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

March 2, 2015

Abstract

In his seminal result, Cleve [STOC'86] established that secure distributed computation—guaranteeing fairness—is impossible in the presence of dishonest majorities. A generous number of proposals for relaxed notions of fairness ensued this seminal result, by weakening in various ways the desired security guarantees. While these works also suggest completeness results (i.e., the ability to design protocols which achieve their fairness notion), their assessment is typically of an all-or-nothing nature. That is, when presented with a protocol which is not designed to be fair according to their respective notion, they most likely would render it unfair and make no further statement about it.

In this work we put forth a comparative approach to fairness. We present new intuitive notions that when presented with two *arbitrary* protocols, provide the means to answer the question "Which of the protocols is fairer?" The basic idea is that we can use an appropriate utility function to express the preferences of an adversary who wants to break fairness. Thus, we can compare protocols with respect to how fair they are, placing them in a partial order according to this relative-fairness relation.

After formulating such utility-based fairness notions, we turn to the question of finding optimal protocols—i.e., maximal elements in the above partial order. We investigate—and answer—this question for secure function evaluation, both in the two-party and multi-party settings.

To our knowledge, the only other fairness notion providing some sort of comparative statement is that of $1/p$-*security* (aka "partial fairness") by Gordon and Katz [Eurocrypt'10]. We also show in this paper that for a special class of utilities our notion strictly implies $1/p$-security. In addition, we fix a shortcoming of the definition which is exposed by our comparison, thus strengthening that result.

Key words: Cryptographic protocols, secure multi-party computation, fairness, rational protocol design.

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, with communication over secure channels.

- 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 would render both protocols equally unfair and make no further statement about their relative fairness. This even applies to definitions for relaxed notions of fairness that circumvent Cleve's impossibility result [10], which excludes the existence of a fully fair protocol computing arbitrary functions in the presence of a dishonest majority. For example, both above protocols would be equally unfair with respect to *resource fairness* [15], 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 any 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 fairness definitions in "rational" cryptography (e.g., [1, 20]), which require the protocol to be an equilibrium strategy with respect to a preference/utility function for *curious-but-exclusive* agents, where each agent prefers learning the output to not learning it, but would rather be the only one that learns it. We point out that some of these frameworks do offer completeness results, in the sense that they show that one *can* construct protocols for contract signing that are fair in the respective notions; nevertheless, none of them provides a comparative statement for protocols as the ones we consider in the example above.

Motivated by the above observation, in this paper we put forth intuitive 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 protocol fairness. After Cleve's impossibility result [10], perhaps the most profuse line of work is on "gradual release" of information [4, 2, 11, 5, 23, 15]. These works adopt the traditional all-or-nothing definitional approach. A notable exception is the notion of $1/p$*-security* by Gordon and Katz [18]. Roughly, their definition, in the two-party case, guarantees fairness to hold with probability at least $1 - 1/p$, for some polynomial p, instead of with overwhelming probability. These results were extended to the multi-party case by Beimel *et al.* [3].

While Gordon and Katz considered the secure evaluation of specific classes of functions (with polynomial input domains and/or output ranges) and with the goal of achieving, at the cost of many rounds, arbitrarily large polynomials p, one can more generally adopt the parameter p as a measure of the protocol's quality, even for small values of p. Returning to the discussion about protocols Π_1 and Π_2 above, for example, this would mean that Π_1 is 0-secure ($p = 1$) and Π_2 is $1/2$-secure ($p = 2$) with respect to the ideal functionality that receives the signing keys from the parties and sends them the respective signed contracts. At a protocol level, the difference in our work is that we design and prove protocols for evaluating *arbitrary* functions, at the cost of achieving worse—but for this generality optimal—parameters. At a definitional level, we observe that our definition always (except with negligible probability) guarantees privacy and correctness, which is not the case for $1/p$-security (as we show in Section 5). In fact, we prove that, for an appropriate choice of the utility function, our utility-based fairness notion *strictly* implies $1/p$-security.

A different line of work tries to capture relaxed notions of fairness by assuming that the protocol participants are rational agents with a fairness-related utility function [1, 20]. Informally, every party/rational agent prefers to learn the output and be the only one who learns it. We point out that 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 whichever protocol is designed for them. 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, however, that similarly to other existing definitions of fairness, the definitions from [1, 20] also 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

[1] We note in passing that our protocols do, in fact, imply an equilibrium, but in the attack "meta-game" defined in [14]. Roughly speaking, an equilibrium in that attack game means that the corresponding protocol tames its adversary in an optimal way. Interested readers are referred to [14] for more details.

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 are presented in the appendix.

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 $f \overset{\text{negl}}{\approx} g \;\Leftrightarrow\; \exists \text{ negligible } \mu : \; |f - g| \leq \mu$, and $f \overset{\text{negl}}{\geq} g \;\Leftrightarrow\; \exists \text{ negligible } \mu : \; f \geq g - \mu$, 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 $z \in \{0, 1\}^*$.

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* $\texttt{D}$ and an *attacker* $\texttt{A}$. The designer $\texttt{D}$ plays first by specifying a protocol Π for the (honest) participants to run; subsequently, the attacker $\texttt{A}$, who is informed about $\texttt{D}$'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 $u_{\texttt{A}}$ of the adversary as the game is zero-sum. (The utility $u_{\texttt{D}}$ of the designer is then $-u_{\texttt{A}}$.)

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

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

[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].

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:

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, 16] 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 $\mathcal{F}_{\text{SFE}}^{\perp}$). In a nutshell, $\mathcal{F}_{\text{SFE}}^{\perp}$ is similar to standard SFE but allows the simulator to ask or not for corrupted parties' outputs, and, subsequently, to send $\mathcal{F}_{\text{SFE}}^{\perp}$ 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 $\mathcal{F}_{\text{SFE}}^{\perp}$-*ideal world.* We point out that the functionality $\mathcal{F}_{\text{SFE}}^{\perp}$ is as usually parametrized by the actual function f to be evaluated; when we want to make this function f explicit we will write $\mathcal{F}_{\text{SFE}}^{f,\perp}$.

