How Fair is Your Protocol? A Utility-based Approach to Protocol Optimality

Juan Garay
Yahoo Labs
garay@yahoo-inc.com

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

Björn Tackmann[*]
UC San Diego
btackmann@eng.ucsd.edu

Vassilis Zikas[†]
ETH Zurich
vzikas@inf.ethz.ch

ABSTRACT

Security of distributed cryptographic protocols usually requires *privacy* (inputs of the honest parties remain hidden), *correctness* (the adversary cannot improperly affect the outcome), and *fairness* (if the adversary learns the output, all honest parties do also). Cleve's seminal result (STOC '86) implies that satisfying these properties simultaneously is impossible in the presence of dishonest majorities, and led to several proposals for relaxed notions of fairness.

In this work we put forth a new approach for defining relaxed fairness guarantees that allows for a quantitative comparison between protocols with regard to the level of fairness they achieve. The basic idea is to use an appropriate utility function to express the preferences of an adversary who wants to violate fairness. We also show optimal protocols with respect to our notion, in both the two-party and multi-party settings.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and Protection*

General Terms

Security, Theory

1. INTRODUCTION

Two parties p_1 and p_2 wishing to sign a contract are considering the following two protocols, Π_1 and Π_2 (communication is done over secure channels):

[*]Research partly done at ETH Zurich and while visiting University of Maryland.

[†]Research partly done at University of Maryland and UCLA.

PODC'15, July 21–23, 2015, Donostia-San Sebastián, Spain.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-3617-8 /15/07 ...$15.00.
http://dx.doi.org/10.1145/2767386.2767431.

- In Π_1, p_1 and p_2 locally digitally sign the contract, compute commitments c_0 and c_1 on the signed versions, and exchange these commitments. Subsequently, p_1 opens its commitment to p_2, and then p_2 opens his commitment to p_1. If during any of the above steps p_i, $i \in \{1, 2\}$, observes that p_{3-i} sends him an inconsistent message, then he aborts.
- Π_2 starts off similarly to Π_1, except that to determine who opens his commitment first, the parties execute a coin tossing protocol [4]: p_1 and p_2 locally commit to random bits b_1 and b_2, exchange the commitments, and then in a single round they open them. For each p_i, if the opening of b_{3-i} is valid then p_i computes $b = b_1 \oplus b_2$; otherwise p_i aborts. The parties then use b to determine which party opens the committed signed contract first.

Which protocol should the parties use? Intuitively, and assuming a party is honest, the answer should be clear: Π_2, since the cheating capabilities of a corrupt party are reduced in comparison to Π_1. Indeed, the probability of a corrupted p_i forcing an unfair abort (i.e., receiving the contract signed by p_{3-i} while preventing p_{3-i} from also receiving it) in protocol Π_2 is roughly half of the probability in protocol Π_1. In other words, one would simply say that protocol Π_2 is "twice as fair as" protocol Π_1.

Yet, most existing cryptographic security definitions, including those specifically intended to model relaxed notions of fairness in an effort to circumvent Cleve's impossibility result [10], would simply say that both protocols are unfair and make no further statement about their relative fairness. For example, both protocols would be considered unfair with respect to *resource fairness* [16], which formalizes the intuition of the *gradual release* paradigm [4, 2, 11, 5, 23] in a simulation-based framework. Indeed, a resource-fair protocol should ensure that, upon abort, the amount of computation that the honest party needs for producing the output is comparable to the adversary's for the same task; this is clearly not the case for either of the protocols, as with probability at least one-half the adversary might learn the output (i.e., receive the signed contract) when it is infeasible for the other party to compute it. The same holds for "rational" definitions of fairness [1, 20], which require the protocol to be an equilibrium strategy with respect to a preference/utility function for rational agents. We stress that some of these definitional frameworks show that one *can* construct protocols that are fair with respect to the given framework; nevertheless, none of them provide a way to quantify the "amount" of fairness achieved by an arbitrary cryptographic protocol.

Motivated by the above observation, in this paper we put forth quantitative definitions of fairness for two-party and multi-party protocols. Our notions are based on the idea that we can use an appropriate utility function to express the preferences of an adversary who wants to break fairness. Our definitions allow for comparing protocols with respect to how fair they are, placing them in a partial order according to a relative-fairness relation. We then investigate the question of finding maximal elements in this partial order (which we refer to as *optimally fair* protocols) for the case of two-party and multi-party secure function evaluation (SFE). Importantly, our quantitative fairness and optimality approach is fully composable (cf. [8]) with respect to standard secure protocols, in the sense that we can replace a "hybrid" in a fair/optimal protocol with a protocol which securely implements it without affecting its fairness/optimality.

Our approach builds on machinery developed in the recently proposed *Rational Protocol Design* (RPD) framework, by Garay *et al.* [14]. In more detail, [14] describes how to design protocols which keep the utility of an attacker aiming at provoking certain security breaches as low as possible. At a high level, we use RPD as follows: first, we specify the class of utility functions that naturally capture an adversary attacking a protocol's fairness, and then we interpret the actual utility that the best attacker (i.e., the one maximizing its respective utility) obtains against a given protocol as a measure of the protocol's fairness. The more a protocol limits its best attacker with respect to our fairness-specific utility function, the fairer the protocol is. Going back to the Π_1 vs. Π_2 example at the beginning of the section, we can now readily use this utility function to formally express that protocol Π_2 is fairer than protocol Π_1, because Π_1 allows the adversary to *always* obtain maximum utility, whereas Π_2 reduces this utility by a factor of $1/2$.

Related work. There is a considerable amount of work on fairness, and on defining relaxed notions of fairness. After Cleve's impossibility result [10], perhaps the most notable line of work is on "gradual release" of information [4, 2, 11, 5, 23, 16]. More recently, Gordon and Katz [18] proposed the notion of $1/p$-*security*. Roughly, their definition guarantees that fairness holds except with probability $1/p$, for some specified polynomial p. One could adopt the parameter p as a measure of a protocol's fairness, although Gordon and Katz show some fundamental limits regarding what functions can be securely computed with regard to their definition. In our work we design protocols for evaluating *arbitrary* functions. At a definitional level, we observe that our definition always (except with negligible probability) guarantees privacy and correctness, which (as already pointed out by Gordon and Katz) is not the case for $1/p$-security; see Section 5. Interestingly, we also show that for an appropriate choice of the utility function, our utility-based fairness notion implies $1/p$-security for some p.

A different line of work tries to capture relaxed notions of fairness by assuming that protocol participants are rational agents with a fairness-related utility function [1, 20]. This approach is incomparable to ours, or to any other existing notion of fairness in the non-rational setting where the honest parties are *not* rational and follow the protocol as specified. In particular, the optimal protocols suggested here (and in other fairness notions in the non-rational setting) do not imply an equilibrium in the sense of [1, 20].[1] We stress also that the definitions from [1, 20] do not imply a comparative notion of fairness, as a protocol either induces an equilibrium or it does not.

Organization of the paper. The remainder of the paper is organized as follows. In Section 2 we describe notation and the very basics of the RPD framework [14] that are needed for understanding and evaluating our results. In Section 3 we define the utility function of attackers who aim to violate fairness, which enables the relative assessment of protocols as well as the notions of "optimal" fairness which we use in this work. In Section 4 we present optimally fair protocols for two-party and multi-party ($n > 2$ parties) secure function evaluation (SFE) (Sections 4.1 and 4.2, resp.) Our protocols are not only optimally fair but also optimal with respect to the number of *reconstruction rounds*—a measure formalized here which has been implicit in the fairness literature. Furthermore, for the case of multi-party SFE, we also provide an alternative (incomparable) notion of optimality that relates to how costly corruptions might be for the adversary.[2] Finally, in Section 5 we compare our utility-based fairness notion to $1/p$-security (aka "partial fairness") as developed by Gordon and Katz [18]. Detailed constructions, proofs, and other complementary material appear in the full version of this work [15].

