13]. Before proceeding to the details of our model and our results, we recall the basics of this model and specify some terminology and notation. Readers familiar with the [Can00, Can05] models can move forward to the notational conventions at the end of this section and refer to it whenever necessary.

At a high level, simulation-based definitions use the real-world/ideal-world paradigm: In the real world, the parties execute the protocol using channels and other functionalities defined by the model, while in the ideal world, the parties securely provide their input to an ideal functionality $\mathcal{F}$ that executes the task to be achieved by the protocol and returns the resulting outputs to the parties. The protocol *securely realizes* the functionality $\mathcal{F}$ if, for any real-world adversary $\mathcal{A}$ attacking the protocol execution, there is an ideal-world adversary $\mathcal{S}$, also called the *simulator*, that emulates $\mathcal{A}$'s attack. The simulation is successful if no distinguisher $\mathcal{Z}$—often called the *environment*—which interacts, in a well-defined manner, with the parties and the adversary/simulator, can distinguish between the two worlds.

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`. We generally denote our protocols by Π and our (ideal) functionalities by $\mathcal{F}$, the adversary by $\mathcal{A}$, the simulator by $\mathcal{S}$, and the environment by $\mathcal{Z}$. The term $\{\text{EXEC}_{\Pi,\mathcal{F},\mathcal{A},\mathcal{Z}}(k,z)\}_{k\in\mathbb{N},z\in\{0,1\}^*}$ denotes the random variable ensemble describing the contents of $\mathcal{Z}$'s output tape after an execution with Π, $\mathcal{F}$,

[4] Other rational works use simultaneous broadcast [HT04, GK06], which is hard to implement even in a traditional (non-rational) cryptographic sense.

and $\mathcal{A}$, on auxiliary input $z \in \{0,1\}^*$. We often drop the parameters k and z and simply refer to the ensemble by $\text{EXEC}_{\Pi,\mathcal{F},\mathcal{A},\mathcal{Z}}$ if the meaning is clear from the context. The *view* of an ITI in an execution is the list of all "external write" requests (that is, input- and output operations of the ITI) that involve the respective ITI—identified by the contents of its identity tape.

In the model of [Can05], a protocol is described by an arbitrary ITM that can in particular instantiate ideal functionalities, cf. [Can05, Sections 3.2 and 4.5]. Consequently, an $\mathcal{F}$-hybrid protocol (for some functionality $\mathcal{F}$) is merely a special type of protocol, and by the way the invocation is defined in [Can05] it is generally even hard to decide for some given protocol whether or not it is an $\mathcal{F}$-hybrid protocol for some functionality $\mathcal{F}$. As this property impedes statements about the nonexistence of protocols, we restrict ourselves to considering protocols of a specific type: we require that each protocol *explicitly* specifies its hybrids (e.g., in a header of some specified format), and does not instantiate any ITMs beyond those specified in the header.

We use the following notational conventions throughout the paper. As usually, $[n]$ denotes the set $\{1,\ldots,n\}$, and $\mathbb{F}$ denotes an arbitrary (but fixed) finite field. Most statements in this paper are asymptotic with respect to an (often implicit) security parameter $k \in \mathbb{N}$. Hence, $f \leq g$ means that $\exists k_0\ \forall k \geq k_0, f(k) \leq g(k)$, and a function $\mu, \mathbb{N} \to \mathbb{R}$ is *negligible* if for all polynomials p, $\mu \leq 1/p$, and *noticeable* if there exists a polynomial p with $\mu \geq 1/p$. Furthermore, we introduce the symbols $f \overset{\text{negl}}{\approx} g$ to denote that $\exists$ negligible μ such that $|f-g| \leq \mu$, and $f \overset{\text{negl}}{\geq} g$ to denote that $\exists$ negligible $\mu :\ f \geq g - \mu$ (analogously with "$\overset{\text{negl}}{\leq}$"). 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.

2 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.

2.1 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* [OR94]. Informally, such a game involves two players, called the *leader* and the *follower*, and proceeds in two steps. In the first step, the leader plays and the follower is (perfectly) informed about the leader's move; in the second step the follower plays and the game terminates. (We refer to [OR94, 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 parameterized 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. $\mathtt{D}$ sends to $\mathtt{A}$ 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 $\mathtt{A}$'s turn to play its strategy. $\mathtt{A}$ chooses a polynomial-time ITM $\mathcal{A}$ to attack protocol Π. 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 (see Section 2.2).

The standard (and most natural) solution concept for Stackelberg games (and, more generally, extensive games with perfect information) is *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. [OR94, 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.

We will denote a strategy profile of an attack game $\mathcal{G}_{\mathcal{M}}$ as a vector $(\Pi, \mathtt{A})$. Note that, formally, $\mathtt{A}$ is a function mapping efficient protocols to corresponding adversarial strategies. We use (real-valued functions) $u_{\mathtt{D}}(\cdot)$ and $u_{\mathtt{A}}(\cdot)$ to denote the utilities of $\mathtt{D}$ and $\mathtt{A}$, respectively (formal definitions of these also in the next section). We are now ready to state the solution concept for our game.

Definition 1. Let $\mathcal{G}_{\mathcal{M}}$ be an attack game. A strategy profile $(\Pi, \mathtt{A})$ is an *ϵ-subgame perfect equilibrium in $\mathcal{G}_{\mathcal{M}}$* if the following conditions hold: (1) for any $\Pi' \in \mathtt{ITM}^n$, $u_{\mathtt{D}}(\Pi', \mathtt{A}(\Pi')) \leq u_{\mathtt{D}}(\Pi, \mathtt{A}(\Pi)) + \epsilon$, and (2) for any $\mathtt{A}' \in \mathtt{ITM}$, $u_{\mathtt{A}}(\Pi, \mathtt{A}'(\Pi)) \leq u_{\mathtt{A}}(\Pi, \mathtt{A}(\Pi)) + \epsilon$.

2.2 Defining the Utilities

As sketched in Section 1.1, the methodology for defining the attacker's utility with respect to a given functionality $\mathcal{F}$ consists of three steps: (1) we "relax" the functionality $\mathcal{F}$ to a weaker version, denoted as $\langle \mathcal{F} \rangle$, which allows for the idealized counterparts of the "security breaches" the attacker wants to provoke; (2) we define the payoff/score of any *ideal-world* adversary as a function v of the view of and ideal evaluation of the relaxed functionality; and (3) we assign to each adversarial strategy for a given protocol the expected payoff/score achieved by the *best* simulator (or ∞ if no such simulator exists). We refer to the triple $\mathcal{M} = (\mathcal{F}, \langle \mathcal{F} \rangle, v)$ as the *attack model*. We next elaborate on the three steps.

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 real-valued random variable ensemble $\{v_{\langle \mathcal{F} \rangle, \mathcal{S}, \mathcal{Z}}(k, z)\}_{k \in \mathbb{N}, z \in \{0,1\}^*}$ ($v_{\langle \mathcal{F} \rangle, \mathcal{S}, \mathcal{Z}}$ for short) 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

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

to be the expected value of $v_{\langle\mathcal{F}\rangle,\mathcal{S},\mathcal{Z}}$, i.e.,

$$U_I^{\langle\mathcal{F}\rangle}(\mathcal{S},\mathcal{Z}) = E(v_{\langle\mathcal{F}\rangle,\mathcal{S},\mathcal{Z}}).^6$$

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,[7] 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. We remark that a non-uniform simulator can minimize the score for every value of the security parameter.

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, $\mathcal{C}_\mathcal{A} = \{\mathcal{S} \in \mathtt{ITM} \mid \forall\mathcal{Z} : \text{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}} \approx \text{EXEC}_{\langle\mathcal{F}\rangle,\mathcal{S},\mathcal{Z}}\}$. 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 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. This is just for completeness of the definition, i.e., for incorporating arbitrary adversarial strategies, and will not be used in this work, as all the considered protocols are required to be secure with respect to the relaxed functionality $\langle\mathcal{F}\rangle$.

Now, having defined the notion of real expected payoff, $U^{\Pi,\langle\mathcal{F}\rangle}(\mathcal{A},\mathcal{Z})$, we can 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 the adversary'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

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

[6] Using expectation to define $U_I^{\langle\mathcal{F}\rangle}(\mathcal{S},\mathcal{Z})$ keeps the description simple and close to the notion of expected utility from classical game theory. In fact, $U_I^{\langle\mathcal{F}\rangle}(\mathcal{S},\mathcal{Z})$ can be any function (e.g., taking into account the variance of $v_{\langle\mathcal{F}\rangle,\mathcal{S},\mathcal{Z}}$).

[7] In slight overload of notation, 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 $u_{\mathtt{D}}(\cdot) := -u_{\mathtt{A}}(\cdot)$.

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 5, 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 $U_I^{\langle\mathcal{F}\rangle,\vec{\gamma}}(\mathcal{S}, \mathcal{Z})$.

3 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 and $\mathcal{M}$ be the corresponding utility. 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 2. 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

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

Note that the notion of maximizing adversaries is specific to the protocol that is being attacked (see Remark 2 in Appendix B.1). The notion of maximizing adversaries naturally induces a notion of optimality for protocols. Intuitively, a protocol for implementing $\langle\mathcal{F}\rangle$ is optimal with respect to some attack model $\mathcal{M}$ if it minimizes the payoff with respect to $\mathcal{M}$-maximizing adversaries. More formally:

Definition 3. 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', $\hat{U}^{\Pi,\langle\mathcal{F}\rangle} \overset{\text{negl}}{\leq} \hat{U}^{\Pi',\langle\mathcal{F}\rangle}$.

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 5 below). We chose to nevertheless state the definition in the above simplified form and refer to Section 1.4 for further discussion.

The quantities $\hat{U}^{\Pi,\langle\mathcal{F}\rangle}$ and $\hat{U}^{\Pi',\langle\mathcal{F}\rangle}$ in Definition 3 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:

Theorem 4. *Let $\mathcal{M} = (\mathcal{F}, \langle\mathcal{F}\rangle, v)$ be an attack model and $\mathcal{G}_{\mathcal{M}}$ be the corresponding utility and attack game.A protocol Π is attack-payoff optimal in $\mathcal{M}$ if and only if the strategy profile $(\Pi, \mathtt{A})$ is λ-subgame-perfect equilibrium in $\mathcal{G}_{\mathcal{M}}$ for some negligible λ, where for each $\Pi \in \mathtt{ITM}^n$, $\mathtt{A}(\Pi)$ is a $\mathcal{M}$-maximizing adversary attacking Π.*

Proof. ($\Rightarrow$) The fact that $(\Pi, \mathtt{A})$ is λ-subgame perfect follows by a simple backwards-induction argument: The optimality of the attacker's choice $\mathtt{A}(\Pi)$ follows from the definition of the $\mathcal{M}$-maximizing adversary; the optimality of Π follows from the assumption that it is attack-payoff optimal in $\mathcal{G}_{\mathcal{M}}$.

($\Leftarrow$) For proving this direction we make use of Lemma 13 which states that the dummy adversary strategy, i.e., the adversary which simply relays messages to and from its environment, is in fact $\mathcal{M}$-maximizing for any protocol. Assume that $(\Pi, \mathtt{A})$ is λ-subgame perfect. Because the dummy strategy $\mathcal{D}$ is best response to any protocol, and Π is $\mathtt{D}$'s move in a subgame-perfect equilibrium, it must be that for any protocol Π', $\hat{U}^{\Pi,\langle\mathcal{F}\rangle}(\mathcal{D}) \overset{\text{negl}}{\leq} \hat{U}^{\Pi',\langle\mathcal{F}\rangle}(\mathcal{D})$. Lemma 13 now implies that $\hat{U}^{\Pi,\langle\mathcal{F}\rangle} \overset{\text{negl}}{\leq} \hat{U}^{\Pi',\langle\mathcal{F}\rangle}$; hence Π is attack-payoff optimal. □

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 5.2.

Definition 5. Let $\mathcal{M} = (\mathcal{F}, \langle\mathcal{F}\rangle, v)$ be an attack model and let Π be a protocol that realizes functionality $\langle\mathcal{F}\rangle$. Protocol Π is *attack-payoff secure* in $\mathcal{M}$ 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 (i.e., the protocol that forwards all inputs to and outputs from the functionality $\mathcal{F}$).

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 $\mathcal{F}$.

4 Protocol Composition

In this section we prove a composition theorem which, informally, allows for replacing an idealized subroutine (within a higher-level "hybrid" protocol) by its cryptographic implementation, without affecting the attack-payoff optimality of the higher-level protocols.

Subroutine replacement is essential for modular security proofs. In particular, modern cryptographic frameworks support the design of protocols in "hybrid" models in which certain idealized

subroutines or functionalities are available; these functionalities are then (implicitly or explicitly) replaced by protocols relying on more realistic assumptions by invocation of composition theorems [DM00, Can00, Can05, BPW03, BPW04, MR11]. For example, the overwhelming majority of the SFE literature assumes ideally secure (i.e., authenticated and private) channels and/or broadcast. Both these primitives can be implemented by use of cryptographic mechanisms, i.e., encryption and signatures, and/or appropriate protocols (e.g., the broadcast protocol from [DS82] relying on digital signatures) using standard insecure communication links (e.g., the Internet) and a public-key infrastructure (PKI). As we argue below, support of subroutine replacement has so far been lacking in existing rational computation models.

Composition theorem(s). Our model admits for protocol composition (in the sense of subroutine replacement). The composition theorem is stated informally below; the formal statement and proof can be found in Appendix B. We point out that the proof relies on a "dummy adversary lemma" (Lemma 13) akin to [Can05, Claim 10] which, as we show, also holds in our model (cf. Remark 2 in Appendix B). Such dummy adversary lemmas have proven to be of great use in traditional cryptographic definitions.

Theorem 6 (informal)**.** *Let $\mathcal{M} = (\mathcal{F}, \langle\mathcal{F}\rangle, v)$ be an attack model and 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 replacing in Π calls to $\mathcal{H}$ by invocations of protocol Ψ does not (noticeably) increase the utility of a $\mathcal{M}$-maximizing adversary.*

Theorem 6 can easily be extended to the case where Π is a $\{\mathcal{H}_1, \ldots, \mathcal{H}_{p(k)}\}$-hybrid protocol, for p a polynomial; this follows immediately from the same statement in the underlying model. Furthermore, as a corollary of Theorem 4 by applying the above subroutine replacement theorem, we show that such a replacement does not affect the stability of the solution in the corresponding attack game $\mathcal{G}_{\mathcal{M}}$. See Corollary 14 in Appendix B.

Our subroutine replacement theorem shows that in the underlying cryptographic model extends to our definitions: the subroutine replacement operation can be applied to protocols that (fully, e.g., UC-)implement a given functionality. If the underlying protocol is secure only in the sense that we can upper-bound the payoff of maximizing adversaries, 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 optimal protocols for replacing a subroutine. The formal statement and its proof can be found in Appendix B (Theorem 15).

(In)Composability in existing rational frameworks. To our knowledge, existing works on rational secure function evaluation make no statement about subroutine replacement. Although this does not imply that such a replacement is not possible in general, we demonstrate that replacement of a secure channel functionality in any of the rational function evaluation protocols/mechanisms from [HT04, KN08b, KN08a, OPRV09, FKN10] by a natural cryptographic implementation in fact destroys the equilibrium. In light of this observation, it seems there is no known way of replacing the (arguably unrealistic) assumptions of secure channels in the above protocols by the simpler assumption of existence of a PKI and insecure channels. The "incomposability" argument follows by using the same backward induction argument as in [KN08a]. In the following we give an informal description and refer to Appendix B.2 for details.

The above protocols have the following common structure: first, they use a cryptographic SFE protocol for implementing some given functionality (the functionality is different in each work); subsequently, the parties engage in an infinite-rounds revealment-protocol for computing

their output from their SFE outputs. Assume that all message transmission in these protocol are replaced by singcrypting the message and sending it over a non-private channel (i.e., a channel that every party can read). Now every player has the following strategy: In each round of the revealment protocol, he checks one cryptographic key for the corresponding sender (by using this key to decrypt and then verify the signature on the signcrypted message which is transmitted through the non-private channel). Clearly, after nK rounds (where n is the number of players and K is the size of the key-space), this player will have checked all the keys and thereby will be able to learn all the SFE (inputs and) outputs. Hence, in round nK every player is better off quitting and using the corresponding key for learning the output. This makes round $nK-1$ of the revealment protocol the last round of the whole protocol, and players have an incentive to deviate (quit) for the same reason. This process, known as backwards induction, can be repeated to show that players will remain silent in rounds $nK-2, nK-3, \ldots, 1$.

5 Scoring Privacy, Correctness, and the Cost of Corruption

In this section, we use our framework to study the feasibility of secure function evaluation (SFE) [GMW87] 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. [GJKY13]) 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. Note that the "costly corruption" assumption is orthogonal to the "costly computation" assumption explored in [HP10]; however, one could formulate such an assumption in the framework of [HP10] by considering an appropriate "complexity function" which assigns different complexity to different sets. Similarly, one can extend our results to consider (also) costly computation by using ideas from [GMPY06]. The implications of such an extension, and in particular on rational definitions of fairness, are the subject of ongoing research.

The ideal functionality $\mathcal{F}_{\text{SFE}}^{f}$ for SFE is parametrized by the function $f : \mathbb{F}^n \to \mathbb{F}$ (this form is without loss of generality—see, e.g., [LP09]) to be computed, receives the inputs of all parties, computes f, and hands all outputs back to the respective parties. Our protocols aim at realizing the fully secure version of SFE (i.e., including *robustness*) in which the parties always obtain their outputs (see Appendix C for a description of the SFE functionality).