Step 2: Events and payoffs. Next, we specify a set of events in the experiment corresponding to the ideal evaluation of $\mathcal{F}_{\text{SFE}}^{\perp}$ 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 $\mathcal{F}_{\text{SFE}}^{\perp}$ 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 $\mathcal{F}_{\text{SFE}}^{\perp}$ 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 $\mathcal{F}_{\text{SFE}}^{\perp}$ for some corrupted party's output and instructs it to abort before any honest party receives the output.

E_{11}: The simulator asks $\mathcal{F}_{\text{SFE}}^{\perp}$ 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]:

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

Step 3: Defining the attacker's utility. Given $U_I^{\mathcal{F}_{\text{SFE}}^{\perp},\vec{\gamma}}(\mathcal{S},\mathcal{Z})$, 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 $\mathcal{F}_{\text{SFE}}^{\perp}$-ideal world in presence of the least favorable environment—i.e., the one that is *most* favorable to the attacker (cf. Remark 1). To make the payoff vector $\vec{\gamma}$ explicit, we sometimes denote the above utility as $\hat{U}^{\Pi,\mathcal{F}_{\text{SFE}}^{\perp},\vec{\gamma}}(\mathcal{A})$ 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, $\mathtt{SIM}_{\mathcal{A}} = \{\mathcal{S} \in \mathtt{ITM} \mid \forall \mathcal{Z} : \text{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}} \approx \text{EXEC}_{\mathcal{F}_{\text{SFE}}^{\perp},\mathcal{S},\mathcal{Z}}\}$. The payoff of strategy $\mathcal{A}$ (for attacking Π) is then defined as:

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

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:

[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].

Definition 1. Let Π and Π' be protocols, and $\vec{\gamma} \in \Gamma_{\text{fair}}$ 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

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

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

Remark 1 (On using the worst-case adversary). The above definition renders a protocol optimal if it defends against the worst adversary/environment that attacks it. This is essential for obtaining a composable security notion. (In fact, as shown in [14, Section 3] when instantiated in Canetti's powerful UC framework [7], our definition allows for replacing any functionality with a corresponding UC-secure protocol.) One might be tempted to try to obtain stronger notions of optimality by formulating a"per-adversary" definition, in which Equation (3) is required to hold for the same adversary both for Π and Π'. Yet, such a definition would render most interesting protocols incomparable, as one can design adversaries that are "tuned" to attack one protocol but not the other.

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 [8], 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 an utility which matches the above 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 Appendix A 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, $\Pi_{\text{2SFE}}^{\text{Opt}}$ invokes an adaptively secure unfair SFE protocol (e.g., the protocol in [16]—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 index $i \in_{\text{R}} \{1, 2\}$ 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 of the proof (see Appendix A) 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}.

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

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 (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.[7]

Theorem 4. *Let $\vec{\gamma} \in \Gamma_{\text{fair}}$, 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 $u_{\mathtt{A}}(\Pi, \mathcal{A}) \overset{\text{negl}}{\geq} \frac{\gamma_{10}+\gamma_{11}}{2}$.*

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

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

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 A.1 for details. Next, we consider multi-party SFE (i.e., $n > 2$).

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, $\Pi^{\text{Opt}}_{\text{nSFE}}$, is as follows (see Appendix B for a detailed description): 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 $i \in [n] \setminus \{i^*\}$, $y_i = \perp$; in addition to y_i, every party p_i receives an authentication tag on y_i.[8] 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 $y \neq \perp$ is broadcast, then the parties adopt it; otherwise, they abort.

As proven in Appendix B (Lemma 11), the utility that any adversary $\mathcal{A}$ accrues against $\Pi^{\text{Opt}}_{\text{nSFE}}$ is

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

which is in fact optimal (Lemma 13).

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 $t' \neq t$.[9] This leads to the notion of *utility-balanced* fairness.

Definition 5. Let $\vec{\gamma} \in \Gamma^{+}_{\text{fair}}$. 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:

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

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

[9] 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 Appendix B.2.

where for $t = 1, \ldots, n-1$, $\mathcal{A}_t$ and $\mathcal{A}'_t$ are t-adversaries attacking protocols Π and Π', respectively.[10]

In Appendix B we show that protocol $\Pi^{\text{Opt}}_{\text{nSFE}}$ is in fact utility-balanced $\vec{\gamma}$-fair. To this end, we first prove (Lemma 14) that the sum of the expected utilities of the different t-adversaries is

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

which we then show to be tight for certain functions (Lemma 16). 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 [16] 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. We demonstrate this in Appendix B.1 by providing separating examples.

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:

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

We will 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.) For the following theorem, we consider cost functions that only depend on the number of parties (i.e., $\mathcal{C}(\mathcal{I}) = c(|\mathcal{I}|)$ for $c : [n] \to \mathbb{R}$). The proof is in Appendix B.2.

Theorem 6. *Let $\vec{\gamma} = (\gamma_{00}, \gamma_{01}, \gamma_{10}, \gamma_{11}) \in \Gamma^{+}_{\text{fair}}$. For a protocol Π that is utility-balanced $\vec{\gamma}$-fair, the following two statements hold:*

[10] 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.

1. Π *is ideally* $\vec{\gamma}^{\mathcal{C}}$*-fair with* $\vec{\gamma}^{\mathcal{C}} = (\gamma_{00}, \gamma_{01}, \gamma_{10}, \gamma_{11}, \{-\mathcal{C}(\mathcal{I})\}_{\mathcal{I}\subseteq\mathcal{P}}) \in \Gamma_{\text{fair}}^{+\mathcal{C}}$ *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* $\mathcal{C}$ *according to Definition 20.*

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] for the two-party setting, and generalized by Beimel *et al.* [3] to the multi-party setting. Roughly speaking, the notion allows a noticeable error of at most $1/p$ (for a polynomial p) in the security statement for a protocol, in contrast to the negligible gap allowed by standard definitions. For a more detailed description of $1/p$-security, we refer to Appendix C and to the original paper of Gordon and Katz [18].