2. PRELIMINARIES

We first establish some notational conventions. For an integer $n \in \mathbb{N}$, the set of positive numbers smaller or equal to n is $[n] := \{1, \ldots, n\}$. In the context of two-party protocols, we will always refer to the parties as p_1 and p_2, and for $i \in \{1, 2\}$ the symbol $\neg i$ refers to the value $3-i$ (so $p_{\neg i} \neq p_i$). Most statements in this paper are actually 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 $\mu \geq 1/p$. We further introduce the symbols $f \overset{\text{negl}}{\approx} g \;:\Leftrightarrow\; \exists$ negligible $\mu : |f - g| \leq \mu$, and $f \overset{\text{negl}}{\geq} g \;:\Leftrightarrow\; \exists$ negligible $\mu : f \geq g - \mu$, with $\overset{\text{negl}}{\leq}$ defined analogously.

For the model of protocol composition, we follow Canetti's adaptive simulation-based model for multi-party computation [6]. The protocol execution is formalized by collections of interactive Turing machines (ITMs); the set of all *efficient* ITMs is denoted by $\texttt{ITM}$. We generally denote our protocols by Π and our (ideal) functionalities (which are also referred to as the trusted party [6]) by $\mathcal{F}$ both with descriptive super- or subscripts, the adversary by $\mathcal{A}$, the simulator by $\mathcal{S}$, and the environment by $\mathcal{Z}$. The random variable ensemble $\{\text{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}(k, z)\}_{k\in\mathbb{N}, z\in\{0,1\}^*}$, which is more compactly often written as $\text{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$, describes the contents of $\mathcal{Z}$'s output tape after an execution with Π, $\mathcal{F}$, and $\mathcal{A}$, on auxiliary input $z \in \{0, 1\}^*$.

[1]We note in passing that our protocols do, in fact, imply an equilibrium, but in the attack "meta-game" defined in [14]. Interested readers are referred to [14] for more details.

[2]Secure multi-party computation with costly corruptions was first studied in [13].

Rational Protocol Design. Our results utilize the *Rational Protocol Design* (RPD) framework [14]. Here we review the basic elements that are needed to motivate and express our definitions and results; we refer to the framework paper [14] for further details. In RPD, security is defined via a two-party sequential zero-sum game with perfect information, called the *attack game*, between a protocol *designer* $\mathtt{D}$ and an *attacker* $\mathtt{A}$. The designer $\mathtt{D}$ plays first by specifying a protocol Π for the (honest) participants to run; subsequently, the attacker $\mathtt{A}$, who is informed about $\mathtt{D}$'s move (i.e., learns the protocol) plays by specifying a polynomial-time attack strategy $\mathcal{A}$ by which it may corrupt parties and try to subvert the execution of the protocol (uncorrupted parties follow Π as prescribed). Note that it suffices to define the utility $u_{\mathtt{A}}$ of the adversary as the game is zero-sum. (The utility $u_{\mathtt{D}}$ of the designer is then $-u_{\mathtt{A}}$.)

The utility definition relies on the simulation paradigm[3] in which a real-world execution of protocol Π in the presence of attack strategy (or adversary) $\mathcal{A}$ is compared to an ideal-world execution involving an ideal-world attack strategy (that is, a simulator $\mathcal{S}$) interacting with a functionality $\mathcal{F}$ which models the task at hand. Roughly speaking, the requirement is that the two worlds be indistinguishable to any environment $\mathcal{Z}$ which provides the inputs and obtains the outputs of all parties, and interacts arbitrarily with the adversary $\mathcal{A}$.

For defining the utilities in RPD, however, the real world is compared to an ideal world in which $\mathcal{S}$ gets to interact with a *relaxed* version of the functionality which, in addition to implementing the task as $\mathcal{F}$ would, also allows the simulator to perform the attacks we are interested in capturing. For example, an attack to the protocol's correctness is modeled by the functionality allowing the simulator to modify the outputs (even of honest parties). Given such a functionality, the utility of any given adversary is defined as the expected utility of the best simulator for this adversary, where the simulator's utility is defined based on which weaknesses of the ideal functionality the simulator is forced to exploit.

3. UTILITY-BASED FAIRNESS AND PROTOCOL OPTIMALITY

In this section, we make use of the RPD framework to introduce a natural fairness relation (partial order) to the space of efficient protocols. Specifically, we consider an instantiation of RPD with an attacker who obtains utility for violating fairness. The RPD machinery can be applied to most simulation-based security frameworks; however, for the sake of clarity we restrict ourselves to the technically simpler framework of Canetti [6] (allowing sequential and modular composition), which considers synchronous protocols with guaranteed termination. Our definitions can be extended to Universally Composable (UC) security [7] using the approach of Katz *et al.* [21] to model terminating synchronous computation in UC.

Now to our approach. We follow the three-step process described in [14] for specifying an adversary's utility, instantiating this process with parameters that capture a fairness-targeted attacker:

[3]In RPD the statements are formalized in Canetti's Universal Composition (UC) framework [7]; however, one could in principle use any other simulation-based model such as Canetti's MPC framework [6].

Step 1: Relaxing the ideal experiment to allow attacks on fairness. First, we relax the ideal world to allow the simulator to perform fairness-related attacks. In particular, we consider the experiment corresponding to standard ideal SFE with abort experiment [6, 17] with the difference that the simulator only receives the outputs of corrupted parties *if he asks for them* (we denote the corresponding trusted-party/functionality as $\mathcal{F}^{\perp}_{\text{SFE}}$). In a nutshell, $\mathcal{F}^{\perp}_{\text{SFE}}$ is similar to standard SFE but allows the simulator to ask for corrupted parties' outputs, and, subsequently, to send $\mathcal{F}^{\perp}_{\text{SFE}}$ a special (`abort`)-message even after having received these outputs (but before some honest parties receive the output). Upon receiving such an abort message, the functionality sets the output of every (honest) party to $\perp$. We refer to the above ideal world as the $\mathcal{F}^{\perp}_{\text{SFE}}$*-ideal world.* We point out that the functionality $\mathcal{F}^{\perp}_{\text{SFE}}$ is as usually parametrized by the actual function f to be evaluated; when we want to make this function f explicit we will write $\mathcal{F}^{f,\perp}_{\text{SFE}}$.

Step 2: Events and payoffs. Next, we specify a set of events in the experiment corresponding to the ideal evaluation of $\mathcal{F}^{\perp}_{\text{SFE}}$ which capture whether or not a fairness breach occurs, and assign to each such event a "payoff" value capturing the severity of provoking the event. The relevant questions to ask with respect to fairness are:

1. Does the adversary learn "noticeable" information about the output of the corrupted parties?
2. Do honest parties learn their output?

The events used to describe fairness correspond to the four possible combinations of answers to the above questions. In particular, we define the events indexed by a string $ij \in \{0,1\}^2$, where i (resp., j) equals 1 if the answer to the first (resp., second) question is yes and 0 otherwise. The events are then as follows:

E_{00}: The simulator does not ask functionality $\mathcal{F}^{\perp}_{\text{SFE}}$ for any of the corrupted parties' outputs and instructs it to abort before all honest parties receive their output. (Thus, neither the simulator nor the honest parties will receive their outputs.)

E_{01}: The simulator does not ask $\mathcal{F}^{\perp}_{\text{SFE}}$ for any of the corrupted parties' outputs and does not abort. (When the protocol terminates, then only the honest parties will receive the output. This event also accounts for cases where the adversary does not corrupt *any* party.)

E_{10}: The simulator asks $\mathcal{F}^{\perp}_{\text{SFE}}$ for some corrupted party's output and instructs it to abort before any honest party receives the output.

E_{11}: The simulator asks $\mathcal{F}^{\perp}_{\text{SFE}}$ for some corrupted party's output and does not abort. (When the protocol terminates, both the honest parties and the simulator will receive their outputs. This event also accounts for cases where the adversary corrupts *all* parties.)

We remark that our definition does not give any advantage to an adversary corrupting all parties. This is consistent with the intuitive notion of fairness, as when there is no honest party, the adversary has nobody to gain an unfair advantage over.