Next, we specify how attacks with the above goals are modeled in our framework. Following the methodology described in Section 2, 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 $(\texttt{out}, \cdot)$ 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, $y := \perp$). To capture a minimal privacy-breaking event (cf. Remark 1), the functionality restricts the class of queries it accepts from the simulator. In particular, for a query $(\mathtt{inp}, \vec{x}_{\mathcal{I}})$, 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 $x_i \in \{x_i', \perp\}$. 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 $(\mathtt{inp}, \cdot)$ 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 $(\mathtt{inp}, \vec{x}_{\mathcal{I}})$ 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.[8]

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:

$$U_I^{\langle \mathcal{F}_{\text{SFE}} \rangle}(\mathcal{S}, \mathcal{Z}) := \gamma_{\mathsf{c}} \Pr[E_{\mathsf{c}}] + \gamma_{\mathsf{p}} \Pr[E_{\mathsf{p}}] - \sum_{\mathcal{I} \subseteq \mathcal{P}} \gamma_{\mathcal{I}} \Pr[E_{\mathcal{I}}],$$

where the probabilities are taken over the random coins of $\mathcal{S}$, $\mathcal{Z}$, and $\langle \mathcal{F}_{\text{SFE}} \rangle$.

Remark 1 (On breaking privacy). We choose one of the least severe ways for breaking privacy, namely, being able to query the functionality on different inputs for corrupted parties. Depending on how severe attacks one is willing to allow, alternative ways of breaking privacy can be considered (the most extreme of which would be so-called passive corruption of $\mathcal{F}_{\text{SFE}}$ [KKZZ11]). The advantage of proving optimality/security for our $\langle \mathcal{F}_{\text{SFE}} \rangle$ is that our results remain secure/optimal in settings where more severe privacy breaking occurs—indeed, a more severe privacy breaking capability would give the simulator more power, thereby making the simulation easier.

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, 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 (PKI) and a broadcast channel.[9]

5.1 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 in Section 5.1.1 with a possibility result for *static corruptions*[10] and small payoff for breaking privacy, i.e., $\gamma_{\mathsf{p}} < \lceil \frac{n}{2} \rceil \gamma_{\$}$. Subsequently, in Section 5.1.2, we prove impossibility of attack-payoff security when, for some $t \geq \frac{n}{2}$, $\gamma_{\mathsf{p}} > t\gamma_{\$}$ and $\gamma_{\mathsf{c}} > (n-t)\gamma_{\$}$. Finally, in Section 5.1.3, we show an attack-payoff secure protocol for a large class of $\vec{\gamma}$'s that are not excluded by the above impossibility, i.e., a protocol for $\gamma_{\mathsf{p}} + \gamma_{\mathsf{c}} < t\gamma_{\$}$ and $\gamma_{\mathsf{c}} < (n-t)\gamma_{\$}$. We point out that

[8] The corruption of parties is evident from the transcript. The event $E_{\mathcal{I}}$ corresponds to the situation where corruptions requests for parties with (*distinct*) IDs in $\mathcal{I}$ have occurred.

[9] For simplicity, as with a PKI broadcast is achievable tolerating any number of corruptions [DS82, HZ10, GKKZ11].

[10] 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.

this protocol is secure even with *adaptive* corruptions (where no erasures are assumed), and that (in contrast to Section 5.3) the protocols described here do not depend on the exact value of the score vector $\vec{\gamma}$, but rather relations among its components. In contrast, most works on rational computation rely on complete knowledge of the utility function (cf. [AL11]).

5.1.1 Feasibility for small privacy-breaking payoff: $\gamma_{\mathsf{p}} < \lceil \frac{n}{2} \rceil \gamma_{\$}$

The general idea of the protocol achieving attack-payoff security for $\gamma_{\mathsf{p}} < \lceil \frac{n}{2} \rceil \gamma_{\$}$ is that it suffices for the protocol to satisfy the following two properties: (1) Full security if up to $n/2$ parties are corrupted, and (2) correctness (i.e., the simulator never provokes the event E_{c}) for arbitrary many corruptions. Indeed, if these properties are satisfied, then the simulator might only need to provoke E_{p}, and this happens only when more than $n/2$ parties are corrupted. Therefore, if the adversary corrupts any party, the payoff will be upper-bounded by $\gamma_{\mathsf{p}} - \lceil \frac{n}{2} \rceil \gamma_{\$} < 0$. As a result, the best strategy is to not corrupt any party (in which case the payoff is 0).

In fact, as argued in [IKLP06], a "tweak" of their main SFE protocol achieves the following security guarantees that are well-suited for our goals: (1) If less than $n/2$ parties are corrupted, then the protocol is fully secure, and (2) for any $t < n$, any adversary corrupting t parties can be simulated by querying the ideal functionality at most t times. Intuitively, our construction builds on the idea of the above protocol with the following modifications:

- We augment it by a mechanism using (perfectly hiding) commitments and signatures which allows us to limit the privacy-breaking power of the adversary, i.e., the set of inputs on which it can force $\mathcal{S}$ to query the functionality (see bellow for details). The security of commitments and signatures relies the existence of enhanced trapdoor permutations (for formal definitions, see Appendix C).
- In [IKLP06], the protocol might abort only before the reconstruction of the output starts, in which case the protocol is restarted. In contrast, to allow for robustness even when $t \geq n/2$, our protocol may restart even in the reconstruction phase. This modification is necessary for ensuring that $\mathcal{S}$ does not need to provoke E_{c}.

Before providing the formal description of our protocol, denoted as $\Pi^{f}_{\text{St-SFE}}$, we remind the reader some concepts from the cryptographic protocols literature:

Security with identifiable abort. A protocol run by parties in some set $\mathcal{P}$ is said to be *secure with identifiable abort* if it either computes according to its specification, or it aborts with the index of some corrupted party $p_i \in \mathcal{P}$—i.e., every honest party learns p_i. Note that the protocol for adaptively secure multi-party computation by Canetti *et al.* [CFGN96], call it Π_{CFGN}, securely realizes any functionality with identifiable abort.[11]

Authenticated d-sharing. Our protocols use Shamir's secret sharing scheme augmented with digital signatures as an authentication mechanism. More precisely, a value $s \in \mathbb{F}$ is *authentically d-shared* among the parties in set $\mathcal{P}$ if there exists some polynomial g of degree d with $g(0) = s$ such that each $p_i \in \mathcal{P}$ holds a *share* $s_i = g(i)$. In addition to his share s_i, p_i holds a signature on s_i, where the public key is known to every $p_j \in \mathcal{P}$ (but no party knows the secret key).[12] Such an authenticated d-sharing scheme has the following properties:

[11] Alternatively, the protocol in [CLOS02] for UC security or the protocol in [GMW87] for static, stand-alone security.

[12] Here, the sharing is generated by (emulating) a functionality which also generates the key pairs and announces the public keys.

d-PRIVACY: The shares of any d parties give them no information on the shared value. This property follows trivially from the privacy of Shamir sharings.

COMMITMENT: No corrupted party can change its share, since modifying a share requires forging a signature. (Signatures contain unique protocol- and message IDs to prevent parties from "copying" shares.)

d-ROBUSTNESS: The secret can be reconstructed by having all parties announce their shares; if more than d parties announce valid (authenticated) shares, Lagrange interpolation reveals the secret.

We can now proceed to the description of our SFE protocol that is attack-payoff secure in the attack model $(\mathcal{F}_{\text{SFE}}^{f}, \langle \mathcal{F}_{\text{SFE}}^{f} \rangle, v^{\vec{\gamma}})$ for any given function f. Recall that in this subsection we assume static corruptions. The high-level idea is as follows. The players maintain a set D which includes all parties that have been publicly detected to misbehave (initially $D = \emptyset$). The evaluation of the function f proceeds in three distinct phases.

In a **first** phase, every party publicly commits to his input and broadcasts the commitment; subsequently, every party computes and broadcasts his signature on each of the commitments. (We use standard commitments and existentially unforgeable signatures for the static corruption case; to tolerate adaptive corruptions, in Section 5.1.3 we assume equivocal commitments.) If in this phase any party is caught misbehaving, it is disqualified from the computation and is included in D (its input is set to some default value). Let $\vec{S}$ denote the vector of all commitments and signatures. Observe that for each $p_i \in \mathcal{P} \setminus D$, $\vec{S}$ includes a commitment from p_i along with signatures on it from every party in $\mathcal{P} \setminus D$, and that $\vec{S}$ is publicly known, i.e., it is consistent among all parties in $\mathcal{P} \setminus D$. Hence, the signatures ensure that no corrupted party can create an ambiguity about the input of any party, honest or corrupted; indeed, to produce a new commitment the corrupted party will need the signatures of the honest parties on it.

In a **second** phase, the parties in $\mathcal{P} \setminus D$ pool the commitments and signatures gathered in the first phase to compute the following functionality $\mathcal{F}_{\text{COM-SFE}}$ (see below) with identifiable abort (by using protocol Π_{CFGN}, for example): $\mathcal{F}_{\text{COM-SFE}}$ receives the commitments and signatures; if the inputs of all parties in $\mathcal{P} \setminus D$ are uniquely defined by the received information, then $\mathcal{F}_{\text{COM-SFE}}$ computes an authenticated $(\lfloor \frac{n}{2} \rfloor - |D|)$-sharing of the output of f on these inputs and outputs the shares to the parties in $\mathcal{P} \setminus D$. Otherwise, $\mathcal{F}_{\text{COM-SFE}}$ uses the information it received to detect some misbehaving party $p_i \in \mathcal{P} \setminus D$ and halts by publicly announcing i; upon abort, the parties update D to include the index i and repeat this step. As breaking privacy pays less than the cost for corrupting $\lceil \frac{n}{2} \rceil$ parties, a rational attacker will not corrupt enough parties to learn the secret shared by $\mathcal{F}_{\text{COM-SFE}}$ in this phase (which requires $\lfloor \frac{n}{2} \rfloor + 1$ corruptions).

In the **third** (and final) phase, the parties attempt to publicly reconstruct the sharing created in the previous phase. If the reconstruction fails then the parties identify at least one party that did not broadcast a valid share, include him in the set D and go back to the beginning of the second phase with this updated setting. Because in each iteration a corrupt party is detected: if less than $n/2$ parties are corrupted the reconstruction will succeed, as there will be enough honest parties who announce correct shares, otherwise, the protocol rolls back to the second phase with a smaller player set (this can happen at most n times). Hence, the simulator never needs to provoke the event E_{c}.

For the remainder of the section, we consider a fixed commitment and a fixed signature scheme. We first describe the protocol $\Pi_{\text{St-SFE}}^{f}$ and the functionality $\mathcal{F}_{\text{COM-SFE}}$ which is used in Phase 2 of the protocols to compute sharings of the function f's output.

Functionality $\mathcal{F}^{f}_{\text{COM-SFE}}\left(\mathcal{P}, t, (\mathsf{pk}_1, \ldots, \mathsf{pk}_{|\mathcal{P}|})\right)$

The functionality is parametrized by a function $f : \mathbb{F}^{|\mathcal{P}|} \to \mathbb{F}$, a player set $\mathcal{P}$, a threshold $t \leq |\mathcal{P}|$, a non-interactive commitment scheme, a signature scheme, and a vector of public (verification) keys $(\mathsf{pk}_1, \ldots, \mathsf{pk}_{|\mathcal{P}|})$. $\mathcal{F}_{\text{COM-SFE}}$ proceeds in rounds/steps as described below, keeping a set D of detected parties which is initialized to $D := \emptyset$.

1. Every $p_i \in \mathcal{P}$ hands $\mathcal{F}_{\text{COM-SFE}}$ a vector $(\mathsf{com}_{i,1}, \ldots, \mathsf{com}_{i,|\mathcal{P}|})$ of commitments along with signatures on each $\mathsf{com}_{i,j}$ matching each of the keys $\mathsf{pk}_k, k \in \{1, \ldots, |\mathcal{P}|\}$; furthermore, every $p_i \in \mathcal{P}$ hands $\mathcal{F}_{\text{COM-SFE}}$ a decommitment dec_i. If some p_i does not send $|\mathcal{P}|$ commitments along with the all $|\mathcal{P}|^2$ corresponding valid signatures then output $(\texttt{detect}, p_i)$ and halt (if there are many such p_i's output the one with the smallest index).[a] Denote by $\vec{S}$ the set of all valid signed commitments.
2. $\mathcal{F}_{\text{COM-SFE}}$ computes the following set for each $p_i \in \mathcal{P}$:

   $$C_i := \{\mathsf{com}_{j,i} \mid \mathsf{com}_{j,i} \in \vec{S} \text{ has valid signatures for every } \mathsf{pk}_k, k \in [|\mathcal{P}|]\}$$

   If for some $p_i : |C_i| \neq 1$ then set $(\texttt{detect}, p_i)$ and halt (if there is many such p_i's output the one with the smallest index). Otherwise, for each $p_i \in \mathcal{P}$ denote by com_i the unique element of C_i.
3. For each p_i use dec_i to open com_i. If opening fails then issue $(\texttt{detect}, p_i)$ and halt. Otherwise, let x_i denote the opened value.
4. Compute $y := f(x_1, \ldots, x_n)$. Choose a polynomial $g \in \mathbb{F}[X]$ of degree at most t uniformly at random so that $g(0) = y$; for each $p_j \in \mathcal{P}$ set $s_j := g(j)$.
5. Use the key generation algorithm for generating a (fresh) signing/verification key-pair $(\mathsf{sk}, \mathsf{pk})$; for each $p_j \in \mathcal{P}$ compute a signature σ_j on s_j using key sk.
6. For each $p_j \in \mathcal{P}$ output $(s_j, \sigma_j, \mathsf{pk})$ to p_j.

[a] The signatures are assumed to have a unique party ID and message ID, which are checked to determine their validity.

Figure 1: The functionality $\mathcal{F}_{\text{COM-SFE}}$

Theorem 7. *Let $\gamma_{\mathsf{p}} < \lceil \frac{n}{2} \rceil \gamma_{\$}$. Assuming the existence of enhanced trapdoor-permutations, protocol $\Pi^{f}_{\text{St-SFE}}$ is attack-payoff secure in the attack model $(\mathcal{F}^{f}_{\text{SFE}}, \langle \mathcal{F}^{f}_{\text{SFE}} \rangle, v^{\vec{\gamma}})$ with static but arbitrarily many corruptions.*

Proof. We prove the statement in the hybrid world, where the protocol $\Pi_{\text{St-SFE}}$ uses a broadcast functionality $\mathcal{F}_{\text{BC}}$ (see Appendix C) and the version of functionality $\mathcal{F}_{\text{COM-SFE}}$ that allows for (identifiable) abort (i.e, in addition to those specified in its code, $\mathcal{F}_{\text{COM-SFE}}$ might accept from the adversary a special input $(\texttt{abort}, p_i)$, where p_i is a corrupted party; upon receiving this input, $\mathcal{F}_{\text{COM-SFE}}$ returns $(\texttt{abort}, p_i)$ to every party and halts). The security statement for the setting where $\mathcal{F}_{\text{BC}}$ and $\mathcal{F}_{\text{COM-SFE}}$ are replaced by invocations of protocols that securely realize them follows then directly by application of Theorem 6 which implies that this replacement might only reduce the adversary's utility. Recall that we also assume trapdoor (or more generally equivocal, e.g., UC) commitments and existentially unforgeable signatures, which we both obtain from the trapdoor permutations.

Protocol $\Pi^f_{\text{St-SFE}}$

PHASE 1 (INPUT COMMITMENT): The following steps are executed. Initially $D := \emptyset$:

1.1 Every party $p_i \in \mathcal{P}$ computes a commitment on his input and broadcasts it. Denote this value by com_i and the corresponding decommitment information (known exclusively to p_i) by dec_i. Any party that does not broadcast a commitment is added to D.

1.2 Every p_j signs (using his private signing key sk_j) all the commitments $\mathsf{com}_{\ell_1}, \ldots, \mathsf{com}_{\ell_{|\mathcal{P}\setminus D|}}$ broadcast by the parties in $\mathcal{P} \setminus D = \{p_{\ell_1}, \ldots, p_{\ell_{|\mathcal{P}\setminus D|}}\}$, and broadcasts these signatures. If some p_j broadcasts an inconsistent message or an invalid signature, then $D := D \cup \{p_j\}$.

PHASE 2 (COMPUTATION): Let $\vec{S}|_{\mathcal{P}\setminus D}$ denote the vector of all commitments and signatures from parties in $\mathcal{P}\setminus D$. Using an SFE protocol which is secure with identifiable abort for arbitrarily many corruptions (e.g., Π_{CFGN}), the parties in $\mathcal{P} \setminus D = \{p_{\ell_1}, \ldots, p_{\ell_{|\mathcal{P}\setminus D|}}\}$ evaluate the functionality $\mathcal{F}^f_{\text{COM-SFE}}(\mathcal{P} \setminus D, \lfloor \frac{n}{2} \rfloor - |D|, (\mathsf{pk}_{\ell_1}, \ldots, \mathsf{pk}_{\ell_{|\mathcal{P}\setminus D|}}))$ on input $\vec{S}|_{\mathcal{P}\setminus D}$. If the evaluation outputs $(\mathtt{detect}, p_i)$ or aborts with p_i then set $D := D \cup \{p_i\}$ and repeat Phase 2 in this updated setting.

PHASE 3 (OUTPUT): Let $s_{\ell_1}, \ldots, s_{\ell_{|\mathcal{P}\setminus D|}}$ be the shares output in Phase 2.