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 $(p-1)/p$ and to an "unfair" outcome with probability $1/p$. This is a better bound than proven in Theorem 3 for our "optimal" protocol—which appears to be a contradiction to the optimality result. The protocols of Gordon and Katz [18], however, only apply to functions for which the size of either (at least) one party's input domain or (at least) one party's output range is bounded by a polynomial. Our protocols do not share this restriction, and the impossibility result in Lemma 4 is shown based on a function which has exponential input domains and output ranges.

A weakness of $1/p$-security. The definition of partial fairness allows for an honest party's input to be leaked with non-negligible probability. Somewhat surprisingly, this even holds if one additionally requires "full privacy," as suggested by Gordon and Katz [18]. The reason is that privacy and $1/p$-security are formalized as two completely separate requirements. We analyze the intuitively insecure protocol $\tilde{\Pi}$, which computes the logical "and" $\wedge : \{0,1\} \times \{0,1\} \to \{0,1\}$ as follows. Denote the inputs as x_1 and x_2 for p_1 and p_2, respectively:

- The first message is a 0-bit that is sent from p_2 to p_1.
- Yet, 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 the standard $\frac{1}{4}$-secure protocol to compute $x_1 \wedge x_2$.

We show in Lemma 27 (Appendix C.5) that this protocol is both 1/2-secure and fully private according to the notion defined in [18].

Analysis of the Gordon-Katz protocols based on our approach. Gordon and Katz [18] propose two protocols: one for functions that have (at least) one domain of (at most) polynomial size, and one for functions in which both domains might be large, but (at least) one range is at most polynomial. The underlying idea of the protocols is to reconstruct the output in multiple rounds and to provide the actual output starting from a round that was 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 sections are in this setting.

The protocols described by Gordon and Katz do not realize the 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 the 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 functionality $\mathcal{F}_{\text{SFE}}^{f,\$}$ we describe in Appendix C.2 allows the adversary to replace the honest party's

output by a randomly chosen one, which intuitively corresponds to contradicting the full security definition in the manner the protocols in [18] do. We show in Theorem 23 (Appendix C.3) that the protocol with 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. Finally, we show that 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 "full privacy" as suggested in [18]. For the payoff vector $\vec{\gamma} = (0, 0, 1, 0)$, a security statement in our model implies $1/p$-security. (See Lemma 25.)

The protocol $\tilde{\Pi}$ described in the above paragraphs, which leaks honest inputs but can still be proven to achieve both conditions of [18], serves as a separating example between the two notions. In fact, Lemma 26 shows that the protocol is not sufficient for our security notion. Its proof formalizes the intuition that in the functionality $\mathcal{F}_{\text{SFE}}^{f,\$}$, privacy is guaranteed, while the protocol easily allows to obtain the input of p_1. This strengthens the result of [18]. (Refer to Appendix C for further details.)

References

[1] Asharov, G., Canetti, R., Hazay, C.: Towards a game theoretic view of secure computation. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 426–445. Springer, Heidelberg (2011)

[2] Beaver, D., Goldwasser, S.: Multiparty computation with faulty majority. In: Proceedings of the 30th Symposium on Foundations of Computer Science. pp. 468–473. IEEE (1989)

[3] Beimel, A., Lindell, Y., Omri, E., Orlov, I.: $1/p$-secure multiparty computation without honest majority and the best of both worlds. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 277–296. Springer, Heidelberg (2011)

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

[5] Boneh, D., Naor, M.: Timed commitments. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 236–254. Springer, Heidelberg (2000)

[6] Canetti, R.: Security and composition of multiparty cryptographic protocols. Journal of Cryptology 13, 143–202 (April 2000), `http://www.springerlink.com/content/cxxd0r683kgk4nya`

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

[8] Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067 (December 2005), `http://eprint.iacr.org/2000/067`, a preliminary version of this work appeared in [7].

[9] Canetti, R., Feige, U., Goldreich, O., Naor, M.: Adaptively secure multi-party computation. In: Twenty-Eighth Annual ACM Symposium on Theory of Computing. pp. 639–648. ACM, ACM Press (1995)

[10] Cleve, R.E.: 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. pp. 364–369. ACM, Berkeley (1986)

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

[12] Freedman, M.J., Nissim, K., Pinkas, B.: Efficient private matching and set intersection. In: Cachin, C., Camenisch, J. (eds.) Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings. Lecture Notes in Computer Science, vol. 3027, pp. 1–19. Springer (2004), `http://dx.doi.org/10.1007/978-3-540-24676-3_1`

[13] Garay, J.A., Johnson, D.S., Kiayias, A., Yung, M.: Resource-based corruptions and the combinatorics of hidden diversity. In: Kleinberg, R. (ed.) Innovations in Theoretical Computer Science, ITCS '13, Berkeley, CA, USA. pp. 415–428. ACM (2013)

[14] Garay, J.A., Katz, J., Maurer, U., Tackmann, B., Zikas, V.: Rational protocol design: Cryptography against incentive-driven adversaries. In: 54th Annual Symposium on Foundations of Computer Science. IEEE (2013)

[15] Garay, J.A., MacKenzie, P., Prabhakaran, M., Yang, K.: Resource fairness and composability of cryptographic protocols. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 404–428. Springer (2006)

[16] Goldreich, O., Micali, S., Wigderson, A.: 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. pp. 218–229. ACM (1987)

[17] Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM J. Comput. 17(2), 281–308 (Apr 1988), http://dx.doi.org/10.1137/0217017

[18] Gordon, D., Katz, J.: Partial fairness in secure two-party computation. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 157–176. Springer (2010)

[19] Gradwohl, R., Vadhan, S., Zuckerman, D.: Random selection with an adversarial majority. In: Proceedings of the 26th Annual International Conference on Advances in Cryptology. pp. 409–426. CRYPTO'06, Springer-Verlag, Berlin, Heidelberg (2006), http://dx.doi.org/10.1007/11818175_25

[20] Groce, A., Katz, J.: Fair computation with rational players. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 81–98. Springer (2012)

[21] Katz, J., Maurer, U., Tackmann, B., Zikas, V.: Universally composable synchronous computation. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 477–498. Springer, Heidelberg (2013)

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

[23] Pinkas, B.: Fair secure two-party computation. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 87–105. Springer, Heidelberg (2003)

A Utility-based Fair Two-Party SFE (cont'd)

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 $\perp$.