To each of the events E_{ij} we associate a real-valued *payoff* γ_{ij} which captures the adversary's utility when provoking this event. Thus, the adversary's payoff is specified by vector $\vec{\gamma} = (\gamma_{00}, \gamma_{01}, \gamma_{10}, \gamma_{11})$, corresponding to events $\vec{E} = (E_{00}, E_{01}, E_{10}, E_{11})$.

Finally, we define the expected payoff of a given simulator $\mathcal{S}$ (for an environment $\mathcal{Z}$) to be[4]:

$$U_I^{\mathcal{F}^{\perp}_{\text{SFE}},\vec{\gamma}}(\mathcal{S},\mathcal{Z}) \quad := \quad \sum_{i,j\in\{0,1\}} \gamma_{ij} \Pr[E_{ij}]. \qquad (1)$$

Step 3: Defining the attacker's utility. Given the expected payoff $U_I^{\mathcal{F}^{\perp}_{\text{SFE}},\vec{\gamma}}(\mathcal{S},\mathcal{Z})$, the utility $u_{\mathtt{A}}(\Pi,\mathcal{A})$ for a pair $(\Pi,\mathcal{A})$ is defined following the methodology in [14] as the expected payoff of the *best* simulator[5] that simulates $\mathcal{A}$ in the $\mathcal{F}^{\perp}_{\text{SFE}}$-ideal world in presence of the least favorable environment, i.e., the one that is *most* favorable to the attacker. To make the payoff vector $\vec{\gamma}$ explicit, we sometimes denote the above utility as $\hat{U}^{\Pi,\mathcal{F}^{\perp}_{\text{SFE}},\vec{\gamma}}(\mathcal{A})$ and refer to it as the *payoff of strategy* $\mathcal{A}$ (for attacking Π).

More formally, for a protocol Π, denote by $\mathtt{SIM}_{\mathcal{A}}$ the class of simulators for $\mathcal{A}$, i.e, $\mathtt{SIM}_{\mathcal{A}} = \{\mathcal{S} \in \mathtt{ITM} \mid \forall \mathcal{Z} : \text{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}} \approx \text{EXEC}_{\mathcal{F}^{\perp}_{\text{SFE}},\mathcal{S},\mathcal{Z}}\}$. The payoff of strategy $\mathcal{A}$ (for attacking Π) is then defined as:

$$u_{\mathtt{A}}(\Pi,\mathcal{A}) := \hat{U}^{\Pi,\mathcal{F}^{\perp}_{\text{SFE}},\vec{\gamma}}(\mathcal{A}) := \sup_{\mathcal{Z}\in\mathtt{ITM}} \inf_{\mathcal{S}\in\mathtt{SIM}_{\mathcal{A}}} \{U_I^{\mathcal{F}^{\perp}_{\text{SFE}},\vec{\gamma}}(\mathcal{S},\mathcal{Z})\}. \qquad (2)$$

To complete our formalization, we now describe a natural relation among the values in $\vec{\gamma}$ which is both intuitive and consistent with existing approaches to fairness, and which we will assume to hold for the remainder of the paper. Specifically, we will consider attackers whose least preferred event is that the honest parties receive their output while the attacker does not, i.e., we assume that $\gamma_{01} = \min_{\gamma\in\vec{\gamma}}\{\gamma\}$. Furthermore, we will assume that the attacker's favorite choice is that he receives the output and the honest parties do not, i.e., $\gamma_{10} = \max_{ij\in\{0,1\}^2}\{\gamma_{ij}\}$. Lastly, we point out that for an arbitrary payoff vector $\vec{\gamma}$, one can assume without loss of generality that any one of its values equals zero, and, therefore, we can set $\gamma_{01} = 0$. This can be seen immediately by setting $\gamma'_{ij} = \gamma_{ij} - \gamma_{01}$. We denote the set of all payoff vectors adhering to the above restrictions by $\Gamma_{\text{fair}} \subseteq \mathbb{R}^4$. Summarizing, our fairness-specific payoff ("preference") vector $\vec{\gamma}$ satisfies

$$0 = \gamma_{01} \leq \min\{\gamma_{00},\gamma_{11}\} \quad \text{and} \quad \max\{\gamma_{00},\gamma_{11}\} < \gamma_{10}.$$

Optimally fair protocols. We are now ready to define our partial order relation for protocols with respect to fairness. Informally, a protocol Π will be *at least as fair* as another protocol Π' if the utility of the best adversary $\mathcal{A}$ attacking Π (i.e, the adversary which maximizes $u_{\mathtt{A}}(\Pi,\mathcal{A})$) is no larger than the utility of the best adversary attacking Π' (except for some negligible quantity). Our notion of fairness is with respect to the above natural class Γ_{fair}; for conciseness, we will abbreviate and say that a protocol is "$\vec{\gamma}$-fair," for $\vec{\gamma} \in \Gamma_{\text{fair}}$. Formally:

DEFINITION 1. *Let* Π *and* Π' *be protocols, and* $\vec{\gamma} \in \Gamma_{fair}$ *be a preference vector. We say that* Π *is* at least as fair as Π' with respect to $\vec{\gamma}$ *(i.e., it is at least as* $\vec{\gamma}$*-fair), denoted* $\Pi \overset{\vec{\gamma}}{\succeq} \Pi'$, *if*

$$\sup_{\mathcal{A}\in\mathtt{ITM}} u_{\mathtt{A}}(\Pi,\mathcal{A}) \quad \overset{\text{negl}}{\leq} \quad \sup_{\mathcal{A}\in\mathtt{ITM}} u_{\mathtt{A}}(\Pi',\mathcal{A}). \qquad (3)$$

We will refer to a protocol which is a maximal element according to the above fairness relation as an *optimally fair* protocol.

DEFINITION 2. *Let* $\vec{\gamma} \in \Gamma_{fair}$*. A protocol* Π *is* optimally $\vec{\gamma}$-fair *if it is at least as* $\vec{\gamma}$*-fair as any other protocol* Π'.

Definition 2 presents our most basic notion of utility-based fairness. With foresight, one issue that arises with this definition in the multi-party setting is that it is not sensitive to the number of corrupted parties, so when an adversary is able to corrupt parties for free, he is better off corrupting all $n-1$ parties. In Section 4.2 we also present an alternative notion of fairness suitable for situations where the number of corrupted parties does matter, as, for example, when corrupting parties carries some cost (cf. [13]).

4. UTILITY-BASED FAIR SFE

In this section we investigate the question of finding optimally $\vec{\gamma}$-fair protocols for secure two-party and multi-party function evaluation, for any $\vec{\gamma} \in \Gamma_{\text{fair}}$. (Recall that Γ_{fair} is a class of natural preference vectors for fairness—cf. Section 3.) In addition, for the case of multi-party protocols, we also suggest an alternative, incomparable notion of fairness that is sensitive to the number of corrupted parties and is therefore relevant when this number is an issue. As we describe our protocols in the model of [6], the protocols are synchronous and parties communicate with each other via bilateral secure channels. We point out that the protocols described here are secure against adaptive adversaries [9].

4.1 The Two-Party Setting

In this section we present an optimally $\vec{\gamma}$-fair protocol, $\Pi^{\text{Opt}}_{\text{2SFE}}$, for computing any given function. Its optimality is established by proving a general upper bound on the utility $u_{\mathtt{A}}(\Pi,\mathcal{A})$ of an adversary $\mathcal{A}$ attacking it, and then presenting a specific function f and an adversary who attacks the protocol $\Pi^{\text{Opt}}_{\text{2SFE}}$ for computing f that obtains a utility which matches the above upper bound.