3.1 Every party broadcasts his share s_i along with the corresponding signature.

3.2 If at least $\lfloor \frac{n}{2} \rfloor - |D| + 1$ announced shares with valid signatures, then interpolate the corresponding polynomial and output the shared value. Otherwise, let D' denote the set of parties that announced no share or an invalid signature. Set $D := D \cup D'$ and repeat Phase 2 in this updated setting.

When no party is corrupted then the protocol $\Pi_{\text{St-SFE}}$ is fully secure, i.e., can be simulated without provoking any of the events E_{p} and E_{c}. Indeed, in that case all commitments and signatures on the inputs handed by the environment are properly generated in phase 1, and the functionality $\mathcal{F}_{\text{COM-SFE}}$ securely computes an authenticated $\lfloor \frac{n}{2} \rfloor$-sharing of the output in phase 2 (no abort or detection occurs as there is no corrupted party) which is then robustly reconstructed in phase 3. By the definition of the broadcast functionality $\mathcal{F}_{\text{BC}}$, no messages are leaked to the adversary during the protocol execution and the simulation is trivially perfect, so by corrupting no party the utility of the adversary is 0. In the remainder of the proof, we show that the best choice for the attacker is to refrain from corrupting any party, as this choice gives him 0 utility, in contrast to any other choice which gives him negative expected utility. For any number $0 < c < n$ of corrupted parties we design an $\langle \mathcal{F}^f_{\text{SFE}} \rangle$-hybrid simulator $\mathcal{S}$ that emulates the adversary's behavior and has negative expected ideal utility independent of the environment's strategy. In the remaining case $c = n$, the protocol can easily be simulated as the functionality does not provide guarantees; the expected utility is $-n\gamma_{\$} < 0$. Because the ideal expected utility of any $\mathcal{S}$ with the best environment upper-bounds the maximal utility of the adversary, this proves that the utility of the adversary will also be negative when $c > 0$.

In the following paragraphs we describe the simulator $\mathcal{S}$ which, when the adversary corrupts $c < n/2$ parties, does not provoke any of the events E_{p} and E_{c}, and therefore $\mathcal{A}$'s utility will be $-c\gamma_{\$} < 0$. If $\mathcal{A}$ corrupts $c \geq n/2$ parties then $\mathcal{S}$ will provoke the event E_{p} (but not E_{c}) in which case the utility of the adversary will again be at most $\gamma_{\mathsf{p}} - \lceil \frac{n}{2} \rceil \gamma_{\$} < 0$.

$\mathcal{S}$ invokes the adversary $\mathcal{A}$ as a black box (in particular it takes over $\mathcal{A}$'s interfaces to the hybrids $\mathcal{F}_{\text{BC}}$ and $\mathcal{F}_{\text{COM-SFE}}$). We consider two cases depending on the number c of corrupted parties (recall that we are in the static-corruption setting): (1) $c < n/2$, and (2) $c \geq n/2$.

Case 1 ($c < n/2$)**:**

Simulation of Phase 1: $\mathcal{S}$ emulates the behavior of $\mathcal{F}_{\text{BC}}$; $\mathcal{S}$ receives from $\mathcal{A}$ the commitments to be broadcast and signed. For simulating the announcing of commitments and signatures of honest parties, $\mathcal{S}$ computes (and outputs to the adversary) commitments to 0 for those inputs along with corresponding signatures. Note that the privacy of the commitment scheme ensures that $\mathcal{A}$'s view is indistinguishable from the corresponding view in the protocol where the commitments are on the actual inputs of the parties. Similarly to the protocol, $\mathcal{S}$ keeps track of parties that broadcast invalid messages in a set $\tilde{D}$ (initially $\tilde{D} := \emptyset$)

Simulation of Phase 2: $\mathcal{S}$ emulates the execution of $\mathcal{F}_{\text{COM-SFE}}$ as follows: it takes from $\mathcal{A}$ the vector $\vec{S}|_{\mathcal{P}\setminus\tilde{D}}$ and does (for the corrupted parties) the checks defined in $\mathcal{F}_{\text{COM-SFE}}$. If $\mathcal{F}_{\text{COM-SFE}}$ would halt with output $(\texttt{detect}, p_i)$, then forward this output to the adversary and set $\tilde{D} := \tilde{D} \cup \{p_i\}$. Otherwise, if $\mathcal{F}_{\text{COM-SFE}}$ would output the authenticated sharing of the output, $\mathcal{S}$ simulates the adversary's view on the output as follows: first invoke the key generation algorithm for generating simulated signing/verification keys $(\tilde{\mathsf{sk}}, \tilde{\mathsf{pk}})$; for each corrupted party, pick a random share $\tilde{s}_i$, compute a signature $\tilde{\sigma}_i$ on $\tilde{s}_i$ using $\tilde{\mathsf{sk}}$ and output $(\tilde{s}_i, \tilde{\sigma}_i, \tilde{\mathsf{pk}})$ to $\mathcal{A}$. Note that as the adversary corrupts $c < n/2$ parties, there will be at most $n/2 - |\tilde{D}|$ corrupted parties in $\mathcal{P} \setminus \tilde{D}$, which, as $\mathcal{F}_{\text{COM-SFE}}$ outputs a $(\lfloor \frac{n}{2} \rfloor - |\tilde{D}|)$-sharing, implies that the adversary's shares in the $\mathcal{F}_{\text{COM-SFE}}$-hybrid protocol look uniformly random, i.e., distributed as in the simulated execution. Finally, if in this phase $\mathcal{S}$ receives $(\texttt{abort}, p_i)$ from the adversary, then it sets $\tilde{D} := \tilde{D} \cup \{p_i\}$ and repeats the simulation of the phase in this updated setting.

Simulation of Phase 3: $\mathcal{S}$ simulates opening of the sharing as follows: $\mathcal{S}$ hands the inputs of the adversary to the ideal functionality $\langle \mathcal{F}_{\text{SFE}}^{f} \rangle$ and receives back the output y of the evaluation (on the actual honest parties' inputs). Subsequently, $\mathcal{S}$ emulates the shares of all honest parties as follows (let C denote the set of corrupted parties): $\mathcal{S}$ constructs a polynomial $\tilde{g}(\cdot)$ of degree (at most) $\lfloor \frac{n-|\tilde{D}|}{2} \rfloor$ with $\tilde{g}(0) = y$ and $\tilde{g}(i) = \tilde{s}_i$ for each $p_i \in C$. $\mathcal{S}$ sets the share of each (simulated) honest p_i to $\tilde{s}_i := \tilde{g}(i)$ and computes corresponding signature $\tilde{\sigma}_i := \tilde{sig}(\tilde{s}_i)$ with the key $\tilde{\mathsf{sk}}$ of the last iteration of Phase 2. Note that because $c < n/2$ parties (in total) are corrupted, the robustness of the sharing scheme ensures that the output will be reconstructed at that point.

Case 2 ($c \geq n/2$)**:**

Simulation of Phase 1: The simulation of this phase is identical to case 1.

Simulation of Phases 2 and 3: $\mathcal{S}$ emulates the execution of $\mathcal{F}_{\text{COM-SFE}}$ as follows: it takes from $\mathcal{A}$ the vector $\vec{S}|_{\mathcal{P}\setminus\tilde{D}}$ and does (for the corrupted parties) the checks defined if $\mathcal{F}_{\text{COM-SFE}}$. If $\mathcal{F}_{\text{COM-SFE}}$ would halt with output $(\texttt{detect}, p_i)$, then forward the output to the adversary and set $\tilde{D} := \tilde{D} \cup \{p_i\}$. Otherwise, if $\mathcal{F}_{\text{COM-SFE}}$ would output the actual authenticated sharing, $\mathcal{S}$ simulates the adversary's view of the output as follows: $\mathcal{S}$ provokes the event E_{p} and queries $\langle \mathcal{F}_{\text{SFE}}^{f} \rangle$ on the inputs corresponding to this execution: i.e., $\perp$ for parties in $\tilde{D}$ and the actual input received from the adversary for the remaining corrupted parties (observe that for those parties their input commitments have been opened by $\mathcal{A}$ in this iteration). $\mathcal{S}$ then receives from $\langle \mathcal{F}_{\text{SFE}}^{f} \rangle$ the output of the function evaluated on this inputs and computes (simulated) shares for both the corrupted and the honest parties as in the simulation of the phase 3 in the previous case (case 1). Having these shares, $\mathcal{S}$ can easily emulate the opening. If $\mathcal{S}$ receives $(\texttt{abort}, p_i)$ from the adversary, then it sets $\tilde{D} := \tilde{D} \cup \{p_i\}$ and repeats the simulation of the phase in this updated setting. Because each

time phase 2 is repeated a sharing of a smaller degree is computed, after at most n iterations this process will finish; the simulator will hand to $\langle \mathcal{F}_{\text{SFE}}^{f} \rangle$ the inputs of the corrupted parties that are not in $\tilde{D}$ and halt.

$\mathcal{S}$ is a good simulator: in Phase 1, the only difference between the real and ideal world transcripts is that the commitments of the honest parties are on 0 instead of the actual inputs, but as the commitment scheme is perfectly hiding, the simulated transcript has (up to this point) the same distribution as the protocol transcript. Furthermore, in Appendix D be prove a result about the security of $\Pi_{\text{St-SFE}}^{f,t}$ (Claim 2) which ensures that (in each iteration of phase 2), unless a signature is forged, $\Pi_{\text{St-SFE}}^{f,t}$ either aborts with an index of a corrupted party or provides the correct output—in either case the simulator produces a transcript with the correct distribution. Moreover, unless the adversary breaks the binding property of the commitment scheme, the inputs to the function computed within $\mathcal{F}_{\text{COM-SFE}}$ remain the same in each iteration of phases 2 and 3, so the simulator can use the privacy breaking capability of $\langle \mathcal{F}_{\text{SFE}}^{f} \rangle$. (A formal proof of course requires to construct from an adversary that provokes these differences in the transcript an adversary against the binding property of the commitment scheme or the unforgeability of the signature scheme, respectively.)

To conclude the proof, we observe that: (1) In Case 1, the simulator does not provoke any of the events E_{c} or E_{p} and therefore his expected utility is $-c\gamma_{\$} < 0$. In Case (2), the simulator provokes E_{p} but not E_{c} and therefore his expected utility is $\gamma_{\mathsf{p}} - c\gamma_{\$} < 0$. Hence, in any case, if the adversary corrupts at least one party his maximal utility will be negative, which implies that a rational attacker will corrupt no party and obtain utility 0. □

5.1.2 A Broad Impossibility Result

If $\gamma_{\mathsf{p}} \geq \lceil \frac{n}{2} \rceil \gamma_{\$}$, then for a large selection of payoff vectors $\vec{\gamma}$, attack-payoff secure protocols for the attack model $(\mathcal{F}_{\text{SFE}}, \langle \mathcal{F}_{\text{SFE}} \rangle, v^{\vec{\gamma}})$ do not exist. We show this by showing impossibility for computing a specific function. Our result makes use of the following impossibility result by Katz [Kat07].

Fact 1 ([Kat07, Theorem 1])**.** *There is a finite, deterministic functionality $\mathcal{F}_{\text{K07}}$ for which there is no polynomial-round protocol Π that simultaneously satisfies the following two properties: (1) Π securely realizes $\mathcal{F}_{\text{K07}}$ in the presence of t corrupted parties, and (2) Π is $(n-t)$-private.*

Unfortunately, we cannot use the result in [Kat07] directly as it uses a standard definition of privacy (roughly speaking, it requires that *just* the output of an adversary attacking the protocol can be simulated by an $\mathcal{F}$-ideal simulator; for completeness, we include the formal definition as Definition 19 in Appendix C). Still, it is not hard to show that if a protocol is simulatable by a $\langle \mathcal{F} \rangle$-ideal simulator which never sends to $\langle \mathcal{F} \rangle$ the command $(\texttt{query}, \cdot)$, then it is private according to the notion in [Kat07]. We state and prove this implication in Lemma 20 (see Appendix C), which allows us to demonstrate that when $\gamma_{\mathsf{p}} \geq \lceil \frac{n}{2} \rceil \gamma_{\$}$ there are no attack-payoff secure protocols for arbitrary functions. The proof idea is to show that existence of a protocol which is attack-payoff secure in the respective attack model would contradict Fact 1.

Theorem 8. *Let $t \geq n/2$. If $\gamma_{\mathsf{p}} > \gamma_{\$} t$ and $\gamma_{\mathsf{c}} > \gamma_{\$}(n-t)$, then there exists no (polynomial-round) attack-payoff secure protocol Π in the attack model $(\mathcal{F}_{\text{K07}}, \langle \mathcal{F}_{\text{K07}} \rangle, v^{\vec{\gamma}})$.*

Proof. Assume, towards contradiction, that such a protocol Π exists. Because Π is attack-payoff secure in the attack model $(\mathcal{F}_{\text{K07}}, \langle \mathcal{F}_{\text{K07}} \rangle, v^{\vec{\gamma}})$, for any $\mathcal{M}$-maximizing adversary we have $\hat{U}^{\Pi, \langle \mathcal{F}_{\text{K07}} \rangle} \overset{\text{negl}}{\leq} 0$. We consider two cases, depending on the actual number c of corrupted parties (we assume that the adversary is static).

(1) $c \leq n-t$ parties are corrupted. In this case, protocol Π must be simulatable by a simulator that does not provoke any of the events E_{p} and E_{c}, as provoking any of those events (with noticeable probability) would give the adversary a positive payoff, thus contradicting the fact that $\hat{U}^{\Pi,\langle\mathcal{F}_{\text{K07}}\rangle} \overset{\text{negl}}{\leq} 0$. But this implies that in this case Π is simulatable in the $\mathcal{F}_{\text{K07}}$-ideal model, which implies that it securely realizes $\mathcal{F}_{\text{K07}}$.

(2) $n-t < c \leq t$ parties are corrupted. In this case, protocol Π must be simulatable by a simulator that does not provoke the events E_{p}, as provoking this event (with noticeable probability) would give the adversary positive payoff contradicting the fact that $\hat{U}^{\Pi,\langle\mathcal{F}_{\text{K07}}\rangle} \overset{\text{negl}}{\leq} 0$. But, by Lemma 20, this implies that Π is private according to the definition of privacy from [Kat07]. The existence of such a protocol, however, contradicts Fact 1. □

5.1.3 Feasibility with Adaptive Corruption

We now show an attack-payoff secure protocol for a large class of payoff vectors that are not excluded by the above impossibility result, namely, $\gamma_{\mathsf{c}} + \gamma_{\mathsf{p}} < t\gamma_{\$}$ and $\gamma_{\mathsf{c}} < t\gamma_{\$}$, for $t \geq n/2$. The protocol is even secure with respect to *adaptive corruptions*.

The protocol works similarly to the protocol $\Pi_{\text{St-SFE}}$ with the following differences:

- In phase 2: the functionality $\mathcal{F}_{\text{COM-SFE}}$ is necessarily evaluated by use of an adaptively secure protocol with identifiable abort (e.g., [CFGN96, CLOS02]). Furthermore, the computed sharings are of degree $t - |D|$, where t is a parameter of the protocol (note that t can be computed for the protocol using knowledge of $\vec{\gamma}$). This will ensure that it is too expensive for the the adversary to provoke the event E_{p} as this will require corrupting more than t parties.
- In phase 3, if the reconstruction aborts, then the protocol also aborts (and every party outputs a default value). This will ensure that when less than t parties are corrupted, the simulator must at most provoke the less rewarding event E_{c}.

A detailed description of the adaptively attack-payoff secure protocol $\Pi_{\text{Ad-SFE}}$ along with its security proof is given in the following.

The following theorem states the security achieved by protocol $\Pi^{f,t}_{\text{Ad-SFE}}$. As a technical remark, we note that we implicitly assume that the SFE functionality $\mathcal{F}_{\text{SFE}}$ that we aim in computing is *adaptively well-formed* which, roughly speaking, means that, if all parties become corrupted, $\mathcal{F}_{\text{SFE}}$ outputs all its inputs and randomness to the simulator (see [CLOS02] for a detailed description).

Theorem 9. *Let $t \geq n/2$ and let $\mathcal{M} = (\mathcal{F}^f_{\text{SFE}}, \langle\mathcal{F}^f_{\text{SFE}}\rangle, v^{\vec{\gamma}})$ 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 the protocol $\Pi^{f,t}_{\text{Ad-SFE}}$ is attack-payoff secure in $\mathcal{M}$.*

Proof. As in the proof of Theorem 7 we prove that the utility of an $\mathcal{M}$-maximizing adversary attacking protocol $\Pi^f_{\text{Ad-SFE}}$ is 0. We provide the simulator for the protocol being executed in a hybrid world where the parties have access to a broadcast functionality $\mathcal{F}_{\text{BC}}$ and the version of functionality $\mathcal{F}_{\text{COM-SFE}}$ which allows for (identifiable) abort (i.e, in addition to its code $\mathcal{F}_{\text{COM-SFE}}$ might accept from its adversary a special input $(\texttt{abort}, p_i)$, where p_i is a corrupted party; upon receiving this input, $\mathcal{F}_{\text{COM-SFE}}$ returns $(\texttt{abort}, p_i)$ to every party and halts). Recall that we assume trapdoor (or more generally equivocal, e.g., UC) commitments. Note, also that $\mathcal{F}_{\text{BC}}$ can be fully securely implemented by use of the protocol from [DS82].The original statement, for the setting where $\mathcal{F}_{\text{BC}}$ and $\mathcal{F}_{\text{COM-SFE}}$ are replaced by invocations of protocols that securely realize them follows

Protocol $\Pi^{f,t}_{\text{Ad-SFE}}$

PHASE 1 (INPUT COMMITMENT): The following steps are executed. Initially $D := \emptyset$:

1.1 Every party $p_i \in \mathcal{P}$ computes a commitment on his input and broadcasts it. Denote this value by com_i and the corresponding decommitment information (known exclusively to p_i) by dec_i. Any party that does not broadcast a commitment is added to D.