Proof of the upper bound. The protocol $\Pi^{\mathrm{Opt}}_{\mathrm{2SFE}}$ implements the functionality $\mathcal{F}^{f,\perp}_{\mathrm{SFE}}$ with the best possible fairness in the two-party case. The "positive" part of this statement is formalized by the following theorem. 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 $\vec{\gamma} \in \Gamma_{\text{fair}}$ and $\mathcal{A}$ be an adversary. Then,*

$$u_{\mathtt{A}}(\Pi^{\text{Opt}}_{\text{2SFE}}, \mathcal{A}) \quad \overset{\text{negl}}{\leq} \quad \frac{\gamma_{10} + \gamma_{11}}{2}.$$

Proof (sketch). We prove the statement for $\Pi^{\text{Opt}}_{\text{2SFE}}$ in the $\mathcal{F}^{f',\perp}_{\text{SFE}}$-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 $\mathcal{F}^{f',\perp}_{\text{SFE}}$, $\mathcal{S}_{\mathcal{A}}$ 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 $\Pi^{\text{Opt}}_{\text{2SFE}}$:

- If $\hat{i} = 1$, then $\mathcal{S}_{\mathcal{A}}$ sends $\hat{x}_1$ (which it obtained because of the $\mathcal{F}^{f',\perp}_{\text{SFE}}$-hybrid model) to $\mathcal{F}^{f,\perp}_{\text{SFE}}$ and asks for the output[11]; 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 $t'_1 := \mathsf{tag}(y, \hat{k}_1)$ and $t'_2 := \mathsf{tag}(y, \hat{k}_2)$ for a uniformly chosen k_2. Set $\hat{s}_2 := (y, t'_1, t'_2) - s_1$ and $\hat{t}_1 := \mathsf{tag}(\hat{s}_2, \hat{k}_1)$. 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 `abort` to $\mathcal{F}^{f,\perp}_{\text{SFE}}$, 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 $\mathcal{F}^{f,\perp}_{\text{SFE}}$ (as p_1's input). Otherwise, it asks $\mathcal{F}^{f,\perp}_{\text{SFE}}$ 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 $\mathcal{F}^{f,\perp}_{\text{SFE}}$ 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

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

error in the simulation of $\mathcal{S}_{\mathcal{A}}$). Therefore, in any case the utility of the attacker choosing adversary $\mathcal{A}$ is $u_{\mathtt{A}}(\Pi^{\mathrm{Opt}}_{\mathrm{2SFE}}, \mathcal{A}) \overset{\mathrm{negl}}{\leq} \frac{\gamma_{10}+\gamma_{11}}{2}$ which concludes the proof. □

Proof of the lower bound. In the following we prove our general impossibility result. To this direction, 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).[12] 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 7. *Let f_{swp} be the swap function, $\mathcal{A}_1$ and $\mathcal{A}_2$ be the strategies defined above, and $\vec{\gamma} \in \Gamma_{\mathrm{fair}}$. Every protocol Π which securely realizes functionality $\mathcal{F}_{\mathrm{SFE}}^{f_{\mathsf{swp}},\perp}$ satisfies:*

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

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 $\mathcal{F}_{\mathrm{SFE}}^{f,\perp}$, 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

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

actual output when the $\mathcal{A}_2$ aborts, and with probability $1 - q_{\bar{L}}$ it does.[13] (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

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

which was exactly the statement we wanted to prove. □

Lemma 7 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 4. *Let $\vec{\gamma} \in \Gamma_{\text{fair}}$, 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 7, we obtain

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

which completes the proof. □

A.1 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,

[13] 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 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 8. 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 9. $\Pi^{\text{Opt}}_{\text{2SFE}}$ *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 4, this is not the case if the adversary aborts in the first round of phase 2. □

Lemma 10. *Assuming $\vec{\gamma} \in \Gamma_{\text{fair}}$, there exists no optimally $\vec{\gamma}$-fair protocol for computing the swap function f_{swp} (see Lemma 7) 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 7, 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. □

B Utility-based Fair Multi-Party SFE (cont'd)

Here we present the formal statements and proofs omitted from Section 4.2. We use the symbols $\overset{\text{negl}}{<}$ and $\overset{\text{negl}}{>}$ as the opposite of $\overset{\text{negl}}{\geq}$ and $\overset{\text{negl}}{\leq}$. We start with a detailed specification of our optimally $\vec{\gamma}$-fair protocol $\Pi^{\text{Opt}f}_{\text{nSFE}}$. The protocol $\Pi^{\text{Opt}f}_{\text{nSFE}}$ and the functionality $\mathcal{F}^{f,\perp}_{\text{PRIV-SFE}}$ which chooses one of the parties at random and hands him the output of f *in a private manner*. Note that, as in the two-party case [22], a private output functionality can be trivially computed by a public output protocol but the following mechanism: Let $f(x_1, \ldots, x_n) = (y_1, \ldots, y_n)$ be the function to be computed,

where each p_i has input x_i and receives private output y_i. Instead of computing f with private outputs, the parties can compute the public output function $f'((x_1,k_1),\ldots,(x_n,k_n)) = (y,\ldots,y)$, where $y = (y_1 \oplus k_1,\ldots,y_n \oplus k_n)$ and each k_i is a key randomly chosen by p_i—i.e., each party p_i hands as input, in addition to its f-input x_i, a one-time-pad key $k_i \in \{0,1\}^{|x_i|}$ and every party receives the entire vector of outputs of f, where each component is one-time pad encrypted (hence perfectly blinded) with the key of the corresponding party. Clearly, each party p_i can retrieve its private output y_i by decrypting the i-th component of y, but obtains no information of y_j's with $j \in [n] \setminus \{i\}$ which are perfectly blinded with the corresponding keys.