4.1.1 The Protocol (Upper Bound)

Our protocol makes use of a well-known cryptographic primitive called *authenticated secret sharing.* An authenticated additive (two-out-of-two) secret sharing scheme is an additive sharing scheme augmented with a message authentication code (MAC) to ensure verifiability. (See the Appendix for a concrete instantiation.) Protocol $\Pi^{\text{Opt}}_{\text{2SFE}}$ works in two phases as follows; f denotes the function to be computed:

1. In the first phase, $\Pi^{\text{Opt}}_{\text{2SFE}}$ invokes an adaptively secure unfair SFE protocol (e.g., the protocol in [17]—call it Π_{GMW})[6] to compute the following function f': f' takes as input the inputs of the parties to f, and outputs an authenticated sharing of the output of f along with an

[4]Refer to [14, Section 2] for the rationale behind this formulation.

[5]The best simulator is taken to be the one that minimizes his payoff [14].

[6]Note that assuming ideally secure channels, the protocol Π_{GMW} is adaptively secure [9].

index $i \in_{\mathrm{R}} \{1,2\}$ chosen uniformly at random. In case the protocol aborts, the honest party takes a default value as the input of the corrupted party and locally computes the function f.

2. In the second phase, if Π_{GMW} did not abort, the protocol continues in two more rounds. In the first round, the output (sharing) is reconstructed towards p_i, and in the second round it is reconstructed towards $p_{\neg i}$. In case $p_{\neg i}$ does not send a valid share to p_i in the first round, p_i again takes a default value as the input of the (corrupted) party $p_{\neg i}$ and computes the function f locally (the second round is then omitted).

As we show in the following theorem, the adversary's payoff in the above protocol is upper-bounded by $\frac{\gamma_{10}+\gamma_{11}}{2}$. The intuition behind the proof is as follows: If the adversary corrupts the party that first receives the output, then he can provoke his most preferred event E_{10} by aborting before sending his last message. However, because this party is chosen at random, this happens only with probability 1/2; with the remaining 1/2 probability the honest party receives the output first, in which case the best choice for the adversary is to allow the protocol to terminate and provoke the event E_{11}.

Without loss of generality, we assume that the function f has a single global output; indeed, a protocol that can compute any such function f can be easily extended to compute functions with multiple, potentially private outputs by using standard techniques, e.g., see [22].

Theorem 3. *Let $\vec{\gamma} \in \Gamma_{fair}$ and $\mathcal{A}$ be an adversary. Then $u_{\mathsf{A}}(\Pi^{Opt}_{2SFE}, \mathcal{A}) \overset{\text{negl}}{\leq} \frac{\gamma_{10}+\gamma_{11}}{2}$.*

Proof (sketch). We prove the statement for $\Pi^{\mathrm{Opt}}_{\mathrm{2SFE}}$ in the $\mathcal{F}^{f',\perp}_{\mathrm{SFE}}$-hybrid model. The theorem then follows by applying the RPD composition theorem [14, Theorem 5], which extends to the case where the framework is instantiated with the model of Canetti [6].

First we remark that if the adversary corrupts both parties or no party, then the theorem follows directly from the definition of the payoff and the properties of Γ_{fair}, as in these cases the payoff the adversary obtains equals γ_{11} or γ_{01}, respectively. 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). To complete the proof it suffices to provide a simulator $\mathcal{S}_{\mathcal{A}}$ for any adversary $\mathcal{A}$, such that $\mathcal{S}_{\mathcal{A}}$ has expected payoff at most $\frac{\gamma_{10}+\gamma_{11}}{2}$. Such a (black-box straight-line) simulator $\mathcal{S}_{\mathcal{A}}$ for an adversary $\mathcal{A}$ works as follows.

To emulate the output of $\mathcal{F}^{f',\perp}_{\mathrm{SFE}}$, $\mathcal{S}_{\mathcal{A}}$ does the following (recall that the output consists of a share for p_1 and a uniformly chosen index $i \in \{1,2\}$): $\mathcal{S}_{\mathcal{A}}$ randomly picks an index $\hat{i} \in_R \{1,2\}$ along with the element $\hat{s}_1, \hat{k}_1$ and a random MAC-tag $\hat{t}_2$; $\mathcal{S}_{\mathcal{A}}$ hands to the adversary the (simulated) share $(\hat{s}_1, \hat{t}_2)$, the key $\hat{k}_1$, and the index $\hat{i}$. Subsequently, $\mathcal{S}_{\mathcal{A}}$ simulates the opening stage of $\Pi^{\mathrm{Opt}}_{\mathrm{2SFE}}$:

- If $\hat{i} = 1$, then $\mathcal{S}_{\mathcal{A}}$ sends $\hat{x}_1$ (which it obtained because of the $\mathcal{F}^{f',\perp}_{\mathrm{SFE}}$-hybrid model) to $\mathcal{F}^{f,\perp}_{\mathrm{SFE}}$ and asks for the output[7]; 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 follows: set $t'_1 := \mathtt{tag}(y, \hat{k}_1)$ and $t'_2 := \mathtt{tag}(y, \hat{k}_2)$ for a uniformly chosen k_2. Set $\hat{s}_2 := (y, t'_1, t'_2) - s_1$ and $\hat{t}_1 := \mathtt{tag}(\hat{s}_2, \hat{k}_1)$. Send $(\hat{s}_2, \hat{t}_1)$ 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 $(\hat{s}_1, \hat{t}_2)$, then it sends $\mathtt{abort}$ to $\mathcal{F}^{f,\perp}_{\mathrm{SFE}}$, before the honest party is allowed to receive its output.
- If $\hat{i} = 0$ then $\mathcal{S}_{\mathcal{A}}$ receives from $\mathcal{A}$ p_1's share. If $\mathcal{S}_{\mathcal{A}}$ receives a share other than $(\hat{s}_1, \hat{t}_2)$, then it sends a default value to $\mathcal{F}^{f,\perp}_{\mathrm{SFE}}$ (as p_1's input). Otherwise, it asks $\mathcal{F}^{f,\perp}_{\mathrm{SFE}}$ 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 $\mathcal{A}$.

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

We now argue that for any adversary $\mathcal{A}$ corrupting p_1, the payoff of $\mathcal{S}_{\mathcal{A}}$ is (at most) $\frac{\gamma_{10}+\gamma_{11}}{2} + \mu$ for some negligible function μ. If $\mathcal{A}$ makes the evaluation of the function f' in the first phase to abort, the simulator sends $\mathcal{F}^{f,\perp}_{\mathrm{SFE}}$ a default input and delivers to the honest party, which provokes the event E_{01}; hence the payoff of this adversary will be $\gamma_{01} < \frac{\gamma_{10}+\gamma_{11}}{2}$. Otherwise, i.e., if $\mathcal{A}$ allows the parties to receive their f'-outputs/shares in the first phase, then we consider the following two cases: (1) if $\hat{i} = 1$ (i.e., the corrupted party gets the value first), then $\mathcal{A}$ can always provoke his most preferred event by receiving the output in the first round of the opening stage and then aborting, which will make $\mathcal{S}_{\mathcal{A}}$ provoke the event E_{10}. (2) if $\hat{i} = 2$ the adversary's choices are to provoke the events E_{01} or E_{11}, out of which his more preferred one is E_{11}. Because $\hat{i}$ is uniformly chosen, each of the cases (1) and (2) occurs with probability 1/2; hence, the payoff of the adversary is $\frac{\gamma_{10}+\gamma_{11}}{2} + \mu$ (where the negligible quantity μ comes from the fact that there might be a negligible error in the simulation of $\mathcal{S}_{\mathcal{A}}$). Therefore, in any case the utility of the attacker choosing adversary $\mathcal{A}$ is $u_{\mathsf{A}}(\Pi^{\mathrm{Opt}}_{\mathrm{2SFE}}, \mathcal{A}) \overset{\text{negl}}{\leq} \frac{\gamma_{10}+\gamma_{11}}{2}$ which concludes the proof. □

4.1.2 Optimality of the Protocol (Lower Bound)

Next, we show that the above bound is tight for protocols that evaluate *arbitrary* functions. We remark that, for specific classes of functions—such as those with polynomial-size range or domain—one is able to obtain fairer protocols. For example, it is easy to verify that for functions which admit $1/p$-secure solutions [18] for an arbitrary polynomial p, we can reduce the upper bound in Theorem 3 to $\frac{\gamma_{10}+\gamma_{11}}{p}$. (Refer to Section 5 for a detailed comparison of our notion to $1/p$-security). Thus, an interesting future direction is to find optimally fair solutions for computing primitives such as random selection [19] and set intersection [12] which could then be used in higher-level constructions.