1.2 Every p_j signs (using his private signing key sk_j) all the commitments $\mathsf{com}_{\ell_1}, \ldots, \mathsf{com}_{\ell_{|\mathcal{P}\setminus D|}}$ broadcast by the parties in $\mathcal{P} \setminus D = \{p_{\ell_1}, \ldots, p_{\ell_{|\mathcal{P}\setminus D|}}\}$, and broadcasts these signatures. If some p_j broadcasts an inconsistent message or an invalid signature, then $D := D \cup \{p_j\}$.

PHASE 2 (COMPUTATION): Let $\vec{S}|_{\mathcal{P}\setminus D}$ denote the vector of all commitments and signatures from parties in $\mathcal{P} \setminus D$. Using an SFE protocol which is secure with identifiable abort for arbitrarily many corruptions (e.g., the protocol from [CFGN96]), the parties in $\mathcal{P} \setminus D = \{p_{\ell_1}, \ldots, p_{\ell_{|\mathcal{P}\setminus D|}}\}$ evaluate the functionality $\mathcal{F}^f_{\text{COM-SFE}}(\mathcal{P} \setminus D, t - |D|, (\mathsf{pk}_{\ell_1}, \ldots, \mathsf{pk}_{\ell_{|\mathcal{P}\setminus D|}}))$ on input $\vec{S}|_{\mathcal{P}\setminus D}$. If the evaluation outputs $(\texttt{detect}, p_i)$ or aborts with p_i then set $D := D \cup \{p_i\}$ and repeat Phase 2 in this updated setting.

PHASE 3 (OUTPUT): Let $s_{\ell_1}, \ldots, s_{\ell_{|\mathcal{P}\setminus D|}}$ be the shares output in Phase 2.

3.1 Every party broadcasts his share s_i along with the corresponding signature.

3.2 If there are at least $t - |D| + 1$ announced shares with valid signatures, then interpolate the corresponding polynomial and output the shared value[a]. Otherwise, abort and have every party output a default value, e.g. 0.

[a] Note that as long as there is an honest party, only correct shares can be announced, because to announce a wrong share, the adversary has to forge the trusted party's signature on it.

Figure 2: *Adaptively attack-payoff secure protocol for* $\gamma_{\mathsf{c}} + \gamma_{\mathsf{p}} < t\gamma_{\$}$ *and* $\gamma_{\mathsf{c}} < t\gamma_{\$}$, $t \geq n/2$.

then directly by application of Theorem 6 which implies that this replacement might only reduce the adversary's utility.

By inspection of the protocol it is straightforward to verify that when no party is corrupted then the protocol $\Pi_{\text{Ad-SFE}}$ is fully secure, i.e., can be simulated without provoking any of the events E_{p} and E_{c}. Indeed, in that case all commitments and signatures on the inputs handed by the environment are properly generated in phase 1, and the functionality $\mathcal{F}_{\text{COM-SFE}}$ securely computes a t-sharing of the output in phase 2 (no abort or detection occurs as there is no corrupted party) which is then robustly reconstructed in phase 3. Hence, by corrupting no party the utility of the adversary is 0. In the remainder of the proof, we show that if the adversary corrupts $c > 0$ parties, then his expected utility is negative; hence a rational attacker will corrupt no party and obtain utility 0.

Let $\mathcal{A}$ be an adversary attacking protocol $\Pi_{\text{Ad-SFE}}$. We describe a simulator $\mathcal{S}$ for this $\mathcal{A}$. The invariant in the simulation is that as long as $\mathcal{A}$ corrupts $c < n - t$ parties, $\mathcal{S}$ does not provoke any of the events E_{p} and E_{c}, and therefore the utility of $\mathcal{A}$ in this case will be at most $-c\gamma_{\$} < 0$. If $\mathcal{A}$ corrupts c parties where $(n - t) < c \leq t$ then $\mathcal{S}$ might at most provoke the event E_{c} in which case the utility of the adversary will again be at most $\gamma_{\mathsf{c}} - c\gamma_{\$} < 0$. Else, if $\mathcal{A}$ corrupts $c > t$ parties then $\mathcal{S}$ might both E_{p} and E_{c}, but the utility will again be $\gamma_{\mathsf{c}} + \gamma_{\mathsf{p}} - c\gamma_{\$} < 0$. Hence, the adversary is always better off corrupting no party and obtaining utility 0. The description of $\mathcal{S}$ follows:

$\mathcal{S}$ invokes the adversary $\mathcal{A}$ as a black box (in particular it takes over $\mathcal{A}$'s interfaces to the hybrids $\mathcal{F}_{\text{BC}}$ and $\mathcal{F}_{\text{COM-SFE}}$). Similarly to the protocol, $\mathcal{S}$ keeps track of parties that broadcast invalid messages in a set $\tilde{D}$ (initially $\tilde{D} := \emptyset$). Note that, later on, $\mathcal{S}$ needs to keep track of when $\tilde{D}$ is updated but he can do that by analyzing his view of the simulation.

Simulation of phase 1: $\mathcal{S}$ receives from $\mathcal{A}$ the commitments to be broadcast and the corresponding signatures. If some party sends an invalid signature he is added in $\tilde{D}$. For simulating the announcing of commitments and signatures of honest parties, $\mathcal{S}$ computes (and outputs to the adversary) commitments to 0 for those inputs along with corresponding signatures.[13] If $\mathcal{A}$ requests to corrupt some party that has already sent his commitment (in the simulated protocol), $\mathcal{S}$ corrupts this party in the ideal world, learns his input, and uses the trapdoor (or, more generally the equivocability) of the commitment scheme to open the commitment to this input. Note that the privacy of the commitment scheme ensures that $\mathcal{A}$'s view is indistinguishable from the corresponding view in the protocol where the commitments are on the actual inputs of the parties. Also, $\mathcal{S}$ emulates the behavior of the broadcast channel. Similarly to the protocol, $\mathcal{S}$ keeps track of parties that broadcast invalid messages. However, unlike the protocol, $\mathcal{S}$ needs to keep track of the order in which the parties were disqualified.

Simulation of phase 2: $\mathcal{S}$ emulates the execution of $\mathcal{F}_{\text{COM-SFE}}$ as follows: It takes from $\mathcal{A}$ the vector $\vec{S}$ and does (for the corrupted parties) the same checks as $\mathcal{F}_{\text{COM-SFE}}$ would do: If $\mathcal{F}_{\text{COM-SFE}}$ would halt with output $(\texttt{detect}, p_i)$, then forward the output to the adversary and set $\tilde{D} := \tilde{D} \cup \{p_i\}$. Otherwise, if $\mathcal{F}_{\text{COM-SFE}}$ would output the actual authenticated sharing, $\mathcal{S}$ simulates the adversary's view on the output as follows: first he invokes the key generation algorithm for generating simulated signing/verification keys $(\tilde{\mathsf{sk}}, \tilde{\mathsf{pk}})$; as long as at most t parties have been corrupted, for each corrupted party $\mathcal{S}$ picks a random share $\tilde{s}_i$, computes a signature $\tilde{\sigma}_i$ on $\tilde{s}_i$ using $\tilde{\mathsf{sk}}$ and outputs $(\tilde{s}_i, \tilde{\sigma}_i, \tilde{\mathsf{pk}})$ to $\mathcal{A}$. Note that as long as the adversary corrupts less than t parties, there will be at most $t - |\tilde{D}|$ corrupted parties in $\mathcal{P} \setminus \tilde{D}$,[14] which implies that the adversary's

[13] Recall that $\mathcal{S}$ controls the PKI towards the adversary, and can therefore generate signatures on arbitrary messages.

[14] In slight abuse of notation we might at times use $\tilde{D}$ as the set of the parties included in any of its (sub)components.

shares in the $\mathcal{F}_{\text{COM-SFE}}$-hybrid protocol look uniformly random, i.e., distributed as in the simulated execution. If the adversary, in this phase, exceeds the threshold of t corruption, then he might learn the output of this and all past attempts to evaluate $\mathcal{F}_{\text{COM-SFE}}$ which resulted in abort (as a result of learning one more sharing than the thresholds). Hence $\mathcal{S}$ needs to simulate this view (i.e., the share of the extra party) for each such attempt. This is done as follows: Let $\tilde{D}_\ell$ denote the set of detected parties at the at the beginning of the ℓth attempt. The binding property of the commitments and the unforgeability of the digital signatures ensures that the adversary cannot arbitrarily modify the input of parties he corrupts (the only thing he can do is have them added to D and set their input to a default value). Hence $\mathcal{S}$ knows exactly on which corrupted-party inputs to query the functionality and no extraction is required. In particular, $\mathcal{S}$ sends the command $(\texttt{query}, \vec{x}^{(\ell)})$ to the relaxed functionality $\langle \mathcal{F}_{\text{SFE}} \rangle$ (thereby provoking the event E_{p}), where $\vec{x}^{(\ell)}$ is the vector containing for the corrupted parties in $\mathcal{P} \setminus \tilde{D}_\ell$ the inputs received from the adversary (observe, that for those parties their input commitments are correctly opened by $\mathcal{A}$ in this iteration), and for all other corrupted parties the input $\perp$; $\mathcal{S}$ receives back the output y_ℓ of the evaluation of f on these inputs (along with the honest inputs for all other parties). Using y_ℓ, $\mathcal{S}$ emulates the shares of all parties that were not corrupted before the t-th corruption-request as follows: let C_ℓ denote the set of (indices of) the first t corruptions and $(s_i^{(\ell)}, \sigma_i^{(\ell)}, \mathsf{pk}^{(\ell)})$ the value $\mathcal{S}$ has given to $\mathcal{A}$ for each of these first t corrupted parties $p_i \in C_\ell$; $\mathcal{S}$ constructs a polynomial $\tilde{g}^{(\ell)}(\cdot)$ of degree $t - |\tilde{D}_\ell|$ with $\tilde{g}^{(\ell)}(0) = y_\ell$ and $\tilde{g}^{(\ell)}(i) = \tilde{s}_i^{(\ell)}$ for each $p_i \in C$. $\mathcal{S}$ sets the share of each simulated $p_i \in \mathcal{P} \setminus (C \cup \tilde{D}_\ell)$ as $\tilde{s}_i^{(\ell)} := \tilde{g}^{\ell)}(i)$ and computes corresponding signature $\tilde{\sigma}_i^{(\ell)}$ with the key $\tilde{\mathsf{sk}}^{(\ell)}$ (matching the verification key $\tilde{\mathsf{pk}}^{(\ell)}$ of this round) corresponding to this iteration of the evaluation of $\mathcal{F}_{\text{COM-SFE}}$. For every corruption request of some p_i (beyond the t threshold) $\mathcal{S}$ hands $(\tilde{s}_i^{(\ell)}, \tilde{\sigma}_i^{(\ell)}, \tilde{\mathsf{pk}}^{(\ell)})$ to the adversary, and opens the commitment corresponding to p_i as x_i (again using the equivocability of the commitment scheme). Finally, if in this phase $\mathcal{S}$ receives $(\texttt{abort}, p_i)$ from the adversary,[15] then it sets $\tilde{D} := \tilde{D} \cup \{p_i\}$ and repeats the simulation of the phase in this updated setting.

Simulation of phase 3: $\mathcal{S}$ simulates the openings of the sharing as follows (where $\tilde{D}$ denotes the set of detected parties at the beginning of this phase): If there are $c > t$ corrupted parties at the beginning of this phase, then $\mathcal{S}$ has already provoked the event E_{p}, and has generated simulated shares for all honest parties as described in the simulation of phase 2. Otherwise, $\mathcal{S}$ hands the inputs of the adversary to the ideal functionality $\langle \mathcal{F}_{\text{SFE}}^f \rangle$ and receives back the output y of the evaluation (on the actual honest parties' inputs). Using y, $\mathcal{S}$ emulates the shares of all parties that were not corrupted at the beginning of phase 3 similar to the simulation of phase 2. Subsequently, for each honest p_i, $\mathcal{S}$ simulates the opening of p_i's share by emulating towards $\mathcal{A}$ the broadcasting of $(\tilde{s}_i, \tilde{\sigma}_i)$. Note that as long as at least $t - |\tilde{D}| + 1$ parties send validly signed shares the robustness of the sharing scheme ensures that the reconstruction will succeed. Therefore, less than$(t - |\tilde{D}| + 1) - (n - c)$ corrupted parties announce (i.e., send to $\mathcal{S}$) validly signed shares, $\mathcal{S}$ sends $\mathcal{F}$ the command $(\texttt{modify-output}, v)$ where v is the default value that the parties output upon abort in this phase. Otherwise, allow the functionality to deliver the outputs. Observe, that an adaptive adversary corrupting $n - t < c \leq t$ parties might force the protocol to abort, and then perform some post-execution corruption of more than t parties (in total). In that case, the adversary learns the actual function output (in particular the shares of the newly corrupted parties that define those outputs) along with the outputs of all aborted iterations of phase 2. To simulate this view, the simulators proceeds as in phase 2 (i.e., provokes E_{p} and emulates all sharings).

[15] Recall that we use a protocol for computing $\mathcal{F}_{\text{COM-SFE}}$ that is secure with abort.

By inspection, one can verify that the above is a good simulator as he produces the same transcript as the actual protocol, with the only difference that in phase 1 the commitments of the honest parties are on 0 instead of the actual inputs. Indeed, the hiding property of the commitment scheme ensures that the simulated transcript is indistinguishable from the protocol transcript. Furthermore, Claim 2 ensures that as long as no signature is forged and no commitment is falsely opened (i.e., with overwhelming probability) whenever $\Pi^{f,t}_{\text{Ad-SFE}}$ does not abort the sharing of the protocol is identically distributed as the output of $\mathcal{F}^{f}_{\text{SFE}}$ on the corresponding inputs

We show that the expected ideal utility of $\mathcal{S}$, which is an upper-bound on the utility of the adversary, is always negative when the adversary corrupts $c > 1$ parties:

1. If at the end of the protocol $c < n - t$ then $\mathcal{S}$ does not provoke the events E_{c} or E_{p} during the protocol simulation, and might be at most provoke the event E_{p} and this only if the adversary corrupts $c' > t$ parties during a post-execution corruption phase. Hence the expected ideal utility in this case is at most $\gamma_{\mathsf{p}} - c'\gamma_{\$} < 0$.
2. Otherwise (i.e. if at some point in the simulation $c \geq n - t$) then we consider two subcases:
   1. If the emulated protocol aborts in phase 3, then: if $\mathcal{A}$ never corrupts $t + 1$ parties (i.e,. not even after the execution) then the simulator needs to provoke the event E_{c} but not E_{p} (the expected utility will be at most $\gamma_{\mathsf{c}} - c\gamma_{\$} < 0$); otherwise, $\mathcal{S}$ might need to provoke both events but as $c' > t$ parties are corrupted the expected utility is again negative.
   2. If the protocol does not abort in phase 3: then: if $\mathcal{A}$ never corrupts $t+1$ parties (i.e,. not even after the execution) then the simulator needs to provoke none of the events E_{c} and E_{p} (the expected utility will be at most $-c\gamma_{\$} < 0$); otherwise, $\mathcal{S}$ might need to provoke both events but as $c' > t$ parties are corrupted the expected utility is again negative.

   Note that in any case, if at any point in the simulation the adversary corrupts every party in $\mathcal{P}$, then the adaptive well-formedness of $\mathcal{F}_{\text{SFE}}$ ensures that $\mathcal{S}$ will receive all the inputs and randomness and will therefore be able to complete his simulation without provoking any additional events. Hence, corrupting all parties is also not a strategy of a rational attacker.

□

5.2 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 3) in settings where attack-payoff security is provably unachievable. More precisely, consider the case of secure two-party computation (i.e., $n = 2$), where the values γ_{p} and γ_{c} are both greater than the cost $\gamma_{\$}$ of corrupting a party. Theorem 8 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 $\Pi_{\mathtt{0p-SFE}}$, consists of two phases; let f denote the function to be computed:

1. $\Pi_{\mathtt{0p-SFE}}$ uses a protocol for SFE with abort (e.g., [CFGN96, CLOS02]) to compute the following function f': f' takes as input the inputs of the parties to f and outputs an authenticated 1-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_{3-i}.

The adversary's payoff in the above protocol is upper-bounded by $\frac{\min\{\gamma_{\mathsf{p}},\gamma_{\mathsf{c}}\}}{2}-\gamma_{\$}$, which is stated in the following theorem. The intuition of the proof is the following: 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.

Theorem 10. *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* $\Pi_{\mathtt{0p-SFE}}$ *in* $\mathcal{M}$, $\hat{U}^{\Pi_{\mathtt{0p-SFE}},\langle\mathcal{F}_{\text{SFE}}\rangle}\overset{\text{negl}}{\leq}\frac{\min\{\gamma_{\mathsf{p}},\gamma_{\mathsf{c}}\}}{2}-\gamma_{\$}$.

Proof. As in Theorems 7 and 9 we prove the statement in the hybrid world, where the protocol $\Pi_{\mathtt{0p-SFE}}$ uses the version of functionality $\mathcal{F}_{\text{SFE}}^{f'}$ which allows for identifiable abort (denote it by $\mathcal{F}_{\text{SFE}}^{f',\perp}$). The security statement for the setting where $\mathcal{F}_{\text{COM-SFE}}$ is replaced by invocation of a protocol that securely realize it (e.g., [CFGN96, CLOS02]) follows then directly by application of Theorem 6 which implies that this replacement might only reduce the adversary's utility.