In the description of protocol $\Pi^{\mathrm{Opt}f}_{\mathrm{nSFE}}$ we make use of an existentially unforgeable digital signature scheme [17], denoted as $(\mathtt{Gen},\mathtt{Sign},\mathtt{Ver})$. Informally, such a scheme ensures that for a signing/verification key-pair $(\mathsf{sk},\mathsf{vk})$ generated by $\mathtt{Gen}$, a signature on any message that is generated by using algorithm $\mathtt{Sign}$ with key sk will always verify using key vk; however, the probability of the adversary generating a signature on a new message without knowing sk—i.e., forging a signature—is negligible, even if the adversary has seen polynomially many signatures on different messages (with this key sk).

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

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., $y_j = \perp$).
4. Each $p_j \in \mathcal{P}$ receives as (private) output the value (y_j,vk).

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

1. The parties use protocol Π_{GMW} [16] to evaluate the functionality $\mathcal{F}^{f,\perp}_{\text{PRIV-SFE}}$. If Π_{GMW} aborts then $\Pi^{\mathrm{Opt},f}_{\mathrm{nSFE}}$ 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.

Lemma 11. *Let $\vec{\gamma} \in \Gamma^{+}_{\mathrm{fair}}$ and $\mathcal{A}$ be a t-adversary. Then,*

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

Proof (sketch). We prove the statement for $\Pi^{\mathrm{Opt}}_{\mathrm{nSFE}}$ in the $\mathcal{F}^{f,\perp}_{\text{PRIV-SFE}}$-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]. To prove the lemma 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 $\frac{(n-1)\gamma_{10}+\gamma_{11}}{n}$. Such a simulator $\mathcal{S}_{\mathcal{A}}$ for an adversary $\mathcal{A}$ works as follows.

To emulate the execution of step 1, $\mathcal{S}_{\mathcal{A}}$ emulates towards $\mathcal{A}$ the functionality $\mathcal{F}^{f,\perp}_{\text{PRIV-SFE}}$ as follows: $\mathcal{S}_{\mathcal{A}}$ chooses $i^* \in [n]$ uniformly at random, generates the signature setup $\mathcal{F}^{f,\perp}_{\text{PRIV-SFE}}$ would, and for each $i \in [n] \setminus \{i^*\}$: $\mathcal{S}_{\mathcal{A}}$ sets $y_i = (\perp,\mathsf{vk})$. Note that as long as $\mathcal{A}$ has not corrupted p_{i^*}, $\mathcal{S}_{\mathcal{A}}$ does not involve the functionality $\mathcal{F}^{f,\perp}_{\text{PRIV-SFE}}$ in its simulation. If at any point the adversary requests to corrupt p_{i^*} then, in order to compute the output y_{i^*}, $\mathcal{S}_{\mathcal{A}}$ invokes the SFE functionality $\mathcal{F}^{f,\perp}_{\text{SFE}}$, receives the output y, and sets $y_{i^*} = ((y,\sigma),\mathsf{vk})$. (But $\mathcal{S}$ does not deliver the outputs to honest parties yet.) Finally, $\mathcal{S}_{\mathcal{A}}$ hands $\mathcal{A}$ his $\mathcal{F}^{f,\perp}_{\text{PRIV-SFE}}$ outputs. If at any point $\mathcal{A}$ requests to corrupt all n parties,

$\mathcal{S}_\mathcal{A}$ corrupts all parties and, given their inputs, completes the simulation in the straightforward manner.

For emulating the second step of the protocol, $\mathcal{S}_\mathcal{A}$ simply emulates the broadcasting of the emulated outputs towards the adversary. Again, $\mathcal{S}_\mathcal{A}$ only invokes the functionality $\mathcal{F}$ when $\mathcal{A}$ expects to see the message broadcast by p_{i^*}. From that point on, $\mathcal{S}_\mathcal{A}$ continues the simulation as he did before, where if the protocol would abort, the simulator halts.

It is straightforward to verify that $\mathcal{S}_\mathcal{A}$ is a good simulator for $\mathcal{A}$. Indeed, the fact that the view of $\mathcal{A}$ in the first step of the simulation is indistinguishable from the protocol execution follows immediately from the security of Π_{GMW}.[14] For the second step, the distribution of the simulated view of $\mathcal{A}$ conditioned on the distribution of his view of the first step, is clearly identical to $\mathcal{A}$'s real protocol view. In the remainder of the proof we show that the expected payoff of the above simulator is at most $\frac{(n-1)\gamma_{10}+\gamma_{11}}{n}$.

First, we show that an adversary maximizing his utility corrupts at most $t < n$ parties, by showing that for any adversary who corrupts all n parties at some point of the protocol execution, there exists an adversary corrupting $t < n$ parties which yields (at least) the same expected payoff: Indeed, if at any point the adversary goes from having corrupted $t < n$ parties to corrupting all n parties, then his payoff can not be increased the simulator will not provoke any of the events E_{ij} other than what he provoked before the extra $n - t$ corruption, because, as soon as every party gets corrupted, the simulator learns all the internal state of $\mathcal{F}_{\text{SFE}}^{f,\perp}$ and can continue the simulation without it (by the definition of the model). For the remainder of the proof consider an adversary who corrupts $t < n$ parties by the end of the protocol.

We argue that that the probability of $\mathcal{A}$ provoking the event E_{10} is t/n: Indeed, the probability that $\mathcal{A}$ provokes the event E_{10} equals the probability that $\mathcal{A}$ corrupts p_{i^*} *before he broadcasts the sharing* (as this is the only way the adversary can learn the output before honest parties do). However, because unless $\mathcal{A}$ corrupts i^* the simulated view gives $\mathcal{A}$ no information on i^* or the output y, the adversary's best chance is to try to guess i^* which, as t parties are corrupted, succeeds with probability t/n. Now, clearly, unless $\mathcal{A}$ can provoke E_{10}, which $\mathcal{A}$ recognizes as soon as he sees the output, the above simulator will provoke either E_{11} (i.e., deliver) or E_{01}. Because by assumption $\gamma_{01} < \gamma_{11}$. The expected utility of the adversary corrupting $t < n$ with above simulation will be $\leq \frac{\gamma_{10}t+(n-t)\gamma_{11}}{n}$.