The general result shows that there are functions for which $\frac{\gamma_{10}+\gamma_{11}}{2}$ is also a lower bound on the adversary's utility for *any* protocol, independently of the number of rounds. Here we prove this for the particular "swap" function $f_{\mathsf{swp}}(x_1, x_2) = (x_2, x_1)$; the result carries over to a large class of functions (essentially those where $1/p$-security is proved impossible in [18]).

At a high level, the proof goes as follows: First, we observe that in any protocol execution there must be one round

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

(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 other party to not receive it, which results in a payoff γ_{10}. With the remaining $1/2$ probability, the adversary does not corrupt the correct party. In this case, finishing the protocol and obtaining payoff γ_{11} is the best strategy.[8]

We first show an intermediate result, where we consider two specific adversarial strategies $\mathcal{A}_1$ and $\mathcal{A}_2$, which are valid against any protocol. 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 functionality is secure with abort); if so, record the output and abort the execution before sending p_1's ℓ-round message(s).[9] Otherwise, let p_1 correctly execute its instructions for round ℓ. The strategy $\mathcal{A}_2$ is defined analogously with roles for p_1 and p_2 exchanged.

Lemma 4. *Let f_{swp} be the swap function, $\mathcal{A}_1$ and $\mathcal{A}_2$ be the strategies defined above, and $\vec{\gamma} \in \Gamma_{fair}$. Every protocol Π which securely realizes functionality $\mathcal{F}_{\text{SFE}}^{f_{\mathsf{swp}},\perp}$ satisfies:*

$$u_{\mathtt{A}}(\Pi, \mathcal{A}_1) + u_{\mathtt{A}}(\Pi, \mathcal{A}_2) \quad \overset{\text{negl}}{\geq} \quad \gamma_{10} + \gamma_{11}.$$

Proof (sketch). For $i \in \{1,2\}$ we consider the environment $\mathcal{Z}_i$ that is executed together with $\mathcal{A}_i$ corrupting p_i. The environment $\mathcal{Z}_i$ will choose a fixed value $x_{\neg i}$, which it provides as an input to $p_{\neg i}$.

For compactness, we introduce the following two events in the protocol execution: We denote by L the event that the adversary aborts in a round where the honest party holds the actual output (in other words the honest party's output is "locked"), and by $\bar{L}$ the event that the adversary aborts at a round where the honest party does not hold the actual output (i.e., if the corrupt party aborts, the honest party outputs some value other than $f(x_1, x_2)$). Observe that, in cases corresponding to the real-world event $\bar{L}$, with overwhelming probability the simulator needs to send to the functionality the "`abort`" messages, provoking γ_{10}; indeed, because Π is secure with abort, in that case $p_{\neg i}$ needs to output $\perp$ with overwhelming probability (otherwise, there is a noticeable probability that he will output a wrong value, which contradicts security with abort of Π). On the other hand, in cases corresponding to L, the simulator must (with overwhelming probability) allow $p_{\neg i}$ to obtain the output from $\mathcal{F}_{\text{SFE}}^{f,\perp}$, provoking the event γ_{11}. Hence, except with negligible error, the adversary obtains γ_{11} and γ_{10} for provoking the events L and $\bar{L}$, respectively. Therefore, the payoff of these adversaries is (at least) $\gamma_{11}\Pr[L] + \gamma_{10}\Pr[\bar{L}] - \mu''$, 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 it 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 output of honest, hence also of honestly behaving, parties are "locked." 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.[10] (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, we obtain

$$\begin{aligned}
&u_{\mathtt{A}}(\Pi, \mathcal{A}_1) + u_{\mathtt{A}}(\Pi, \mathcal{A}_2) \\
&\geq \gamma_{11} \Pr^{\mathcal{A}_1}[L] + \gamma_{10} \Pr^{\mathcal{A}_1}[\bar{L}] + \gamma_{11} \Pr^{\mathcal{A}_2}[L] + \gamma_{10} \Pr^{\mathcal{A}_2}[\bar{L}] \\
&\geq \gamma_{10} \cdot (2 \cdot \Pr[R_1 = R_2] + (1 + q_{\bar{L}}) \cdot \Pr[R_1 \neq R_2]) \\
&\quad + \gamma_{11} \cdot (1 - q_{\bar{L}}) \cdot \Pr[R_1 \neq R_2] \\
&\geq \gamma_{10} \cdot (\Pr[R_1 = R_2] + \Pr[R_1 < R_2] + \Pr[R_1 > R_2]) \\
&\quad + \gamma_{11} \cdot (\Pr[R_1 = R_2] + \Pr[R_1 < R_2] + \Pr[R_1 > R_2]) \\
&\geq \gamma_{10} + \gamma_{11} - \mu,
\end{aligned}$$

which was exactly the statement we wanted to prove. □

Lemma 4 provides a bound involving two adversaries. (It can be viewed as a statement that "one of $\mathcal{A}_1$ and $\mathcal{A}_2$ must be good"). However, we can use it to prove our lower bound on the payoff by considering the single adversarial strategy, call it $\mathcal{A}_{\text{gen}}$, that is the "mix" 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. In the sequel, for a given function f we say that a protocol securely realizes the functionality $\mathcal{F}_{\text{SFE}}^{f,\perp}$ if it securely evaluates f in the $\mathcal{F}_{\text{SFE}}^{f,\perp}$-ideal world.

Theorem 5. *Let $\vec{\gamma} \in \Gamma_{fair}$, f_{swp} be the swap function. There exists an adversary $\mathcal{A}$ such that for every protocol Π which securely realizes functionality $\mathcal{F}_{\text{SFE}}^{f_{\mathsf{swp}},\perp}$, it holds that $u_{\mathtt{A}}(\Pi, \mathcal{A}) \quad \overset{\text{negl}}{\geq} \quad \frac{\gamma_{10}+\gamma_{11}}{2}$.*

Proof. Let $\mathcal{A}$ be the adversary $\mathcal{A}_{\text{gen}}$ described above. As adversary $\mathcal{A}_{\text{gen}}$ chooses one of the strategies $\mathcal{A}_1$ or $\mathcal{A}_2$ uniformly at random, it obtains the average of the utilities of $\mathcal{A}_1$ and $\mathcal{A}_2$. Indeed, using Lemma 4, we obtain

$$u_{\mathtt{A}}(\Pi, \mathcal{A}_{\text{gen}}) = \frac{1}{2} \cdot u_{\mathtt{A}}(\Pi, \mathcal{A}_2) + \frac{1}{2} \cdot u_{\mathtt{A}}(\Pi, \mathcal{A}_2) \overset{\text{negl}}{\geq} \frac{1}{2} \cdot (\gamma_{10} + \gamma_{11} - \mu),$$

which completes the proof. □

[8]The adversary could also obtain γ_{01} by aborting, but will not play this strategy as, by assumption, $\gamma_{01} \leq \min\{\gamma_{00}, \gamma_{11}\}$.

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

[10]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.

The above theorem establishes that $\Pi^{\text{Opt}}_{\text{2SFE}}$ is optimally γ-fair. We also remark that the protocol is optimal with respect to the number of reconstruction rounds. See Appendix 4.1.3 for details. Next, we consider multi-party SFE (i.e., $n > 2$).

4.1.3 *Round Complexity of the Reconstruction Phase*

Most, if not all, protocols in the literature designed to achieve a (relaxed) notion of fairness have a similar structure: They first invoke a general (unfair) SFE protocol for computing a sharing of the output, and then proceed to a reconstruction phase where they attempt to obtain the output by reconstructing this sharing. Since the first (unfair SFE) phase is common in all those protocols, the number of rounds of the reconstruction phase is a reasonable complexity measure for such protocols.