If both parties are corrupted, then the adversary's payoff is $-2\gamma_{\$}$; if no party is corrupted, the payoff is 0. Assume for the remainder of the proof that the adversary corrupts p_1 (the case where the adversary corrupts p_2 is dealt with symmetrically). Note that by assumption this is the functionality that is securely realized in the first phase of our protocol. Let $\mathcal{A}$ denote the adversary's strategy and let $\mathcal{S}_{\mathcal{A}}$ denote the following (black-box straight-line) simulator. To emulate the output of $\mathcal{F}_{\text{SFE}}^{f',\perp}$, $\mathcal{S}_{\mathcal{A}}$ does the following (recall that the output consists of a share for p_1 and a uniformly chosen index $i\in\{1,2\}$): $\mathcal{S}_{\mathcal{A}}$ generates a signature key pair (κ_s,κ_v), randomly picks an index $\hat{i}\in_R\{1,2\}$, and outputs a random element $\hat{s}_1\in\mathbb{F}$ along with κ_v, a signature on s_1, and the index $\hat{i}$. Subsequently, $\mathcal{S}_{\mathcal{A}}$ does the following to simulate the opening stage of $\Pi_{\mathtt{0p-SFE}}$:

- If $i=1$, then $\mathcal{S}_{\mathcal{A}}$ sends to $\langle\mathcal{F}_{\text{SFE}}^{f}\rangle$ the value $\hat{x}_1$ which $\mathcal{A}$ inputs to $\mathcal{F}_{\text{SFE}}^{f,\perp}$ and asks for the output[16]; let y denote this output. $\mathcal{S}_{\mathcal{A}}$ computes a share for p_2 which, together with the simulated share of p_1, results in a valid sharing of y, as $\hat{s}_2:=y-\hat{s}_1$ with a valid signature (using κ_s). $\mathcal{S}_{\mathcal{A}}$ then sends the signed share to p_1 for reconstructing the sharing of y. In the next round, receive from $\mathcal{A}$ p_1's share; if $\mathcal{S}_{\mathcal{A}}$ receives a share other than $\hat{s}_1$ or an invalid signature, then $\mathcal{S}_{\mathcal{A}}$ sends `abort` to $\langle\mathcal{F}_{\text{SFE}}^{f}\rangle$, before the honest party is allowed to receive the output.
- If $i=2$ then $\mathcal{S}_{\mathcal{A}}$ receives from $\mathcal{A}$ p_1's share. If $\mathcal{S}_{\mathcal{A}}$ receives a share other than $\hat{s}_1$ or an invalid signature, then it sends a default value to $\langle\mathcal{F}_{\text{SFE}}^{f}\rangle$ (as p_1's input). Otherwise, it asks $\langle\mathcal{F}_{\text{SFE}}^{f}\rangle$ for p_1's output y, and computes a share for p_2 which, together with the simulated share of p_1, results in a valid sharing of y (as above). $\mathcal{S}_{\mathcal{A}}$ sends this share to $\mathcal{A}$.

As the simulated keys and shares are distributed identically to the actual sharing in the protocol execution, $\mathcal{S}_{\mathcal{A}}$ is a good simulator for $\mathcal{A}$.

[16] Recall that we assume wlog that f has one global output.

We argue that for any adversary $\mathcal{A}$, the score of $\mathcal{S}_{\mathcal{A}}$ is (at most) $\frac{\min\{\gamma_{\mathsf{p}},\gamma_{\mathsf{c}}\}}{2} - \gamma_{\$} + \mu$ for some negligible function μ: If $\mathcal{A}$ makes $\mathcal{F}_{\mathrm{SFE}}^{f',\perp}$ abort, the simulator sends to $\langle \mathcal{F}_{\mathrm{SFE}}^{f} \rangle$ a default input and delivers to the honest party, which results in a payoff of $-\gamma_{\$}$. (This is the worst outcome for $\mathcal{A}$.) Hence, with overwhelming probability, $\mathcal{A}$ makes $\mathcal{F}_{\mathrm{SFE}}^{f',\perp}$ output the shares. Now, if $\hat{i} = 1$ (i.e., the corrupted party gets the value first), then $\mathcal{A}$ can get the input and abort; this makes p_2 output a value based on a default input of p_1. The simulator $\mathcal{S}$ can account for this either by provoking the privacy event and inputting the default value, or by inputting the intended value, obtaining the output, and using the correctness event to adapt p_2's output. Hence, the payoff in this case will be $\min\{\gamma_{\mathsf{p}},\gamma_{\mathsf{c}}\}-\gamma_{\mathsf{c}}$. Otherwise ($\hat{i} = 2$), the adversary can only either deliver the authenticated share ($\mathcal{S}$ will provide the correct input) or not deliver it ($\mathcal{S}$ will provide the default input), either case leads to a payoff of $-\gamma_{\$}$. Because $\hat{i}$ is uniformly chosen, the payoff of the adversary is $\frac{1}{2}(\min\{\gamma_{\mathsf{p}},\gamma_{\mathsf{c}}\}-\gamma_{\mathsf{c}})-\frac{\gamma_{\$}}{2}$ (where the negligible quantity μ comes from the fact that there might be a negligible error in the simulation of $\mathcal{S}_{\mathcal{A}}$). This concludes the proof. □

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 $(\mathcal{F}_{\mathrm{SFE}}^{f}, \langle \mathcal{F}_{\mathrm{SFE}}^{f} \rangle, v^{\vec{\gamma}})$. Roughly, we first observe that in any such protocol there must be a round (not necessarily fixed; for each of the parties p_i) in which p_i "learns the output of the evaluation." An adversary corrupting one of the parties at random has probability 1/2 of corrupting the party that receives the output first; in that case the adversary learns the output and can abort the computation, forcing the simulator to provoke at least one of the events E_{p} or E_{c}. While we state and prove the optimality statements for the particular swap function $f_{\mathsf{swp}}(x_1, x_2) = (x_2, x_1)$, the results carry over to a large class of functions (essentially those where $1/p$-security [GK10] is impossible).

Theorem 11. *Let f_{swp} be the swap function described above, and for the attack-model $\mathcal{M} = (\mathcal{F}_{\mathrm{SFE}}^{f_{\mathsf{swp}}}, \langle \mathcal{F}_{\mathrm{SFE}}^{f_{\mathsf{swp}}} \rangle, v^{\vec{\gamma}})$, let $\mathcal{A}$ be the adversary described above. For every protocol Π which securely implements functionality $\langle \mathcal{F}_{\mathrm{SFE}}^{f_{\mathsf{swp}}} \rangle$, it holds that $\hat{U}^{\Pi,\langle \mathcal{F}_{\mathrm{SFE}}^{f_{\mathsf{swp}}} \rangle} \overset{\mathrm{negl}}{\geq} \frac{\min\{\gamma_{\mathsf{p}},\gamma_{\mathsf{c}}\}}{2} - \gamma_{\$}$.*

Proof. Toward proving the above lower bound we first show the following intermediate result (Claim 1), which will prove useful later on, and from which the lower bound for a generic adversarial strategy will be derived. We remark that Claim 1 is similar to the main theorem in [Cle86, Section 2.2]. We consider two specific adversarial strategies $\mathcal{A}_1$ and $\mathcal{A}_2$, which are valid against any protocol for implementing $\mathcal{F}_{\mathrm{SFE}}^{f_{\mathsf{swp}}}$. In strategy $\mathcal{A}_1$, the adversary (statically) corrupts p_1, and proceeds as follows: In each round ℓ, receive all the messages from p_2. Check whether p_1 *holds his actual output* ($\mathcal{A}_1$ generates a copy of p_1, simulates to this copy that p_2 aborted the protocol, obtains the output of p_1 and checks whether the output of p_1 is the default output—this strategy works since the environment is universally quantified and we can assume that it tells the honest party's input to the adversary); if so, record the output and abort the execution before sending p_1's ℓ-round message(s).[17] Otherwise, let p_1 correctly execute its instructions for round ℓ. The strategy $\mathcal{A}_2$ is defined analogously with exchanged roles for p_1 and p_2. In the following, for $i \in \{1, 2\}$ we let $p_{\bar{i}}$ denote the player p_{3-i}.

[17] This attack is possible because the adversary is rushing.

Claim 1. *Let f_{swp} be the swap function described above, $\mathcal{A}_1$ and $\mathcal{A}_2$ be the strategies defined above, and $\vec{\gamma} > 0$. For every protocol Π which securely implements functionality $\langle \mathcal{F}_{\text{SFE}}^{f_{\mathsf{swp}}} \rangle$ the following inequality holds:*

$$\hat{U}^{\Pi,\langle \mathcal{F}_{\text{SFE}}^{f_{\mathsf{swp}}} \rangle \vec{\gamma}}(\mathcal{A}_1) + \hat{U}^{\Pi,\langle \mathcal{F}_{\text{SFE}}^{f_{\mathsf{swp}}} \rangle, \vec{\gamma}}(\mathcal{A}_2) \overset{\text{negl}}{\geq} \min\{\gamma_{\mathsf{p}}, \gamma_{\mathsf{c}}\} - 2\gamma_{\$}.$$

Proof. For $i \in \{1, 2\}$ we consider the environment $\mathcal{Z}_i$ that is executed together with $\mathcal{A}_i$. The environment $\mathcal{Z}_i$ will initially choose a fixed value $x_{\bar{i}}$, which it provides as an input at the interface of $p_{\bar{i}}$ (and also communicates to $\mathcal{A}_i$). Subsequently, $\mathcal{Z}_i$ admits the execution between $p_{\bar{i}}$ and $\mathcal{A}_i$ until it is either finished or $\mathcal{A}_i$ aborts.

For compactness, we introduce the following two events in the protocol execution: We denote by L the event that the adversary aborts at a round where the honest party holds the actual output, and by $\bar{L}$ the event that the adversary aborts at a round where the honest party does not hold the actual input (i.e., if the corrupt party aborts, the honest party outputs some value other than x_i). Observe that, in case of $\bar{L}$, with overwhelming probability the simulator needs to send to the functionality a "corrupt" message, which will result in payoff $\min\{\gamma_{\mathsf{p}}, \gamma_{\mathsf{c}}\} - \gamma_{\$}$; indeed, the environment will choose the honest party's input with high entropy such that the event $\bar{L}$ will force the simulator to allow for the payoff. On the other hand, in case of L, the simulator must (with overwhelming probability) allow p_2 to obtain the output from $\langle \mathcal{F}_{\text{SFE}}^{f} \rangle$, resulting in payoff $-\gamma_{\$}$. Hence, except with negligible probability, whenever the adversary provokes the event L or $\bar{L}$, it obtains a payoff of $-\gamma_{\$}$ and $\min\{\gamma_{\mathsf{p}}, \gamma_{\mathsf{c}}\} - \gamma_{\$}$, respectively. Therefore, the payoff of these adversaries is (at least) $(-\gamma_{\$}) \Pr[L] + (\min\{\gamma_{\mathsf{p}}, \gamma_{\mathsf{c}}\} - \gamma_{\$}) \Pr[\bar{L}] - \mu''$, where μ'' is a negligible function (corresponding to the difference in the payoff that is created due to the simulation error of the optimal simulator).

To complete the proof, we compute the probability of each of the events L and $\bar{L}$ for $\mathcal{A}_1$ and $\mathcal{A}_2$. One important observation for both strategies $\mathcal{A}_1$ and $\mathcal{A}_2$, the adversary instructs the corrupted party to behave honestly until the round when he holds the actual output, hence all messages in the protocol execution have exactly the same distribution as in an honest execution until that round. For each party p_i, the protocol implicitly defines the rounds in which the honest, hence also honestly behaving, p_i holds the actual output (i.e., the test sketched above returns the output; note that, as the adversary learns the inputs from the environment, it can detect when this occurs). In such an execution, let R_i denote the first round where p_i holds the actual output. There are two cases: (i) $R_1 = R_2$ and (ii) $R_1 \neq R_2$. In case (i), both $\mathcal{A}_1$ and $\mathcal{A}_2$ provoke the event $\bar{L}$. In case (ii), if $R_1 < R_2$, then $\mathcal{A}_1$ always provokes the event $\bar{L}$, while for $\mathcal{A}_2$, with some probability (denoted as $q_{\bar{L}}$), the honest party does not hold the actual output when the $\mathcal{A}_2$ aborts, and with probability $1 - q_{\bar{L}}$ it does.[18] (Of course, the analogous arguments with switched roles hold for $R_1 > R_2$.

For the particular adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$, the considered values R_1 and R_2 are indeed relevant, since the adversaries both use the honest protocol machine as a "black box" until it starts holding the output. The probability of $\bar{L}$ for $\mathcal{A}_1$ is $\Pr[R_1 = R_2] + \Pr[R_1 < R_2] \cdot (1 - q_L)$, and the overall probability of L is $\Pr[R_1 < R_2] \cdot q_L + \Pr[R_1 < R_2]$, the probabilities for $\mathcal{A}_2$ are analogous. Hence,

[18] The reason is that we don't exclude protocols in which the output of a party which has been locked in some round gets unlocked in a future round.

we obtain

$$\begin{aligned}
&\hat{U}^{\Pi,\langle\mathcal{F}_{\text{SFE}}^{f_{\mathsf{swp}}}\rangle,\vec{\gamma}}(\mathcal{A}_1) + \hat{U}^{\Pi,\langle\mathcal{F}_{\text{SFE}}^{f_{\mathsf{swp}}}\rangle,\vec{\gamma}}(\mathcal{A}_2) \\
&\geq \Pr^{\mathcal{A}_1}[L]\cdot(-\gamma_{\$}) + \Pr^{\mathcal{A}_1}[\bar{L}]\cdot(\min\{\gamma_{\mathsf{p}},\gamma_{\mathsf{c}}\}-\gamma_{\$}) + \Pr^{\mathcal{A}_2}[L]\cdot(-\gamma_{\$}) + \Pr^{\mathcal{A}_2}[\bar{L}]\cdot(\min\{\gamma_{\mathsf{p}},\gamma_{\mathsf{c}}\}-\gamma_{\$}) \\
&\geq (2\cdot\Pr[R_1=R_2] + (\Pr[R_1<R_2]+\Pr[R_1>R_2])\cdot(1+q_{\bar{L}}))\cdot(\min\{\gamma_{\mathsf{p}},\gamma_{\mathsf{c}}\}-\gamma_{\$}) \\
&\qquad + ((\Pr[R_1<R_2]+\Pr[R_1>R_2])\cdot(1-q_{\bar{L}}))\cdot(-\gamma_{\$}) \\
&\geq (\Pr[R_1=R_2]+\Pr[R_1<R_2]+\Pr[R_1>R_2])\cdot(\min\{\gamma_{\mathsf{p}},\gamma_{\mathsf{c}}\}-\gamma_{\$}) \\
&\qquad + (\Pr[R_1=R_2]+\Pr[R_1<R_2]+\Pr[R_1>R_2])\cdot(-\gamma_{\$}) \\
&\geq (\min\{\gamma_{\mathsf{p}},\gamma_{\mathsf{c}}\}-\gamma_{\$}) + (-\gamma_{\$}) - \mu,
\end{aligned}$$

which was exactly the statement we wanted to prove. □

The lower bound in the above claim does not provide a bound for a *single* adversary (it is merely a statement that "either $\mathcal{A}_1$ or $\mathcal{A}_2$ must be good"). In the following, we prove a lower bound on the payoff of a fixed adversarial strategy, which we call $\mathcal{A}_{\text{gen}}$. This strategy is the random "mixture" of the two strategies $\mathcal{A}_1$ and $\mathcal{A}_2$ described above: The adversary corrupts one party chosen at random, checks (in each round) whether the protocol would compute the correct output on abort, and stops the execution as soon as it obtains the output. The lower bound is proven as follows: Since the adversary $\mathcal{A}_{\text{gen}}$ chooses one of the strategies $\mathcal{A}_1$ or $\mathcal{A}_2$ uniformly at random, it obtains payoff the average of the payoffs of $\mathcal{A}_1$ and $\mathcal{A}_2$. Hence, using Claim 1, we obtain

$$\hat{U}^{\Pi,\langle\mathcal{F}_{\text{SFE}}^{f_{\mathsf{swp}}}\rangle,\vec{\gamma}}(\mathcal{A}) = \frac{1}{2}\cdot\hat{U}^{\Pi,\langle\mathcal{F}_{\text{SFE}}^{f_{\mathsf{swp}}}\rangle,\vec{\gamma}}(\mathcal{A}_1) + \frac{1}{2}\cdot\hat{U}^{\Pi,\langle\mathcal{F}_{\text{SFE}}^{f_{\mathsf{swp}}}\rangle,\vec{\gamma}}(\mathcal{A}_2) \overset{\text{negl}}{\geq} \frac{1}{2}\cdot(\min\{\gamma_{\mathsf{p}},\gamma_{\mathsf{c}}\}-2\gamma_{\$}).$$

□

5.3 Feasibility for any (Given) Payoff Vector

We finally consider the case where the payoffs for breaking privacy and/or correctness may be arbitrarily large constants. As Theorem 8 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 [GK10, BLOO11]. We state the result informally below and refer to Appendix D for a formal statement and proof. The general 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 12 (informal)**.** *Let f be a (two-party or multi-party) function which can be evaluated by a $1/p$-secure protocol [GK10, BLOO11] and $\mathcal{M} = (\mathcal{F}_{\text{SFE}}^{f}, \langle\mathcal{F}_{\text{SFE}}^{f}\rangle, v^{\vec{\gamma}})$ 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

[ACH11] Gilad Asharov, Ran Canetti, and Carmit Hazay. Towards a game-theoretic view of secure computation. In Kenneth G. Paterson, editor, *Advances in Cryptology — EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 426–445. IACR, Springer, 2011.