Note that this does not assume a simultaneous broadcast channel, but merely the standard ideal broadcast channel from the distributed computation literature in which once the message is out—and an adversary who does not corrupt the sender can see it—it will be seen by all parties. □

Lower bound (multi-party case). Consider the adversaries $\mathcal{A}_{\bar{1}}, \ldots, \mathcal{A}_{\bar{n}}$ that behave as follows; similarly to the ones described in Section 4.1. In the beginning, $\mathcal{A}_{\bar{i}}$ corrupts *all parties but* p_i. Then, in each round, it obtains the received messages and computes the messages sent by each of the p_j with $j \neq i$ following the honest strategy. Before sending the messages to p_i, however, $\mathcal{A}_{\bar{i}}$ checks (by running the protocol machines hypothetically, but storing and later reverting to the previous state) whether any one of the p_j would provide output if the execution continued without the participation of p_i. If this is the case, $\mathcal{A}_{\bar{i}}$ aborts the protocol before sending (and has obtained the output), otherwise it simply proceeds in the same manner in the next round. For the described adversaries, we prove the following lemma.

Lemma 12. *Let $f : \{0,1\}^{*n} \to \{0,1\}^*$ such that $f(x_1, \ldots, x_n) = x_1 \parallel x_2 \parallel \ldots \parallel x_n$, $\mathcal{A}_{\bar{1}}, \ldots, \mathcal{A}_{\bar{n}}$ be the strategies defined above, and $\vec{\gamma} \in \Gamma_{\text{fair}}$. Every protocol Π which securely realizes functionality*

[14] Recall that we are assuming ideal secure channels, thus [16] is secure even against an adaptive adversary.

$\mathcal{F}_{\text{SFE}}^{f,\perp}$ *satisfies:*

$$\sum_{i=1}^{n} u_{\mathtt{A}}(\Pi, \mathcal{A}_{\bar{i}}) \quad \overset{\text{negl}}{\geq} \quad (n-1)\cdot\gamma_{10} + \gamma_{11}.$$

Proof. For simplicity, we assume that the protocol is perfectly correct. This is used at two steps below, where we assume that *if* the protocol outputs something at a preliminary stage, *then* this output is correct. Also, we assume that the protocol will definitely lead to the correct output if it is executed in full.

We define events $L_{r,j}^{i}$ on the execution of the protocol with $\mathcal{A}_{\bar{i}}$ as follows: Given the state $\rho_{r,j}$ of the protocol of p_j after round r,[15] the event $L_{r,j}^{i}$ means that "if one executes the protocol on state of p_j after round r (given that no further messages are received), then the protocol will output (the correct[16]) output, but not so for any p_k with $k \neq j$." Since up to the point where such an event becomes active, the adversary simply executes the protocol on behalf of the corrupted parties, the events $L_{r,j}^{1}, \ldots, L_{r,j}^{n}$ all appear with the same probabilities and we will simply refer to them by the same symbol $L_{r,j}$. Based on these events, we then also define the events $L_j = \bigcup_{\ell=1}^{\#rounds} L_{\ell,j}$ which can intuitively be described as p_j obtains the output first.

Now we have the following observations:

- The L_i are pairwise disjoint, i.e., $L_i \cap L_j = \emptyset$ for $i \neq j$;
- this in particular implies that for all $i \neq j$, $L_i \subseteq \bar{L}_j$,
- which in turn means that each element in $\bigcup_{i=1}^{n} L_i$ is contained in the multiset $\bar{L}_1 \uplus \cdots \uplus \bar{L}_n$ exactly $n-1$ times.
- Furthermore, each element in $\overline{\bigcup_{i=1}^{n} L_i}$ is contained in the multiset $\bar{L}_1 \uplus \cdots \uplus \bar{L}_n$ even n times.
- Now all we have to see is that in case of the event $\bar{L}_i$, the adversary $\mathcal{A}_{\bar{i}}$ gets payoff γ_{10}, whereas in case L_i it gets payoff γ_{11}.

As a result, the multiset $\bar{L}_1 \uplus \cdots \uplus \bar{L}_n$ contains all elements of the event space at least $(n-1)$ times, which by the assumption that $\gamma_{10} \geq \gamma_{11}$ means that the sum of the adversaries' utilities is at least $(n-1)\cdot\gamma_{10} + \gamma_{11}$. Note that, as in Lemma 7, the events translate to the corresponding events in the ideal execution (except for negligible probability differences) because otherwise an environment could distinguish (i.e., the assumed simulator would not be good). □

As in the two-party case, we can "mix" the above-described adversaries probabilistically and obtain a bound for one specific adversary.

Lemma 13. *Let $f : \{0,1\}^{*n} \to \{0,1\}^{*}$ such that $f(x_1, \ldots, x_n) = x_1 \parallel x_2 \parallel \ldots \parallel x_n$ and $\vec{\gamma} \in \Gamma_{\text{fair}}$. There exists an adversary $\mathcal{A}$ such that for every protocol Π which realizes functionality $\mathcal{F}_{\text{SFE}}^{f,\perp}$:*

$$u_{\mathtt{A}}(\Pi, \mathcal{A}) \quad \overset{\text{negl}}{\geq} \quad \frac{(n-1)\cdot\gamma_{10} + \gamma_{11}}{n}.$$

B.1 Utility-Balanced Fairness (cont'd)

Lemma 14. *Let $\vec{\gamma} \in \Gamma_{\text{fair}}^{+}$ and $\mathcal{A}$ be an adversary. Then,*

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

[15] Remember that adversary $\mathcal{A}_{\bar{i}}$ runs the honest protocol on the behalf of the honest parties, so the states of all protocol machines are well defined and so is the event.