As we show below, protocol $\Pi^{\text{Opt}}_{\text{2SFE}}$ is not only optimally $\vec{\gamma}$-fair but is also optimal with respect to the number of reconstruction rounds, i.e., the number of rounds it invokes after the sharing of the output has been generated. To demonstrate this we first provide a formal definition of reconstruction rounds. Note that also the notion of reconstruction rounds is implicit in many works in the fairness literature, to our knowledge, a formal definition such as the one described here has not been provided elsewhere.

Intuitively, a protocol has ℓ reconstruction rounds if up to ℓ rounds before the end, the adversary has not gained any advantage in learning the output, but the next round is the one where the reconstruction starts. Formally,

Definition 6. *Let Π be an SFE protocol for evaluating the (multi-party) function $f : (\{0,1\}^*)^n \to (\{0,1\}^*)^n$ which terminates in m rounds. We say that Π has ℓ* reconstruction-rounds *if it implements the (fair) functionality $\mathcal{F}^{f}_{\text{SFE}}$ in the presence of any adversary who aborts in any of the rounds $1, \ldots, m-\ell$, but does not implement it if the adversary aborts in round $m-\ell+1$.*

Lemma 7. *Π^{Opt}_{2SFE} has two reconstruction rounds.*

Proof (sketch). The security of the protocol used in phase 1 of $\Pi^{\text{Opt}}_{\text{2SFE}}$ and the privacy of the secret sharing, ensures that the view of the adversary during this phase (including his output) can be perfectly simulated without ever involving the functionality. Thus if the adversary corrupting, say, p_1 (the case of a corrupted p_2 is symmetric) aborts during this phase, then p_2 can simply locally evaluate the function on his input and a default input by the adversary. To simulate this, the simulator will simply hand the default input to the fair functionality. However, as implied by the lower bound in Theorem 5, this is not the case if the adversary aborts in the first round of phase 2. $\square$

Lemma 8. *Assuming $\vec{\gamma} \in \Gamma_{fair}$, there exists no optimally $\vec{\gamma}$-fair protocol for computing the swap function f_{swp} (see Lemma 4) with a single reconstruction round.*

Proof (sketch). Assume towards contradiction that a protocol Π with a single reconstruction round exists. Clearly, before the last round the output should not be "locked" for neither of the parties. Indeed, if this is the case the adversary corrupting this party can, as in the proof of Lemma 4, force an unfair abort which cannot be simulated in the $\mathcal{F}^{f_{\mathsf{swp}}}_{\text{SFE}}$-ideal model. Now, in the (single) reconstruction round, a rushing adversary receives the message from the honest party but does not send anything, which can only be simulated by making the honest party abort. This adversary obtains maximum payoff, γ_{10} (except with negligible probability). Thus Π is less $\vec{\gamma}$-fair than $\Pi^{\text{Opt}}_{\text{2SFE}}$ and hence is not optimally $\vec{\gamma}$-fair. $\square$

4.2 The Multi-Party Setting

Throughout this section, we make the simplifying assumption that the attacker prefers learning the output over not learning it, i.e., $\gamma_{00} \leq \gamma_{11}$. Although this assumption is natural and standard in the rational fairness literature, it is *not* without loss of generality. It is, however, useful in proving multi-party fairness statements, as it allows us to compute the utility of the attacker for a protocol which is fully secure for $\mathcal{F}_{\text{SFE}}$, including fairness. Indeed, while such a protocol might still allow the attacker to abort and hence obtain utility γ_{00}, in this case the optimal utility is γ_{11} as the event E_{11} is the "best" event which $\mathcal{A}$ can provoke. Combined with the inequalities from Section 3, the entries in vector $\vec{\gamma}$ satisfy $0 = \gamma_{01} \leq \gamma_{00} \leq \gamma_{11} < \gamma_{10}$. We denote by $\Gamma^{+}_{\text{fair}} \subseteq \Gamma_{\text{fair}}$ the class of payoff vectors with the above restriction.

The intuition behind protocol $\Pi^{\text{Opt}}_{\text{2SFE}}$ can be extended to also work in the multi-party ($n > 2$) setting. The idea for the corresponding multi-party protocol $\Pi^{\text{Opt}}_{\text{nSFE}}$, which is described below in more detail, is as follows: In a first phase, $\Pi^{\text{Opt}}_{\text{nSFE}}$ computes the private output function $f'(x_1, \ldots, x_n) = (y_1, \ldots, y_n)$, where for some random $i^* \in [n]$, y_{i^*} equals the output of the function f we wish to compute, whereas for all $i \in [n] \setminus \{i^*\}$, $y_i = \bot$; in addition to y_i, every party p_i receives an authentication tag on y_i.[11] If this phase aborts then the protocol also aborts. In phase 2, all parties announce their output y_i (by broadcasting them). If a validly authenticated message $y \neq \bot$ is broadcast, then the parties adopt it; otherwise, they abort.

Functionality $\langle \mathcal{F}^{f,\bot}_{\text{PRIV-SFE}} \rangle$

1. Compute the function f on the given inputs and store the (public) output in variable y.
2. Chose $(\mathsf{sk}, \mathsf{vk}) \xleftarrow{\$} \mathtt{Gen}(1^k)$ and compute a signature $\sigma = \mathtt{Sign}(y, \mathsf{sk})$.
3. Choose a uniformly random $i^* \in [n]$ and set $y_{i^*} = (y, \sigma)$ and for each $i \in [n] \setminus \{i^*\}$, set y_i to a default value (e.g., $y_j = \bot$).
4. Each $p_j \in \mathcal{P}$ receives as (private) output the value (y_j, vk).

Protocol $\Pi^{\text{Opt},f}_{\text{nSFE}}$

1. The parties use protocol Π_{GMW} [17] to evaluate the functionality $\mathcal{F}^{f,\bot}_{\text{PRIV-SFE}}$. If Π_{GMW} aborts then $\Pi^{\text{Opt},f}_{\text{nSFE}}$ also aborts' otherwise every party p_i denotes its output by (y_j, vk).
2. Every party broadcasts y_j. If no party broadcast a pair $y_j = (y, \sigma)$ where σ is a valid signature on y for the key vk then every party aborts. Otherwise, every party output y.

As proven in the full version of this work [15], the utility that any adversary $\mathcal{A}$ accrues against $\Pi^{\text{Opt}}_{\text{nSFE}}$ is

$$u_{\mathtt{A}}(\Pi^{\text{Opt}}_{\text{nSFE}}, \mathcal{A}) \overset{\text{negl}}{\leq} \frac{(n-1)\gamma_{10} + \gamma_{11}}{n},$$

[11] In fact, we do not need to authenticate the default value.

which is in fact optimal (also proven there).

Utility-balanced fairness. A closer look at the above results shows that an adversary who is able to corrupt parties for free is always better off corrupting $n-1$ parties. While this is natural in the case of two parties, in the multi-party case one might be interested in more "fine-grain" optimality notions, which are sensitive to the number of corrupted parties. One such natural notion, which we now present, requires that the allocation of utility to adversaries corrupting different numbers of parties be tight, in the sense that the utility of a best *t-adversary*—i.e., any adversary that optimally attacks the protocol while corrupting up to t parties—cannot be decreased unless the utility of a best t'-adversary increases, for $t' \neq t$.[12] This leads to the notion of *utility-balanced* fairness.