[ADGH06] Ittai Abraham, Danny Dolev, Rica Gonen, and Joseph Y. Halpern. Distributed computing meets game theory: robust mechanisms for rational secret sharing and multiparty computation. In Eric Ruppert and Dahlia Malkhi, editors, *25th ACM PODC*, pages 53–62, Denver, Colorado, USA, July 23–26, 2006. ACM Press.

[ADH08] Ittai Abraham, Danny Dolev, and Joseph Y. Halpern. Lower bounds on implementing robust and resilient mediators. In Ran Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 302–319, San Francisco, CA, USA, March 19–21, 2008. Springer, Berlin, Germany.

[AKL+09] Joël Alwen, Jonathan Katz, Yehuda Lindell, Giuseppe Persiano, Abhi Shelat, and Ivan Visconti. Collusion-free multiparty computation in the mediated model. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 524–540, Santa Barbara, CA, USA, August 16–20, 2009. Springer, Berlin, Germany.

[AKMZ12] Joël Alwen, Jonathan Katz, Ueli Maurer, and Vassilis Zikas. Collusion-preserving computation. In *CRYPTO 2012*, LNCS, pages 124–143, Santa Barbara, CA, USA, August 2012. Springer, Berlin, Germany.

[AL07] Yonatan Aumann and Yehuda Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. In Salil Vadhan, editor, *Theory of Cryptography*, volume 4392 of *Lecture Notes in Computer Science*, pages 137–156. IACR, Springer, 2007.

[AL11] Gilad Asharov and Yehuda Lindell. Utility dependence in correct and fair rational secret sharing. *Journal of Cryptology*, 24(1):157–202, January 2011.

[ASV08] Joël Alwen, Abhi Shelat, and Ivan Visconti. Collusion-free protocols in the mediated model. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 497–514, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Berlin, Germany.

[BLOO11] Amos Beimel, Yehuda Lindell, Eran Omri, and Ilan Orlov. $1/p$-secure multiparty computation without honest majority and the best of both worlds. In Phillip Rogaway, editor, *Advances in Cryptology — CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 277–296. IACR, Springer, 2011.

[BPW03] Michael Backes, Birgit Pfitzmann, and Michael Waidner. Reactively secure signature schemes. In Colin Boyd and Wenbo Mao, editors, *ISC 2003*, volume 2851 of *LNCS*, pages 84–95, Bristol, UK, October 1–3, 2003. Springer, Berlin, Germany.

[BPW04] Michael Backes, Birgit Pfitzmann, and Michael Waidner. A general composition theorem for secure reactive systems. In Moni Naor, editor, *TCC 2004*, volume 2951 of

LNCS, pages 336–354, Cambridge, MA, USA, February 19–21, 2004. Springer, Berlin, Germany.

[BvDG+12] Kevin D. Bowers, Marten van Dijk, Robert Griffin, Ari Juels, Alina Oprea, Ronald L. Rivest, and Nikos Triandopoulos. Defending against the unknown enemy: Applying FlipIt to system security. In Jens Grossklags and Jean Walrand, editors, *Decision and Game Theory for Security*, volume 7638 of *Lecture Notes in Computer Science*, pages 248–263. Springer, 2012.

[Can00] Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13:143–202, April 2000.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145, Las Vegas, Nevada, USA, October 14–17, 2001. IEEE Computer Society Press.

[Can05] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, December 2005. A preliminary version of this work appeared in [Can01].

[CFGN96] Ran Canetti, Uriel Feige, Oded Goldreich, and Moni Naor. Adaptively secure multiparty computation. In *28th ACM STOC*, pages 639–648, Philadephia, Pennsylvania, USA, May 22–24, 1996. ACM Press.

[Cle86] Richard E. Cleve. Limits on the security of coin flips when half the processors are faulty. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing*, pages 364–369, Berkeley, 1986. ACM.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th ACM STOC*, pages 494–503, Montréal, Québec, Canada, May 19–21, 2002. ACM Press.

[CV12] Ran Canetti and Margarita Vald. Universally composable security with local adversaries. In *SCN 12*, LNCS, pages 281–301. Springer, Berlin, Germany, 2012.

[DHR00] Yevgeniy Dodis, Shai Halevi, and Tal Rabin. A cryptographic solution to a game theoretic problem. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 112–130, Santa Barbara, CA, USA, August 20–24, 2000. Springer, Berlin, Germany.

[DM00] Yevgeniy Dodis and Silvio Micali. Parallel reducibility for information-theoretically secure computation. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 74–92, Santa Barbara, CA, USA, August 20–24, 2000. Springer, Berlin, Germany.

[DS82] Danny Dolev and H. Raymond Strong. Polynomial algorithms for multiple processor agreement. In *Proceedings of the fourteenth Symposium on Foundations of Computer Science*, pages 401–407. ACM, 1982.

[FKN10] Georg Fuchsbauer, Jonathan Katz, and David Naccache. Efficient rational secret sharing in standard communication networks. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 419–436, Zurich, Switzerland, February 9–11, 2010. Springer, Berlin, Germany.

[GJKY13] Juan A. Garay, David Johnson, Aggelos Kiayias, and Moti Yung. Resource-based corruptions and the combinatorics of hidden diversity. In Robert D. Kleinberg, editor, *ITCS*, pages 415–428. ACM, 2013.

[GK06] S. Dov Gordon and Jonathan Katz. Rational secret sharing, revisited. In Roberto De Prisco and Moti Yung, editors, *SCN 06*, volume 4116 of *LNCS*, pages 229–241, Maiori, Italy, September 6–8, 2006. Springer, Berlin, Germany.

[GK10] Dov Gordon and Jonathan Katz. Partial fairness in secure two-party computation. In Henry Gilbert, editor, *Advances in Cryptology — EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 157–176. IACR, Springer, 2010.

[GK12] Adam Groce and Jonathan Katz. Fair computation with rational players. In *Advances in Cryptology—EUROCRYPT 2012*, July 2012.

[GKKZ11] Juan A. Garay, Jonathan Katz, Ranjit Kumaresan, and Hong-Sheng Zhou. Adaptively secure broadcast, revisited. In Cyril Gavoille and Pierre Fraigniaud, editors, *30th ACM PODC*, pages 179–186, San Jose, California, USA, June 6–8, 2011. ACM Press.

[GKTZ12] Adam Groce, Jonathan Katz, Aishwarya Thiruvengadam, and Vassilis Zikas. Byzantine agreement with a rational adversary. In *ICALP 2012, Part II*, LNCS, pages 561–572. Springer, Berlin, Germany, 2012.

[GLR10] Ronen Gradwohl, Noam Livne, and Alon Rosen. Sequential rationality in cryptographic protocols. In *51st FOCS*, pages 623–632. IEEE Computer Society Press, 2010.

[GMPY06] Juan A. Garay, Philip D. MacKenzie, Manoj Prabhakaran, and Ke Yang. Resource fairness and composability of cryptographic protocols. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 404–428, New York, NY, USA, March 4–7, 2006. Springer, Berlin, Germany.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game, or a completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229, New York City,, New York, USA, May 25–27, 1987. ACM Press.

[Hal08] Joseph Y. Halpern. Beyond nash equilibrium: solution concepts for the 21st century. In Rida A. Bazzi and Boaz Patt-Shamir, editors, *27th ACM PODC*, pages 1–10, Toronto, Ontario, Canada, August 18–21, 2008. ACM Press.

[HP10] Joseph Y. Halpern and Rafael Pass. Game theory with costly computation: Formulation and application to protocol security. In Andrew Chi-Chih Yao, editor, *ICS 2010*, pages 120–142. Tsinghua University Press, 2010.

[HT04] Joseph Y. Halpern and Vanessa Teague. Rational secret sharing and multiparty computation: Extended abstract. In László Babai, editor, *36th ACM STOC*, pages 623–632, Chicago, Illinois, USA, June 13–16, 2004. ACM Press.

[HZ10] Martin Hirt and Vassilis Zikas. Adaptively secure broadcast. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 466–485, French Riviera, May 30 – June 3, 2010. Springer, Berlin, Germany.

[IKLP06] Yuval Ishai, Eyal Kushilevitz, Yehuda Lindell, and Erez Petrank. On combining privacy with guaranteed output delivery in secure multiparty computation. In *Advances in Cryptology—CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 483–500. IACR, Springer, 2006.

[ILM08] Sergei Izmalkov, Matt Lepinski, and Silvio Micali. Verifiably secure devices. In Ran Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 273–301, San Francisco, CA, USA, March 19–21, 2008. Springer, Berlin, Germany.

[IML05] Sergei Izmalkov, Silvio Micali, and Matt Lepinski. Rational secure computation and ideal mechanism design. In *46th FOCS*, pages 585–595, Pittsburgh, PA, USA, October 23–25, 2005. IEEE Computer Society Press.

[Kat07] Jonathan Katz. On achieving the 'best of both worlds' in secure multiparty computation. In *ACM Symposium on Theory of Computing (STOC) 2007*, 2007.

[KKZZ11] Jonathan Katz, Aggelos Kiayias, Hong-Sheng Zhou, and Vassilis Zikas. Secure computation with corruptible setups. Public-Key Cryptography, Dagstuhl Reports, vol. 1, pp. 92, 2011.

[KMTZ13] Jonathan Katz, Ueli Maurer, Björn Tackmann, and Vassilis Zikas. Universally composable synchronous computation. In Amit Sahai, editor, *Theory of Cryptography — TCC 2013*, volume 7785 of *Lecture Notes in Computer Science*, pages 477–498. IACR, Springer, March 2013.

[KN08a] Gillat Kol and Moni Naor. Cryptography and game theory: Designing protocols for exchanging information. In Ran Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 320–339, San Francisco, CA, USA, March 19–21, 2008. Springer, Berlin, Germany.

[KN08b] Gillat Kol and Moni Naor. Games for exchanging information. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 423–432, Victoria, British Columbia, Canada, May 17–20, 2008. ACM Press.

[LMPS04] Matt Lepinski, Silvio Micali, Chris Peikert, and Abhi Shelat. Completely fair SFE and coalition-safe cheap talk. In *23rd ACM PODC*, pages 1–10, St. John's, Newfoundland, Canada, July 25–28, 2004. ACM Press.

[LMS05] Matt Lepinski, Silvio Micali, and Abhi Shelat. Collusion-free protocols. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 543–552, Baltimore, Maryland, USA, May 22–24, 2005. ACM Press.

[LP09] Yehuda Lindell and Benny Pinkas. A proof of security of Yao's protocol for two-party computation. *Journal of Cryptology*, 22(2):161–188, April 2009.

[LT06] Anna Lysyanskaya and Nikos Triandopoulos. Rationality and adversarial behavior in multi-party computation (extended abstract). In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 180–197, Santa Barbara, CA, USA, August 20–24, 2006. Springer, Berlin, Germany.

[MR11] Ueli Maurer and Renato Renner. Abstract cryptography. In Bernard Chazelle, editor, *The Second Symposium in Innovations in Computer Science, ICS 2011*, pages 1–21. Tsinghua University Press, January 2011.

[MS09] Silvio Micali and Abhi Shelat. Purely rational secret sharing (extended abstract). In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 54–71. Springer, Berlin, Germany, March 15–17, 2009.

[OPRV09] Shien Jin Ong, David C. Parkes, Alon Rosen, and Salil P. Vadhan. Fairness with an honest minority and a rational majority. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 36–53. Springer, Berlin, Germany, March 15–17, 2009.

[OR94] M.J. Osborne and A. Rubinstein. *A Course in Game Theory.* MIT Press, 1994.

A Related Literature

In this section we briefly survey some prior work on rational cryptography.

Rational secret sharing and SFE. Capturing incentives in security definitions has been the goal of a long line of exciting results on rational cryptography, initiated by the work of Dodis, Rabin, and Halevi [DHR00], which put forward the vision of bridging Game-theory and Cryptography. The seminal work of Halpern and Teague [HT04] considered rational secret sharing and (secure) function evaluation, where parties are rational with a "curious-then-exclusive" utility function: everyone prefers primarily learning the output and secondarily, if he learns it, having as few other parties learning it as possible. The solution concept suggested in [HT04] for such a setting is the notion of *Nash equilibrium surviving iterated deletion of weakly dominated strategies*. Refinements of this solution concept which are more robust (e.g., take coalitions into consideration) and/or more suitable for modeling security in a computational world were later on presented [GK06, ADGH06, KN08b, KN08a, Hal08, ADH08, FKN10], along with protocols implementing them. This fully rational model was also extended in various ways: from considering, in addition to rational, adversarial (non-rational) parties [LT06] or honest parties [OPRV09], to incorporating cost of computation [HP10].

Implementing the mediator in mediated games (cryptographic cheap talk). In a separate line of research, initiated by Dodis et al. [DHR00], the problem of implementing the mediator in mediated games by a cryptographic protocol was considered. In a series of papers, Micali et al [LMPS04, LMS05, IML05, ILM08] provided robust rational security definitions for the problem and implementations using physical communication primitives such as ballot-boxes and physical envelopes. Following up, Micali and shelat looked at the problem of rational secret sharing under physical communication assumptions [MS09]. Solutions in a more computational setting, where parties and resources are implemented as interactive Turing machines and an untrusted mediator is assumed, were also given [ASV08, AKL$^+$09, AKMZ12, CV12]. Although they rely on strong communication primitives the solutions from [ILM08] and [AKL$^+$09, CV12] explicitly deal with the issue of protocol composition in their respective models.

Cryptographic properties in rational frameworks. Recently, Asharov and Lindell [AL11] initiated a translation of the security of rational secret sharing in terms of cryptographic properties such as correctness and fairness. Following up, Asharov, Canetti, and Hazay [ACH11] provided rational cryptography notions of fair, correct, and private two-party SFE (by defining appropriate utilities for the parties and corresponding protocols) and showed an equivalence between these notions and natural cryptographic counterparts. Their results on fairness were extended in [GK12].

In comparison to these works, our simulation-based approach to defining the utilities allows us to trivially deal with multi-party computation and to have a more fine-grained definition of the utility function. For example, traditional notions of fairness in the rational setting [ACH11, GK12] define the utilities of the parties based on whether or not they obtain considerable information on each other's output, e.g., the first bits or the whole output. However, it would be natural to consider a protocol unfair if some party obtains *noticeable*, i.e., not negligible, information about the output and the other does not. This is not necessarily a well-defined event in the (real-world) protocol execution, but if one looks at the ideal world, then it corresponds to the simulator obtaining the output from the ideal functionality. By using the simulation-based paradigm, we are able to score even such events.

Covert and rational adversaries. A different line of work incorporates several types of incentives into standard cryptographic definitions. Aumann and Lindell [AL07] demonstrated how to take advantage of the adversary's "fear" of getting caught cheating to build more efficient protocols. Their model can be trivially captured in our framework by assigning a negative payoff to the event of (identifiable) abort. In a recent work closer in spirit to our own, Groce at al. [GKTZ12] investigated feasibility of Byzantine Agreement (BA) in a setting where the parties are rational but are split in two categories: the "selfish corrupt" parties that have some known utility function representing potential attacks to BA, and the "honest" parties who are guaranteed to follow the protocol. Their results show how to circumvent well-established cryptographic impossibility results by making plausible assumptions on the knowledge of the corrupt parties' preferences, thus confirming the advantages of incorporating incentives in the model when building practical protocols. However, their approach and security definitions are ad hoc and tailored to the specifics of the property-based definition of BA. In fact, our model can be tuned (by appropriately instantiating the utility function) to formalize the results of [GKTZ12] in a simulation-based manner, providing a further demonstration of the applicability of our model. The FlipIt model of [BvDG+12] also considers a game between one honest and one adversarial entity to model persistent threats; the entities are there called *defender* and *attacker*. In their model, however, the game itself is a perpetual interaction between those two entities, whereas in our case the game only consists of two steps of choosing strategies, and the interaction between those strategies occurs in a cryptographic model.

B Additional Details on Protocol Composition

This section includes supplementary material to Section 4.

B.1 Composition Theorems

The following discussion clarifies the dependency of the $\mathcal{M}$-maximizing adversary strategy to the protocol which is attacked and discusses the optimality of the "dummy" adversary (underlying the dummy lemma which follows).

Remark 2 (Protocol specific adversary and the "dummy" lemma). The notion of maximizing adversaries is specific to the protocol that is being attacked: An adversary that is optimal with respect to one particular protocol is not necessarily optimal with respect to some other protocol. Consequently, when talking about "the $\mathcal{M}$-maximizing adversary" we formally refer to the family $\{\mathcal{A}_\Pi\}_{\Pi\in\mathtt{ITM}}$ of ITMs such that $\mathcal{A}_\Pi$ is $\mathcal{M}$-maximizing with respect to protocol Π, according to Definition 2. However, the fact that we consider an environment that maximizes the adversary's expected payoff allows us to shift all the adversary's maximization effort to the environment. This leads to a type of "maximizing dummy adversary" result, which is useful when composing protocols.

The following lemma follows states that the dummy adversary performs at least as well as any other adversarial strategy against any protocol.

Lemma 13. *Let $(\mathcal{F}, \langle\mathcal{F}\rangle, v)$ be an attack model. Then for any $\mathcal{M}$-maximizing adversary $\mathcal{A}$:*

$$\hat{U}^{\Pi,\langle\mathcal{F}\rangle}(\mathcal{A}) \overset{\text{negl}}{\leq} \hat{U}^{\Pi,\langle\mathcal{F}\rangle}(\mathcal{D}),$$

where $\mathcal{D}$ denotes the dummy adversary that simply forwards messages to and from its environment.