[16] See assumption above.

Proof. As we prove Lemma 11 for the expected payoff of any t-adversary $\mathcal{A}_t$ with $t < n$ it holds that

$$u_{\mathtt{A}}(\Pi^{\mathrm{Opt}}_{\mathrm{nSFE}}, \mathcal{A}_t) \quad \overset{\mathrm{negl}}{\leq} \quad \frac{\gamma_{10}t + (n-t)\gamma_{11}}{n}.$$

Therefore

$$\sum_{t=1}^{n-1} u_{\mathtt{A}}(\Pi^{\mathrm{Opt}}_{\mathrm{nSFE}}, \mathcal{A}_t) \quad \overset{\mathrm{negl}}{\leq} \quad \sum_{t=0}^{n-1} \frac{t\gamma_{10} + (n-t)\gamma_{11}}{n} \quad = \quad \frac{(n-1)(\gamma_{10}+\gamma_{11})}{2}.$$

□

Here, we show negative results. In the following lemma, we consider the two adversarial strategies $\hat{\mathcal{A}}_t$ and $\bar{\mathcal{A}}_{n-t}$, which corrupt the parties $1, \ldots, t$ and $t+1, \ldots, n$, respectively.

Lemma 15. *Let $\vec{\gamma} = (\gamma_{00}, \gamma_{01}, \gamma_{10}, \gamma_{11}) \in \mathbb{R}^4$ and let $f : \{0,1\}^{*n} \to \{0,1\}^*$ such that $f(x_1, \ldots, x_n) = x_1 \parallel x_2 \parallel \ldots \parallel x_n$. Let Π be an n-party protocol realizing functionality $\mathcal{F}^{f,\perp}_{\mathrm{SFE}}$. Then,*

$$u_{\mathtt{A}}(\Pi, \hat{\mathcal{A}}_t) + u_{\mathtt{A}}(\Pi, \bar{\mathcal{A}}_{n-t}) \quad \overset{\mathrm{negl}}{\geq} \quad \gamma_{10} + \gamma_{11},$$

The proof follows immediately from the two-party case, as we can consider the protocol emulating $1, \ldots, t$ for p_1 and $t+1, \ldots, n$ for p_2 as a two-party protocol to which Lemma 7 applies. This simple lemma implies the following result.

Lemma 16. *Let $\vec{\gamma} \in \Gamma_{\mathrm{fair}}$ and $f(x_1, \ldots, x_n) = x_1 \parallel x_2 \parallel \ldots \parallel x_n$. Then there exists a vector $(\mathcal{A}_1, \ldots, \mathcal{A}_{n-1})$ where each $\mathcal{A}_t$ is a t-adversary such that for every protocol Π which securely realizes functionality $\mathcal{F}^{f,\perp}_{\mathrm{SFE}}$, it holds that*

$$\sum_{t=1}^{n-1} u_{\mathtt{A}}(\Pi, \mathcal{A}_t) \quad \overset{\mathrm{negl}}{\geq} \quad \frac{(n-1)}{2} \cdot (\gamma_{10} + \gamma_{11}).$$

Proof. We first assume that the number n of parties is odd. Then, by setting $\mathcal{A}_i = \hat{\mathcal{A}}_t$ and $\mathcal{A}_{n-t} = \bar{\mathcal{A}}_{n-t}$ for each $1 \leq t \leq \frac{n-1}{2}$ and using Lemma 15, we immediately obtain that

$$\sum_{t=1}^{n-1} u_{\mathtt{A}}(\Pi, \mathcal{A}_t) \quad \geq \quad \sum_{i=1}^{\frac{n-1}{2}} (u_{\mathtt{A}}(\Pi, \hat{\mathcal{A}}_i) + u_{\mathtt{A}}(\Pi, \hat{\mathcal{A}}_i)) \quad \overset{\mathrm{negl}}{\geq} \quad \frac{n-1}{2} \cdot (\gamma_{10} + \gamma_{11}),$$

which is exactly the bound we intended to show. For an even number n of parties, we additionally use Lemma 15 with $\hat{\mathcal{A}}_{n/2}$ and $\bar{\mathcal{A}}_{n/2}$ to obtain the result. □

Utility-balanced fairness vs. optimal fairness and the GMW protocol. In the following, we denote by $\Pi^{1/2}_{\mathrm{GMW}}$ the version of the SFE protocol in [16] which is secure (and fair) for any dishonest minority, i.e., as long as $t < n/2$.

Lemma 17. *Protocol $\Pi^{1/2}_{\mathrm{GMW}}$ is not utility-balanced $\vec{\gamma}$-fair.*

Proof (sketch). Protocol $\Pi^{1/2}_{\mathrm{GMW}}$ [16] guarantees full security (including fairness) against any adversary corrupting at most $\lfloor\frac{n-1}{2}\rfloor$ parties; but an adversary that corrupts more than $\lfloor\frac{n-1}{2}\rfloor$ parties can trivially violate fairness with probability 1, i.e., can always obtain the output and prevent the honest parties from receiving it. The reason is that the protocol computes an $\lceil\frac{n}{2}\rceil$-out-of-n verifiable secret sharing,[17] which is then publicly reconstructed. Hence, any coalition of at most $\lfloor\frac{n-1}{2}\rfloor$ parties has no chance of preventing the remaining $\geq \lceil\frac{n}{2}\rceil$ parties from reconstructing, whereas a coalition of $\lceil\frac{n}{2}\rceil$ can trivially block the reconstruction (and has already the shares it needs to learn the secret.) This means that for an even number of parties $n = 0 \mod 2$:

- For $n-1 \geq t \geq \frac{n}{2}$: $u_{\mathtt{A}}(\Pi^{1/2}_{\mathrm{GMW}}, \mathcal{A}_t) \overset{\text{negl}}{\approx} \gamma_{10}$, and
- for $1 < t < \frac{n}{2}$: $u_{\mathtt{A}}(\Pi^{1/2}_{\mathrm{GMW}}, \mathcal{A}_t) \overset{\text{negl}}{\approx} \gamma_{11}$