DEFINITION 9. *Let* $\vec{\gamma} \in \Gamma^+_{fair}$. *A multi-party protocol* Π *is* utility-balanced $\vec{\gamma}$-fair (w.r.t. corruptions) *if for any protocol* Π', *for every* $(\mathcal{A}_1, \ldots, \mathcal{A}_{n-1})$ *and* $(\mathcal{A}'_1, \ldots, \mathcal{A}'_{n-1})$ *the following holds:*

$$\sum_{t=1}^{n-1} u_{\mathtt{A}}(\Pi, \mathcal{A}_t) \overset{\text{negl}}{\leq} \sum_{t=1}^{n-1} u_{\mathtt{A}}(\Pi', \mathcal{A}'_t),$$

where for $t = 1, \ldots, n-1$, $\mathcal{A}_t$ *and* $\mathcal{A}'_t$ *are t-adversaries attacking protocols* Π *and* Π', *respectively.*[13]

In the full version, we show that protocol $\Pi^{\text{Opt}}_{\text{nSFE}}$ is in fact utility-balanced $\vec{\gamma}$-fair. To this end, we first prove that the sum of the expected utilities of the different t-adversaries is

$$\sum_{t=1}^{n-1} u_{\mathtt{A}}(\Pi^{\text{Opt}}_{\text{nSFE}}, \mathcal{A}_t) \overset{\text{negl}}{\leq} \frac{(n-1)}{2}(\gamma_{10} + \gamma_{11}), \tag{4}$$

which we then show to be tight for certain functions. In fact, our upper bound provides a good criterion for checking whether or not a protocol is utility-based $\vec{\gamma}$-fair: if for a protocol there are t-adversaries, $1 \leq t \leq n-1$, such that the sum of their utilities non-negligibly exceeds this bound, then the protocol is not utility-balanced $\vec{\gamma}$-fair. We observe that protocols that are fair according to the traditional fairness notion [17] are not necessarily utility-balanced $\vec{\gamma}$-fair—the reason is that they "give up" completely for $n/2$ parties if n is even. Furthermore, although the protocol $\Pi^{\text{Opt}}_{\text{nSFE}}$ presented above satisfies both utility-based notions (optimal and utility-balanced), these two notions are in fact incomparable; separating examples are given in the full version.

Utility-balanced fairness as optimal fairness with corruption costs. As discussed above, the notion of utility-balanced fairness connects the ability (or willingness) of the adversary to corrupt parties with the utility he obtains. Thus, a natural interpretation of utility-balanced $\vec{\gamma}$-fairness is as a desirable optimality notion when some information about the cost of corrupting parties is known; for example, it is known that certain sets of parties might be easier to corrupt than others. We now show that if we associate a cost to party corruption (as a negative utility for the adversary) then there is a natural connection between utility-balanced $\vec{\gamma}$-fairness and optimal $\vec{\gamma}$-fairness. We first slightly modify the definition of an attacker's utility to account for corruption cost, along the lines of [14].

Specifically, in addition to the events E_{ij} specified in Section 3, we also define, for each subset $\mathcal{I} \subseteq [n]$ of parties, the event $E_{\mathcal{I}}$ that occurs when the adversary corrupts *exactly* the parties in $\mathcal{I}$. The cost of corrupting each such set $\mathcal{I}$ is specified via a function $\mathcal{C} : 2^{\mathcal{P}} \to \mathbb{R}$, where for any $\mathcal{I} \subseteq \mathcal{P}$, $\mathcal{C}(\mathcal{I})$ describes the cost associated with corrupting the players in $\mathcal{I}$. We generally let the corruption costs $\mathcal{C}(\mathcal{I})$ be non-negative. Thus, the adversary's payoff is specified by the events $\vec{E}^{\mathcal{C}} = (E_{00}, E_{01}, E_{10}, E_{11}, \{E_{\mathcal{I}}\}_{\mathcal{I}\subseteq\mathcal{P}})$ and by the corresponding payoffs $\vec{\gamma}^{\mathcal{C}} = (\gamma_{00}, \gamma_{01}, \gamma_{10}, \gamma_{11}, \{-\mathcal{C}(\mathcal{I})\}_{\mathcal{I}\subseteq\mathcal{P}})$. The expected payoff of a given simulator $\mathcal{S}$ (for an environment $\mathcal{Z}$) is redefined as:

$$U_I^{\mathcal{F}^{\perp}_{\text{SFE}}, \vec{\gamma}^{\mathcal{C}}}(\mathcal{S}, \mathcal{Z}) := \sum_{i,j\in\{0,1\}} \gamma_{ij} \Pr[E_{ij}] - \sum_{\mathcal{I}\subseteq\mathcal{P}} \mathcal{C}(\mathcal{I}) \Pr[E_{\mathcal{I}}]. \tag{5}$$

We write $\vec{\gamma}^{\mathcal{C}} \in \Gamma^{+\mathcal{C}}_{\text{fair}}$ to denote the fact that for the sub-vector $\vec{\gamma} = (\gamma_{00}, \gamma_{01}, \gamma_{10}, \gamma_{11})$ of $\vec{\gamma}^{\mathcal{C}}$, $\vec{\gamma} \in \Gamma^+_{\text{fair}}$. Given that the adversary incurs a cost for corrupting parties, we can show that protocols are *ideally* $\vec{\gamma}^{\mathcal{C}}$*-fair* which, roughly speaking, means that the protocol restricts its adversary as much as a completely fair protocol—according to the standard notion of fairness—would. We show that utility-balanced fairness implies an optimality (with respect to the cost function) on ideal $\vec{\gamma}^{\mathcal{C}}$-fairness. (See Definition 19 in [15].) For cost functions that only depend on the number of parties (i.e., $\mathcal{C}(\mathcal{I}) = c(|\mathcal{I}|)$ for $c : [n] \to \mathbb{R}$), we show the following theorem in the full version.

THEOREM 10. *Let* $\vec{\gamma} = (\gamma_{00}, \gamma_{01}, \gamma_{10}, \gamma_{11}) \in \Gamma^+_{fair}$. *For a protocol* Π *that is utility-balanced* $\vec{\gamma}$*-fair, the following two statements hold:*

1. Π *is ideally* $\vec{\gamma}^{\mathcal{C}}$*-fair with respect to cost vector* $\vec{\gamma}^{\mathcal{C}} = (\gamma_{00}, \gamma_{01}, \gamma_{10}, \gamma_{11}, \{-\mathcal{C}(\mathcal{I})\}_{\mathcal{I}\subseteq\mathcal{P}}) \in \Gamma^{+\mathcal{C}}_{fair}$ *for the function* $\mathcal{C}(\mathcal{I}) = c(|\mathcal{I}|) = u_{\mathtt{A}}(\Pi, \mathcal{A}_{|\mathcal{I}|})$, *where* $\mathcal{A}_{|\mathcal{I}|}$ *is the best adversary strategy corrupting up to* $|\mathcal{I}|$ *parties.*

2. *The cost function* $\mathcal{C}$ *above is optimal in the sense that there is no protocol which is ideally* $\vec{\gamma}^{\mathcal{C}'}$*-fair with a cost function* $\mathcal{C}'$ *that is strictly dominated by* $\mathcal{C}$ *(see full version for formal definition).*

5. UTILITY-BASED VS. PARTIAL FAIRNESS

A notion that is closely related to our fairness notion is the concept of $1/p$-security—also called *partial fairness*—introduced by Gordon and Katz [18]. Roughly speaking, the notion allows a "distinguishing gap" of at most $1/p$ (for a polynomial p) between the real-world protocol execution and the ideal-world evaluation of the function. Furthermore, all statements discussed informally in this section are proven in the full version of this work.

At a high level, $1/p$-security appears to correspond to bounding the adversary's utility to $\frac{p-1}{p} \cdot \gamma_{11} + \frac{1}{p} \cdot \gamma_{10}$, since the protocol leads to a "fair" outcome with probability $1 - 1/p$ and to an "unfair" outcome with probability $1/p$. This is

[12] One can define an even more fine-grain notion of utility balancing, which explicitly puts a bound on the utility of the best t-adversary $\mathcal{A}_t$ *for every* t (instead of bounding the sum). See next subsection and the full version.

[13] Note that we exclude from the sum the utilities of adversaries that do not corrupt any party ($t = 0$) or corrupt every party ($t = n$), since by definition for every protocol these utilities are γ_{01} and γ_{11}, respectively.

a better bound than proven in Theorem 3 for our "optimal" protocol—which appears to be a contradiction to our optimality result. The protocols of Gordon and Katz [18], however, only apply to functions for which the size of either one party's input domain or one party's output range is bounded by a polynomial. Our protocols do not share this restriction, and the impossibility result in Lemma 5 is shown using a function which has exponential input domains and output ranges.