The proof follows easily from the definition of the maximal utility. Indeed, because we consider the environment which maximizes the adversary's payoff, we can shift all the maximization effort to $\mathcal{Z}$.

Theorem 6 (Utility-preserving subroutine replacement)**.** *Let $\mathcal{M} = (\mathcal{F}, \langle\mathcal{F}\rangle, v)$ be an attack model. 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*

$$\hat{U}^{\Pi^{\Psi},\langle\mathcal{F}\rangle}(\mathcal{A}) \overset{\text{negl}}{\leq} \hat{U}^{\Pi^{\mathcal{H}},\langle\mathcal{F}\rangle}(\hat{\mathcal{A}})$$

for any $\mathcal{M}$-maximizing adversaries $\mathcal{A}$ and $\hat{\mathcal{A}}$, where Π^{Ψ} denotes the protocol where all calls to $\mathcal{H}$ are replaced by invocations of the sub-protocol Ψ and $\hat{\mathcal{A}}$ is the corresponding hybrid-model adversary guaranteed by the composition theorem. In particular for the "dummy" adversary $\mathcal{D}$ who simply forwards messages to and from its environment,

$$\hat{U}^{\Pi^{\Psi},\langle\mathcal{F}\rangle}(\mathcal{D}) \overset{\text{negl}}{\leq} \hat{U}^{\Pi^{\mathcal{H}},\langle\mathcal{F}\rangle}(\mathcal{D}).$$

Proof. The composition theorems in the considered frameworks provide us with the guarantee that for every $\mathcal{A}$ attacking Π^{ψ} there exists an adversary $\hat{\mathcal{A}}$ attacking $\Pi^{\mathcal{H}}$, and the security of Π means that there also exists a simulator $\hat{\mathcal{S}}$ such that

$$\text{EXEC}_{\Pi^{\psi},\mathcal{A},\mathcal{Z}} \overset{\text{negl}}{\approx} \text{EXEC}_{\Pi^{\mathcal{H}},\hat{\mathcal{A}},\mathcal{Z}} \overset{\text{negl}}{\approx} \text{EXEC}_{\langle\mathcal{F}\rangle,\hat{\mathcal{S}},\mathcal{Z}} \tag{1}$$

for any environment $\mathcal{Z}$. Now

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

But as by equation (1) any simulator $\bar{\mathcal{S}}$ for $\hat{\mathcal{A}}$ is also a good simulator for $\mathcal{A}$, this also holds for any simulator $\bar{\mathcal{S}}_{\varepsilon}$ with $\varepsilon > 0$ and $\sup_{\mathcal{Z}\in\texttt{ITM}}\{U_I^{\Pi^{\mathcal{H}},\langle\mathcal{F}\rangle}(\hat{\mathcal{S}}_{\varepsilon},\mathcal{Z})\} < \hat{U}^{\Pi^{\mathcal{H}},\langle\mathcal{F}\rangle}(\hat{\mathcal{A}}) + \varepsilon$ and we conclude that

$$\hat{U}^{\Pi^{\psi},\langle\mathcal{F}\rangle}(\mathcal{A}) = \sup_{\mathcal{Z}\in\texttt{ITM}}\inf_{\mathcal{S}\in\mathcal{C}_{\hat{\mathcal{A}}}}\{U_I^{\Pi^{\mathcal{H}},\langle\mathcal{F}\rangle}(\mathcal{S},\mathcal{Z})\} \leq \sup_{\mathcal{Z}\in\texttt{ITM}}\{U_I^{\Pi^{\mathcal{H}},\langle\mathcal{F}\rangle}(\bar{\mathcal{S}}_{\varepsilon},\mathcal{Z})\} < \hat{U}^{\Pi^{\mathcal{H}},\langle\mathcal{F}\rangle}(\hat{\mathcal{A}}) + \varepsilon$$

and the theorem follows. □

The following corollary show that such a replacement does not affect the stability of the solution in the corresponding attack game $\mathcal{G}_{\mathcal{M}}$. The proof follows in a straightforward manner from Theorem 4 by applying the above subroutine replacement theorem (Theorem 6).

Corollary 14. *Let $\mathcal{M} = (\mathcal{F}, \langle\mathcal{F}\rangle, v)$ be an attack model and $\mathcal{G}_{\mathcal{M}}$ be the corresponding attack game. Assume a strategy profile $(\Pi^R, \mathcal{A}(\cdot))$ is λ-subgame perfect in $\mathcal{G}_{\mathcal{M}}$, where Π^R is an R-hybrid protocol for some functionality R. Then for any protocol ρ which securely realizes R, $(\Pi^{\rho}, \mathcal{A}(\cdot))$ is also λ-subgame-perfect in $\mathcal{G}_{\mathcal{M}}$, where Π^{ρ} is the protocol that is derived by replacing in Π^R the call to R by invocation of ρ.*[19]

[19]Recall that $\mathcal{A}$ is here a mapping of protocols to adversary strategies.

Theorem 6 shows that the composition of the underlying cryptographic model extends rationally designed protocols: the subroutine replacement operation can be applied for protocols that (fully, e.g. UC-)implement a certain assumed functionality. If the underlying protocol is not (fully, e.g. UC-)secure, but only secure in the sense that we can upper-bound the utility of the attacker, we obtain (of course) only a weaker composition statement.

Theorem 15. *Let $\mathcal{M}_1 = (\mathcal{F}, \langle\mathcal{F}\rangle, v_1)$ and $\mathcal{M}_2 = (\mathcal{H}, \langle\mathcal{H}\rangle, v_2)$ be attack models, and let the payoff functions v_1 and v_2 be defined via event vectors $\vec{E}^*$ (for $\langle\mathcal{F}\rangle$) and $\vec{E}^{**}$ (for $\langle\mathcal{H}\rangle$), where the events in $\vec{E}^{**}$ are real-world decidable, together with the score vectors $\vec{\gamma}^*$ and $\vec{\gamma}^{**}$, respectively, such that every difference in behavior between $\langle\mathcal{F}\rangle$ and $\mathcal{F}$ (with respect to the same random tapes) is captured by some E_i^*. Let Π_1 be a protocol implementing $\langle\mathcal{F}\rangle$, and let Π_2 be an $\langle\mathcal{F}\rangle$-hybrid protocol implementing $\langle\mathcal{H}\rangle$. Assume that $\vec{\gamma}^* \geq 0$. Then,*

$$\hat{U}^{\Pi_2^{\Pi_1}} \overset{\text{negl}}{\leq} \frac{\hat{U}^{\Pi_1}}{\check{\gamma}_i^*} \cdot \hat{\gamma}_i^{**} + \min\left\{\hat{U}^{\Pi_2^{\mathcal{F}}}, \left(1 - \frac{\hat{U}^{\Pi_1}}{\check{\gamma}_i^*}\right) \cdot \hat{\gamma}_i^{**}\right\},$$

where $\check{\gamma}^* := \min_{1\leq i\leq m} \gamma_i^*$ and $\hat{\gamma}^* := \max_{1\leq i\leq m} \gamma_i^*$.

Intuitively, whenever the underlying functionality fails, no guarantees can be inferred for the composed protocol—and we have to assign the maximum utility to the adversary. Furthermore, even for the cases in which the underlying protocol indeed implements the given functionality (i.e., none of the "weakened" events occur; we infer a lower bound on this from the security statement for Π_1), we have to assume that these cases correspond to the ones where Π_2 shows the worst performance.

Proof. The inequality with respect to the second term of the minimization is trivial. To argue for the first term, we—intuitively—have to grant full Π_2-advantage to the cases whenever a weakness of $\langle\mathcal{F}\rangle$ is provoked, and we still have to assign "the worst part" of the Π_2-advantage to the case where nothing happens. The following arguments say that if the composed protocol $\Pi_2^{\Pi_1}$ performs even worse, then we can use the respective adversary (and environment) to construct a distinguisher for the protocol Π_2. Assume, toward a contradiction, that

$$\hat{U}^{\Pi_2^{\Pi_1}} > \hat{U}^{\Pi_2^{\mathcal{F}}} + \hat{\gamma}^{**} \cdot \frac{\hat{U}^{\Pi_1}}{\check{\gamma}_i^*} + 1/q$$

for some polynomial q. For any $\varepsilon, \varepsilon' > 0$, we construct distinguishers $\tilde{\mathcal{Z}}_{i,\varepsilon,\varepsilon'}$ with $1 \leq i \leq m$ for $\Pi_2^{\langle\mathcal{F}\rangle}$ and $\langle\mathcal{H}\rangle$ with $\langle\mathcal{S}_{2,\varepsilon'}\rangle$ as follows: $\tilde{\mathcal{Z}}_{i,\varepsilon,\varepsilon'}$ executes the environment $\mathcal{Z}_{\varepsilon,\varepsilon'}$ for the "generic" simulator $\mathcal{S}_{\varepsilon'}$ built from $\mathcal{S}_{1,\varepsilon'}$ and $\langle\mathcal{S}_{2,\varepsilon'}\rangle$ as well as $\mathcal{S}_{1,\varepsilon'}$, forwarding the parties' communication and the communication between $\mathcal{S}_{1,\varepsilon'}$ and the adversarial "interface" of the connected setting (either real or ideal) More explicitly, the simulator $\mathcal{S}_{1,\varepsilon'} \in \mathcal{C}_{\mathcal{A}}$ is chosen such that

$$\sup_{\mathcal{Z}\in\mathtt{ITM}} U_I^{\Pi_1}(\mathcal{S}_{1,\varepsilon'}, \mathcal{Z}) < \hat{U}^{\Pi_1} + \varepsilon',$$

and the analogous condition holds for $\langle\mathcal{S}_{2,\varepsilon'}\rangle$. Likewise, the condition for $\mathcal{Z}_{\varepsilon,\varepsilon'}$ is that $U_I^{\Pi_1}(\mathcal{S}_{1,\varepsilon'}, \mathcal{Z}_{\varepsilon,\varepsilon'}) > \sup_{\mathcal{Z}\in\mathtt{ITM}} U_I^{\Pi_1}(\mathcal{S}_{1,\varepsilon'}, \mathcal{Z}) - \varepsilon$. $\tilde{\mathcal{Z}}_{i,\varepsilon,\varepsilon'}$ then outputs 1 if the event E_i^{**} occurs—here

we assume that these events are real-world decidable. Overall, we obtain

$$\begin{aligned}\hat{U}^{\Pi_1}_{\Pi_2} &= \sup_{\bar{\mathcal{Z}}\in\mathtt{ITM}} \inf_{\bar{\mathcal{S}}\in\mathcal{C}_{\mathcal{A}}} \sum_{1\le i\le m} \gamma_i^{**}\cdot \Pr^{\langle\mathcal{H}\rangle,\bar{\mathcal{S}},\bar{\mathcal{Z}}}\left[E_i^{**}\right] \\ &\le \sup_{\bar{\mathcal{Z}}\in\mathtt{ITM}} \sum_{1\le i\le m} \gamma_i^{**}\cdot \Pr^{\langle\mathcal{H}\rangle,\mathcal{S}_{\varepsilon'},\bar{\mathcal{Z}}}\left[E_i^{**}\right] = \sum_{1\le i\le m} \gamma_i^{**}\cdot \Pr^{\langle\mathcal{H}\rangle,\mathcal{S}_{\varepsilon'},\mathcal{Z}_{\varepsilon,\varepsilon'}}\left[E_i^{**}\right] + \varepsilon \\ &= \sum_{1\le i\le m} \gamma_i^{**}\cdot \Pr^{\langle\mathcal{H}\rangle,\langle\mathcal{S}_{2,\varepsilon'}\rangle,\tilde{\mathcal{Z}}_{i,\varepsilon,\varepsilon'}}\left[E_i^{**}\right] + \varepsilon,\end{aligned}$$

by the way we chose environment $\mathcal{Z}_{\varepsilon,\varepsilon'}$ and simulator $\mathcal{S}_{\varepsilon'}$. Similarly,

$$\begin{aligned}\hat{U}^{\Pi_1}_{\Pi_2^{\mathcal{F}}} + \hat{\gamma}^{**}\cdot\frac{\hat{U}^{\Pi_1}}{\check{\gamma}_i^*} &\ge \sup_{\bar{\mathcal{Z}}\in\mathtt{ITM}} \sum_{1\le i\le m} \gamma_i^{**}\cdot\left(\Pr^{\langle\mathcal{H}\rangle,\mathcal{S}_{2,\varepsilon'},\bar{\mathcal{Z}}}\left[E_i^{**}\right]-\varepsilon'\right) + \hat{\gamma}^{**}\cdot\left(\Pr^{\Pi_2^{\mathcal{F}},\mathcal{S}_{1,\varepsilon'},\mathcal{Z}}\left[\vec{E}^*\right]-\varepsilon'\right) \\ &\overset{\text{negl}}{\ge} \sum_{1\le i\le m} \gamma_i^{**}\cdot\left(\Pr^{\Pi_2^{\langle\mathcal{F}\rangle},\mathcal{S}_{1,\varepsilon'},\mathcal{Z}_{\varepsilon,\varepsilon'}}\left[E_i^{**}\wedge\neg\vec{E}^*\right]-\varepsilon'\right) \\ &\quad + \hat{\gamma}^{**}\cdot\left(\Pr^{\Pi_2^{\langle\mathcal{F}\rangle},\mathcal{S}_{1,\varepsilon'},\mathcal{Z}_{\varepsilon,\varepsilon'}}\left[\vec{E}^{**}\wedge\vec{E}^*\right]-\varepsilon'\right) \\ &\ge \sum_{1\le i\le m} \gamma_i^{**}\cdot\left(\Pr^{\Pi_2^{\langle\mathcal{F}\rangle},\mathcal{S}_{1,\varepsilon'},\mathcal{Z}_{\varepsilon,\varepsilon'}}\left[E_i^{**}\right]-\varepsilon'\right) = \sum_{1\le i\le m} \gamma_i^{**}\cdot\left(\Pr^{\Pi_2^{\langle\mathcal{F}\rangle},\tilde{\mathcal{Z}}_{i,\varepsilon,\varepsilon'}}\left[E_i^{**}\right]-\varepsilon'\right).\end{aligned}$$

If there is a noticeable gap between the two sums, then at least one of the $\tilde{\mathcal{Z}}_{i,\varepsilon,\varepsilon'}$ must be good. For $\varepsilon,\varepsilon'\to 0$, this proves the claim. □

B.2 (In)composability in Rational Frameworks

Existing works on rational secure function evaluation make no statement about sub-routine replacement. Although this does not imply that such a replacement is not possible, in this section we demonstrate that replacement of secure channel in any of the rational function evaluation protocols/mechanisms from [HT04, KN08b, KN08a, OPRV09, FKN10] by their natural cryptographic implementation destroys the equilibrium. In light of this observation, there is no known way of replacing the (arguably unrealistic) assumptions of secure channels in the above protocols by simpler assumption of a PKI and insecure channels.

A first obstacle in formalizing the above statement (which is also due to the distance between traditional rational and cryptographic models) is that it is not even clear how to model an insecure channel, as (with the exception of [LT06]) there is no adversary in existing rational frameworks. However, for our argument we do not need a formal specification. In the following, we will refer to any resource which allows two parties to exchange a message so that any of the other parties can learn it as a *non-private channel*. Our argument shows that by replacing in any of the protocols in [HT04, KN08b, KN08a, FKN10] the secure channels assumption by a protocol sending messages signed and encrypted over any non-private channel (assuming a PKI) makes the protocols in the above rational cryptography works instable.

The argument follows by the same backward induction argument as in [KN08a]: The above protocols first use a cryptographic SFE protocol for implementing some given functionality (the functionality is different in each work); subsequently, the parties engage in an infinite-rounds revealment-protocol for computing their output from their SFE outputs. Assume that every player has the

following strategy: In each round of the revealment protocol, he checks one cryptographic key for the corresponding sender (by using this key to decrypt and then verify the signature on the sign-crypted message which is transmitted through the non-private channel). Clearly, after sufficiently many rounds (bounded by nK where n is the number of players and K is the size of the key-space, this player will have checked all the keys and thereby will be able to learn all the SFE (inputs and) outputs. Hence, in round nK every player is better off quitting and using the corresponding key for learning the output. This makes round $nK-1$ of the revealment protocol the last round of the whole protocol, and players have an incentive to deviate (quit) for the same reason. This process, known as backwards inductions, can be repeated to show that players will remain silent in rounds $nK-2, nK-3, \ldots, 1$. This proves the following:

Lemma 16 (informal)**.** *Let f be a multi-party NCC function and Π be any of the protocols for evaluating f from [HT04, KN08b, KN08a, FKN10] inducing the corresponding stable solution. Let Π' denote the protocol which results by replacing in Π all (bilateral) message transmissions by a protocol which signs-then-encrypts the message to be sent and then has the result transmitted over a non-private channel. Protocol Π' does not induce the corresponding solution.*

It should be noted that in the model suggested by Halpern and Pass [HP10], the above impossibility argument would not go through as they construct protocols with a finite number of rounds. In fact, [HP10, Theorem 4.2] proves an equivalence of (a version of) their security definition to standard cryptographic security. Although it might seem that such an equivalence directly provides a game-theoretic notion supporting subroutine replacement (hence, allowing secure channels to be replaced by insecure communication and a PKI), this is not the case as: (1) insecure communication is not captured in traditional game-theoretic frameworks (as there is no adversary), and (2) the above equivalence theorem holds only for subclasses of games or under "arguably unnatural" (cf. [HP10, Page 10]) assumptions about their complexity functions.[20] Note that a subroutine replacement theorem would use both directions of the equivalence, as one would need to translate the equilibrium into an equivalent cryptographic security statement, perform the replacement using a composition theorem, and, finally, translate the resulting protocol into an equilibrium statement. Hence, proving a composition theorem for the model of [HP10] is an open problem.

C Some Ideal Functionalities and Security Definitions

Ideal functionalities. The following functionality is essentially taken from [Can05]. To simplify notation, however, we generally drop the session IDs from the messages with the understanding that the messages implicitly contain the session ID and the functionalities (as ITMs) treat different sessions independently.

[20] In fact, the authors state that their equivalence results can be seen as an impossibility that considering only rational players does not facilitate protocol design, except if only subclasses of games are assumed.

Functionality $\mathcal{F}_{\text{SFE}}^{f}(\mathcal{P})$

$\mathcal{F}_{\text{SFE}}^{f}$ proceeds as follows, given a function $f : (\{0,1\}^* \cup \{\bot\})^n \times R \to (\{0,1\}^*)^n$ and a player set $\mathcal{P}$. Initialize the variables $x_1, \ldots, x_n, y_1, \ldots, y_n$ to a default value $\bot$.

- Upon receiving input $(\texttt{input}, v)$ from some party p_i with $i \in \mathcal{P}$, set $x_i := v$ and send a message $(\texttt{input}, i)$ to the adversary.
- Upon receiving input $(\texttt{output})$ from some party p_i with $i \in \mathcal{P}$, do:
  - If x_j has been set for all $j \in \mathcal{H}$, and $y_1, \ldots, y_n$ have not yet been set, then choose $r \xleftarrow{R} R$ and set $(y_1, \ldots, y_n) := f(x_1, \ldots, x_n, r)$.
  - Output y_i to p_i.

Our protocols assume the existence of a broadcast channel, which is formalized by the functionality $\mathcal{F}_{\text{BC}}$ below. In particular, if no party is corrupted, the adversary does not obtain the transmitted messages. This property is important for our protocols to be secure.

Functionality $\mathcal{F}_{\text{BC}}(\mathcal{P})$

The broadcast functionality $\mathcal{F}_{\text{BC}}$ is parametrized by a player set $\mathcal{P}$.

- Upon input x_s from $p_s \in \mathcal{P}$, $\mathcal{F}_{\text{BC}}$ sends x_s to every $p \in \mathcal{P}$. (If $\mathcal{F}_{\text{BC}}$ is considered a UC functionality, the output is given in a delayed manner, cf. [Can05].)

Definitions from the literature. Our protocols make use of (standard) commitment and signature schemes, which we define below.

Definition 17. A *commitment scheme* is a pair of two (efficient) algorithms $\texttt{commit} : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ and $\texttt{open} : \{0,1\}^* \times (\{0,1\}^*)^2 \to \{0,1\}$ such that $\texttt{open}(\texttt{commit}(m,r),(m,r)) = 1$ for any $m, r \in \{0,1\}^*$. We will often use the notation $\mathsf{com} \leftarrow \texttt{commit}(m,r)$ and $\mathsf{dec} \leftarrow (m,r)$.

Definition 18. A *signature scheme* is a triple $(\texttt{keygen}, \texttt{sign}, \texttt{verify})$ of (efficient) algorithms $\texttt{keygen} : \emptyset \to \{0,1\}^*$, $\texttt{sign} : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$, and $\texttt{verify} : (\{0,1\}^*)^3 \to \{0,1\}$, such that with $(\mathsf{pk}, \mathsf{sk}) \leftarrow \texttt{keygen}$, it holds that $\texttt{verify}(\mathsf{pk}, m, \texttt{sign}(\mathsf{sk}, m)) = 1$.

D Scoring Privacy, Correctness, and the Cost of Corruption

This section includes complementary material to Section 5.

A useful claim for the proof of Theorem 7

The following claim states what is achieved by invocation of the ideal functionality $\mathcal{F}_{\text{COM-SFE}}$ on page 16.

Claim 2 (Informal)**.** *Let $\mathcal{P} \subseteq [n]$, let $(\mathsf{pk}_1, \ldots, \mathsf{pk}_{|\mathcal{P}|})$ be a vector of public (verification) keys for the signature scheme. Consider the functionality $\mathcal{F} = \mathcal{F}_{\text{COM-SFE}}^{f}(\mathcal{P}, t, (\mathsf{pk}_1, \ldots, \mathsf{pk}_{|\mathcal{P}|}))$ invoked on the inputs $(\vec{S}_i, \mathsf{dec}_i)$—with*

$$\vec{S}_i = ((\mathsf{com}_{i,1}, \sigma_{1,1}^{(i)}, \ldots, \sigma_{1,|\mathcal{P}|}^{(i)}), \ldots, (\mathsf{com}_{i,1}, \sigma_{|\mathcal{P}|,1}^{(i)}, \ldots, \sigma_{|\mathcal{P}|,|\mathcal{P}|}^{(i)})),$$

and with[21] $\mathsf{dec}_i = (x_i, r_i)$ *and* $r_i \in \{0,1\}^*$*—from each* p_i*. Assume that*

(i) (at least) one $p \in \mathcal{P}$ *remains uncorrupted,*

(ii) for all honest p_i, p_j*:* $\mathtt{open}(\mathsf{com}_{i,i}, \mathsf{dec}_i) = 1$*,* $\mathtt{verify}(\mathsf{pk}_k, \mathsf{com}_{i,l}, \sigma^{(i)}_{l,k}) = 1$ *for all* $l, k \in [|\mathcal{P}|]$*, and* $\vec{S}_i = \vec{S}_j$*,*

(iii) for each honest $p_i \in \mathcal{P}$ *and each* $p_j \in \mathcal{P}$*: if* $\mathsf{com}_{j,i} \neq \mathsf{com}_{i,i}$*, then there is some* $p_k \in [|\mathcal{P}|]$ *such that* $\mathtt{verify}(\mathsf{pk}_k, \mathsf{com}_{j,i}, \sigma^{(j)}_{i,k}) = 0$*.*

Then, $\mathcal{F}$ *either outputs a random authenticated* t*-sharing of* $y = f(x_1, \ldots, x_n)$ *or it outputs* $(\mathtt{detect}, p_j)$*, for some corrupted* $p_j \in \mathcal{P}$*.*

Intuitively, the above claim means that to attack the protocol, an adversary must either break the binding property of the commitment scheme (to replace the input by a different value) or forge a signature on a fake commitment.

Proof (sketch). $\mathcal{F}_{\text{COM-SFE}}$ outputs $(\mathtt{detect}, p_j)$ only when p_j is corrupted: In step 1., honest parties will not be "detected" by assumption (ii). In step 2., no honest party will be "detected" because assumptions (ii) and (iii) imply that for $\mathsf{com}_{j,i} \neq \mathsf{com}_{i,i}$, p_j would have been "detected" in step 1. already, and step 3, follows exactly as step 1. If $\mathcal{F}_{\text{COM-SFE}}$ does not output $(\mathtt{detect}, \cdot)$, then all commitments have been opened successfully in step 3. and the output is as described by the definition of the functionality. □

The definition of t-privacy from [Kat07] For completeness we have included a definition of t-privacy from [Kat07]

Definition 19. Let $\mathcal{F}$ be an n-party randomized functionality, and Π be an n-party protocol. Then Π *t-privately computes f in the presence of malicious adversaries* if for any PPT adversary $\mathcal{A}$ there exists a PPT adversary $\mathcal{A}'$ such that for any $I \subset [n]$ with $|I| \leq t$:

$$\text{OUTPUT}_{\pi,\mathcal{A},I} \approx \text{OUTPUT}_{\mathcal{F},\mathcal{A}',I}.$$

Here, the random variables $\text{OUTPUT}_{\pi,\mathcal{A},I}$ and $\text{OUTPUT}_{\mathcal{F},\mathcal{A}',I}$ denote the output of the adversary in the real and ideal models, respectively, and $\approx$ denotes computational indistinguishability.

The following lemma shows that if a protocol is simulatable by a $\langle\mathcal{F}\rangle$-ideal simulator who never sends to the functionality the command $(\mathtt{inp}, \cdot)$, then it is private according to the notion in [Kat07]. We use the following notation: for a relaxed functionality $\langle\mathcal{F}\rangle$ we denote by $\langle\mathcal{F}\rangle^S$ the "strengthened" version of $\langle\mathcal{F}\rangle$ which ignores any $(\mathtt{inp}, \cdot)$ command. I.e., the difference between $\langle\mathcal{F}\rangle^S$ and $\mathcal{F}$ is that the former may only accept $(\mathtt{modify\ output}, \cdot)$ commands.

Lemma 20. *Let* $\langle\mathcal{F}\rangle^S$ *be as above. If a protocol* Π *securely realizes* $\langle\mathcal{F}\rangle^S$ *while tolerating* t *corruptions, then* Π *is* t*-private according to Definition 19.*

[21] We implicitly assume an encoding of group elements as bit strings.

Proof. Towards a contradiction, we assume that there is an input vector $\vec{x}$, an auxiliary string $z \in \{0,1\}^*$, and an adversary $\mathcal{A}$ such that for each $\mathcal{A}'$ there is a distinguisher $\mathcal{D}$ that tells $\text{OUTPUT}_{\pi,\mathcal{A},I}(k,\vec{x},z)$ and $\text{OUTPUT}_{\mathcal{F},\mathcal{A}',I}(k,\vec{x},z)$ apart (with noticeable probability for $k \to \infty$). Let $I \subset [n]$ with $|I| \leq t$ be the set of corrupted parties.

From $\vec{x}$, z, $\mathcal{A}$, and $\mathcal{D}$ we construct an environment for Π and $\langle \mathcal{F} \rangle^S$ as follows: the environment provides input x_i to party $i \in [n] \setminus I$ and instantiates $\mathcal{A}$ with input z. $\mathcal{Z}$ then executes $\mathcal{A}$ on the protocol: all messages obtained in the execution are given to $\mathcal{A}$, and all messages that $\mathcal{A}$ intends to deliver or inject in the execution are handled accordingly. As soon as $\mathcal{A}$ provides output, $\mathcal{Z}$ feeds this output into $\mathcal{D}$ and provides the output of $\mathcal{D}$ as its local output.

To conclude the proof, we need to show that any simulator $\mathcal{S}$ for $\langle \mathcal{F} \rangle^S$ can be used to build an adversary $\mathcal{A}'$ such that the output of $\mathcal{A}'$ given to $\mathcal{D}$ is the same in Katz's model an within $\mathcal{Z}$. In more detail, this $\mathcal{A}'$ is constructed from $\mathcal{A}$ and $\mathcal{S}$ (formally $\mathcal{A}'$ instantiates both $\mathcal{A}$ and $\mathcal{S}$, forwards the messages between them as well as $\mathcal{S}$'s interaction with the ideal functionality, and outputs the same as $\mathcal{A}$). Consequently, the input to $\mathcal{D}$ in Katz's model is distributed exactly as the input to the simulated $\mathcal{D}$ within $\mathcal{Z}$ in an ideal model execution with simulator $\mathcal{S}$, so the advantage of $\mathcal{D}$ directly translates into an advantage of $\mathcal{Z}$. □

Feasibility for any (Given) Payoff Vector (cont'd). In the following, we formalize the statements from Section 5.3. Roughly speaking, we show that for any function f for which there exist a $1/p$-secure protocol implementing f, we can construct an attack-payoff secure protocol with respect to an arbitrary score vector $\vec{\gamma}$.

More specifically, recall the notion of ε-security introduced by Gordon and Katz [GK10]: Informally, a protocol is ε-secure if it securely realizes the corresponding ideal functionality except with probability $1-\varepsilon$ [22]. Further, it was shown in [GK10] that $1/p$-security for an arbitrary polynomial p is possible for a large class of interesting two-party functions. These results were later extended by Beimel *et al.* [BLOO11] to the multi-party setting. Note that as these results hold for an arbitrary polynomial p, they also imply ε-security for an arbitrary small constant ε.

We now show that for any possible choice of γ_{p} and γ_{c}, if there exists an ε-secure protocol protocol for an arbitrarily small $\varepsilon > 0$ for computing a function f, then there exists a protocol which is attack-payoff secure in the attack-model $(\mathcal{F}_{\text{SFE}}^f, \langle \mathcal{F}_{\text{SFE}}^f \rangle, v^{\vec{\gamma}})$. In order for this to work we need some additional requirements that the simulator for Π should satisfy than what is required by the protocols in [GK10, BLOO11]:

Non-Triviality: When every party is honest, $\mathcal{S}$ can simulate Π in the $\mathcal{F}_{\text{SFE}}^f$-ideal world (i.e., without provoking any of the events E_{p} or E_{c}).

Taming: Let $t > 0$ be the number of corrupted parties at the end of the protocol execution. For any environment $\mathcal{Z}$, the expected ideal utility of $\mathcal{S}$, $U_I^{\Pi,\langle \mathcal{F}_{\text{SFE}}^f \rangle}(\mathcal{S},\mathcal{Z}) \overset{\text{negl}}{\leq} \varepsilon(\gamma_{\mathsf{p}} + \gamma_{\mathsf{c}}) - t\gamma_{\$}$.

We say that a protocol is *enhanced ε-secure* in the attack-model $\mathcal{M}$ if it is ε-secure with a simulator that satisfies the above properties. Intuitively, the goal of the taming property is to ensure that there exists a simulator in the ε-secure protocol which only uses his extra power (i.e., the relaxation commands) in the executions that would otherwise not be simulatable. As we argue below, the protocols in [GK10, BLOO11] are indeed enhanced ε-secure for the corresponding attack model.

[22] Technically, the notion introduced in [GK10] is called "$1/p$-security," where p is some polynomial of the security parameter; the notion of ε-security is trivially obtained from their definition by replacing p with the constant polynomial $p = \varepsilon^{-1}$.

Theorem 21. *Let f be a function and let $\mathcal{M} = (\mathcal{F}_{\text{SFE}}^f, \langle \mathcal{F}_{\text{SFE}}^f \rangle, v^{\vec{\gamma}})$ be an attack model where the elements of $\vec{\gamma}$, γ_{p}, γ_{c}, and $\gamma_{\$}$ are arbitrary (known) positive constants. Then if there exists an enhanced ε-secure protocol Π in $\mathcal{M}$ for arbitrarily small $\varepsilon > 0$, then there exists an attack-payoff secure protocol in $\mathcal{M}$.*

Proof. Let ε_0 be such that $\varepsilon_0(\gamma_{\mathsf{p}} + \gamma_{\mathsf{c}}) < \gamma_{\$}$. We show that a protocol Π which is enhanced ε_0-secure in $\mathcal{M}$ (guaranteed to exist by the theorem's premise) is also attack-payoff secure in $\mathcal{M} = (\mathcal{F}_{\text{SFE}}^f, \langle \mathcal{F}_{\text{SFE}}^f \rangle, v^{\vec{\gamma}})$. Indeed, the enhanced ε_0-security of Π guarantees that there exists a simulator $\mathcal{S}$ such that:

- If no party is corrupted then for any environment $\mathcal{Z}$ the ideal expected utility of the simulator $U^{\Pi,\langle \mathcal{F}_{\text{SFE}}^f \rangle}(\mathcal{Z}, \mathcal{S}) = 0$. This follows directly from the non-triviality property of enhanced ε-security.
- If $t > 0$ parties are corrupted during the protocol execution, then for every environment $\mathcal{Z}$, the ideal expected utility of $\mathcal{S}$ is $U_I^{\Pi,\langle \mathcal{F}_{\text{SFE}}^f \rangle}(\mathcal{S}, \mathcal{Z}) \overset{\text{negl}}{\leq} \varepsilon_0(\gamma_{\mathsf{p}} + \gamma_{\mathsf{c}}) - t\gamma_{\$} < 0$. Note that this property also implies that Π securely realizes $\langle \mathcal{F}_{\text{SFE}}^f \rangle$, as otherwise we would, by definition, have $U_I^{\Pi,\langle \mathcal{F}_{\text{SFE}}^f \rangle}(\mathcal{S}, \mathcal{Z}) = \infty$.

Because the maximal (over the set of all $\mathcal{Z}$'s) expected utility of any $\mathcal{S}$ is an upper bound on the adversary's utility, the best strategy of the adversary is not to corrupt any party, which proves that the protocol is attack-payoff secure. □

We conclude this section by arguing that the $1/p$-secure protocols from [GK10, BLOO11] are indeed enhanced $1/p$-secure in the corresponding attack model. (We defer a formal statement and proof to the full version of the paper.) The proof idea is as follows: These protocols start by generating many sharings of either "dummy" outputs or of the real output using a protocol which can be simulated without ever querying the functionality $\mathcal{F}_{\text{SFE}}^f$ for evaluating the corresponding function f. Furthermore, the protocols choose a round r^* in which the output should be actually revealed, and encode this round into the sharings sequence. Subsequently, these sharings are reconstructed in multiple sequential rounds.

The key observation is that because the sharings from the first phase are authenticated, the only way that the adversary might cheat in the reconstruction is by aborting prematurely. Further, the simulator samples the actual round r^* himself, and can therefore identify (by seeing whether or not the adversary aborts in this round) whether the simulation might fail, which would require him to invoke the events E_{c} and E_{p}. This means that the simulator only invokes these events with the probability that the simulation might fail, which for a ε-secure protocol is $\varepsilon \pm \lambda$, for some negligible λ. Hence, the adversary's utility is upper-bounded by $\varepsilon(\gamma_{\mathsf{p}} + \gamma_{\mathsf{c}}) - t\gamma_{\$} + \lambda$.