A weakness of $1/p$-security. In general, $1/p$-security allows privacy (and not only fairness) to be violated with probability $1/p$. Noticing this, Gordon and Katz [18] already suggested that one might additionally require that a $1/p$-protocol be *private*. We point out, however, that even protocols that are both $1/p$-secure and private may have subtle problems. Intuitively, the issue is that privacy and correctness are considered separately, rather than jointly as in standard simulation-based security definitions. For example, consider the following protocol $\tilde{\Pi}$ for computing the logical "and" of two parties' inputs x_1 and x_2:

- The first message is a 0-bit sent from p_2 to p_1.
- If p_2 sent a 1-bit instead of a 0-bit, then p_1 tosses a biased coin C with $\Pr[C = 1] = \frac{1}{4}$, and sends its input x_1 to p_2 if $C = 1$ (or otherwise an empty message).
- Then, p_1 and p_2 engage in any $\frac{1}{4}$-secure protocol to compute $x_1 \wedge x_2$.

In the full version of this work, we show that this protocol is private; this is because p_2 can learn x_1 even in the ideal world (by sending a 1 to the ideal functionality). We also show that the protocol is $1/2$-secure. Yet it allows p_2 to learn x_1 and simultaneously force p_1 to output 0.

Analysis of the Gordon-Katz protocols using our approach. Gordon and Katz [18] propose two protocols: one for functions that have (at least) one domain of polynomial size, and one for functions in which (at least) one range is polynomial size. The underlying idea of the protocols is to reconstruct the output in multiple rounds and to provide the actual output starting at a round chosen at random. In all previous rounds, a random output is given. We stress that the protocols are proven secure only with respect to static corruptions; all the statements we make in this section are in this setting.

The protocols described by Gordon and Katz do not realize functionality $\mathcal{F}_{\text{SFE}}^{f,\perp}$, as the correctness of the honest party's output is not guaranteed. In fact, it is inherent to their protocols that if the adversary aborts early, then the honest party may output a random output instead of the correct one. Hence, to formalize the guarantees achieved by those protocols, we weaken our definition by modifying the functionality $\mathcal{F}_{\text{SFE}}^{f,\perp}$ to allow for a correctness error; specifically, the weakened functionality allows the adversary to replace the honest party's output by a randomly chosen output. The original protocol for functions with one polynomial domain (see [18, Section 3.2]) achieves this functionality and bounds the adversary's payoff. The statements about the protocol for functions with polynomial size range transfer analogously.

Comparing $1/p$-security with our notion. Our definition as described in the previous paragraph is *strictly stronger* than $1/p$-security, even if the latter notion is strengthened by additionally requiring privacy as suggested in [18]. Indeed, for the payoff vector $\vec{\gamma} = (0, 0, 1, 0)$ a security statement in our model implies $1/p$-security, and protocol $\tilde{\Pi}$ described above shows that our notion is strictly stronger.

6. ACKNOWLEDGMENTS

Jonathan Katz was supported by NSF grants #0830464, #1111599, and #1223623. Björn Tackmann was supported by the Swiss National Science Foundation (SNF) via the Fellowship no. P2EZP2-155566 and the NSF grant CNS-1228890. Vassilis Zikas was supported by the Swiss National Science Foundation (SNF) via the Ambizione grant PZ00P-2142549.

7. REFERENCES

[1] Gilad Asharov, Ran Canetti, and Carmit Hazay. Towards a game theoretic view of secure computation. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 426–445, Heidelberg, 2011. Springer.

[2] Donald Beaver and Shafi Goldwasser. Multiparty computation with faulty majority. In *Proceedings of the 30th Symposium on Foundations of Computer Science*, pages 468–473. IEEE, 1989.

[3] 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, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 277–296, Heidelberg, 2011. Springer.

[4] Manuel Blum. How to exchange (secret) keys. *ACM Transactions on Computer Science*, 1:175–193, 1984.

[5] Dan Boneh and Moni Naor. Timed commitments. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 236–254, Heidelberg, 2000. Springer.

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

[7] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*, pages 136–145. IEEE, 2001.

[8] 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 [7].

[9] Ran Canetti, Uri Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *Twenty-Eighth Annual ACM Symposium on Theory of Computing*, pages 639–648. ACM, ACM Press, 1995.

[10] 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.

[11] Ivan Damgård. Practical and provably secure release of a secret and exchange of signatures. *Journal of Cryptology*, 8(4):201–222, 1995.

[12] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, volume 3027 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 2004.

[13] Juan A. Garay, David S. Johnson, Aggelos Kiayias, and Moti Yung. Resource-based corruptions and the combinatorics of hidden diversity. In Robert D. Kleinberg, editor, *Innovations in Theoretical Computer Science, ITCS '13, Berkeley, CA, USA*, pages 415–428. ACM, 2013.

[14] Juan A. Garay, Jonathan Katz, Ueli Maurer, Björn Tackmann, and Vassilis Zikas. Rational protocol design: Cryptography against incentive-driven adversaries. In *54th Annual Symposium on Foundations of Computer Science*. IEEE, 2013.

[15] Juan A. Garay, Jonathan Katz, Björn Tackmann, and Vassilis Zikas. How fair is your protocol? Cryptology ePrint Archive, Report 2015/187, March 2015.

[16] Juan A. Garay, Philip 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. Springer, 2006.

[17] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game—A completeness theorem for protocols with honest majority. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*, pages 218–229. ACM, 1987.

[18] Dov Gordon and Jonathan Katz. Partial fairness in secure two-party computation. In Henry Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 157–176. Springer, 2010.

[19] Ronen Gradwohl, Salil Vadhan, and David Zuckerman. Random selection with an adversarial majority. In *Proceedings of the 26th Annual International Conference on Advances in Cryptology*, CRYPTO'06, pages 409–426, Berlin, Heidelberg, 2006. Springer-Verlag.

[20] Adam Groce and Jonathan Katz. Fair computation with rational players. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 81–98. Springer, 2012.

[21] Jonathan Katz, Ueli Maurer, Björn Tackmann, and Vassilis Zikas. Universally composable synchronous computation. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 477–498, Heidelberg, 2013. Springer.

[22] 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.

[23] Benny Pinkas. Fair secure two-party computation. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 87–105, Heidelberg, 2003. Springer.

APPENDIX

This section contains material deferred from Section 4.1.

An authenticated secret sharing scheme. The sharing of a secret s (field element) is a pair (s_1, s_2) of random field elements (in some larger field) with the property that $s_1 + s_2 = (s, \mathsf{tag}(s, k_1), \mathsf{tag}(s, k_2))$, where k_1 and k_2 are MAC keys associated with the parties p_1 and p_2, respectively, and $\mathsf{tag}(x, k)$ denotes a MAC tag for the value x computed with key k. We refer to the values s_1 and s_2 as the *summands*. Each $p_i \in \{p_1, p_2\}$ holds his *share* $(s_i, \mathsf{tag}(s_i, k_{\neg i}))$ along with the MAC key k_i which is used for the generation of the MAC tags he is supposed to verify. We denote by $\langle s \rangle$ a sharing of s and by $\langle s \rangle_i$ party p_i's share. The above sharing can be reconstructed towards any of the parties p_i as follows: $p_{\neg i}$ sends his share $\langle s \rangle_{\neg i} = (s_{\neg i}, t_{\neg i})$ to p_i who, using k_i, verifies that $t_{\neg i}$ is a valid MAC for $s_{\neg i}$. Subsequently, p_i reconstructs the authenticated secret s by computing $s_1 + s_2 := (s, t'_1, t'_2)$ and verifying, using key k_i, that t'_i is a valid MAC for s. If any of the MAC verifications fails then p_i aborts and outputs $\perp$.