Thus,

$$\sum_{t=1}^{n-1} u_{\mathtt{A}}(\Pi^{1/2}_{\mathrm{GMW}}, \mathcal{A}_t) \overset{\text{negl}}{\geq} \frac{n-1}{2} \cdot (\gamma_{10}+\gamma_{11}) + (\gamma_{10}-\gamma_{11}) \overset{\text{negl}}{>} \frac{n-1}{2} \cdot (\gamma_{10}+\gamma_{11}) \overset{\text{negl}}{\approx} \sum_{t=1}^{n-1} u_{\mathtt{A}}(\Pi^{\mathrm{Opt}}_{\mathrm{nSFE}}, \mathcal{A}_t)$$

□

In the following we show that one can obtain a protocol which is utility-balanced $\vec{\gamma}$-fair but not optimally $\vec{\gamma}$-fair, thereby establishing a separation between the two notions. To this direction, we observe that if we restrict to an odd number of parties, then $\Pi^{1/2}_{\mathrm{GMW}}$ does achieve the bound proved in Lemma 11 for protocol $\Pi^{\mathrm{Opt}}_{\mathrm{nSFE}}$, i.e., for $n = 1 \mod 2$:

$$\sum_{t=1}^{n-1} u_{\mathtt{A}}(\Pi^{1/2}_{\mathrm{GMW}}, \mathcal{A}_t) \quad = \quad \frac{n-1}{2} \cdot (\gamma_{10}+\gamma_{11})$$

Thus the protocol Π' which invokes $\Pi^{1/2}_{\mathrm{GMW}}$ for $n = 1 \mod 2$ and $\Pi^{\mathrm{Opt}}_{\mathrm{nSFE}}$ for $n = 0 \mod 2$ is in fact utility-balanced $\vec{\gamma}$-fair for every n. However, protocol Π' is not optimally $\vec{\gamma}$-fair since it is less $\vec{\gamma}$-fair than $\Pi^{\mathrm{Opt}}_{\mathrm{nSFE}}$. Indeed, an optimal attack against Π' is by the adversary who corrupts $\lceil\frac{n}{2}\rceil$ parties (thereby learning the output from the sharing) and makes all corrupted parties abort after learning the output. Because the honest parties have no information on the output (the have less than $\lceil\frac{n}{2}\rceil$ shares) for the function f from Lemma 15, this adversary obtains utility γ_{10} which is bigger than the upper bound $\frac{(n-1)\gamma_{10}+\gamma_{11}}{2}$ on the utility that any attacker might obtain in attacking $\Pi^{\mathrm{Opt}}_{\mathrm{nSFE}}$ (Lemma 11).

Optimal fairness does not imply utility-balanced fairness. The above result showed that utility-based fairness does not imply optimal fairness. Here, we show that the other direction also does not hold, i.e., there exists a protocol which is optimally fair but does not achieve utility-balanced fairness. The protocol works as follows:

1. The parties execute phase 1 of the protocol $\Pi^{\mathrm{Opt}}_{\mathrm{nSFE}}$, i.e., until all parties obtained their output from the underlying (unfair) SFE-protocol. Assume that party p_i received the value y_i, while $y_j = \perp$ for all $j \neq i$.
2. Each party sends the value "0" to all other parties.

[17] An t-out-of-n verifiable secret sharing ensures that the shares of any $t-1$ parties (jointly) contain obtain no information on the shared value, but if at least t honest parties announce their shares then the output will be reconstructed (i.e, a $(t-1)$-adversary cannot confuse the honest parties into accepting a wrong value).

3. If p_i received only 0's from the other parties, it broadcasts the value y_i. Otherwise, p_i tosses a coin, and broadcasts the value only of the coin lands "heads." Otherwise, it sends the value only to those parties that did not send a 0.

4. All parties (including p_i) that received the value y_i output it.

The described protocol is artificial and does not achieve utility-balanced fairness; still, it is optimally fair according to Definition 2. This is shown in the following lemma.

Lemma 18. *Let Π be the protocol described above. The following two statements hold:*

1. *Π is optimally $\vec{\gamma}$-fair.*
2. *Π is not utility-balanced fair.*

Proof. We first show the first statement. Indeed, if the adversary corrupts $n-1$ parties, the utility is exactly as before, since the probability of the remaining party to receive the value is $1/n$, and the party will output the value in step 4. If the adversary sends only 0's in step 2, then the protocol is essentially the same as $\Pi^{\text{Opt}}_{\text{nSFE}}$ and hence the adversary's utility is also the same. Hence, we only have to show that the adversary cannot achieve more utility $\overset{\text{negl}}{>} ((n-1)\cdot\gamma_{10}+\gamma_{11})/n$ by corrupting $t < n-1$ parties and sending other messages in step 2. In this case, however, the probability that one of the remaining $n-t$ honest parties receives the value in step 1 is $(n-t)/n \geq 2/n$, and the probability of a broadcast (after the coin-toss) in step 3 is hence still $\geq 1/n$, so the adversary's utility is at most as large as when corrupting exactly $n-1$ parties. Overall, this means that the protocol is optimally $\vec{\gamma}$-fair.

For the second statement, we only have to observe that if the adversary corrupts a single party, say p_j. If p_j receives the output in step 1, the adversary aborts. Otherwise, p_j and sends a 1 to all parties in step 2, and obtains the correct value otherwise. This adversary will obtain utility $1/n\cdot\gamma_{10}+(n-1)/n\cdot(\gamma_{10}+\gamma_{11})/2$. Clearly, the utilities of this adversary and the "standard" $\mathcal{A}_{n-1}$ add up to

$$\frac{(n-1)\gamma_{10}+\gamma_{11}}{n}+\frac{\gamma_{10}}{n}+\frac{(n-1)(\gamma_{10}+\gamma_{11})}{2n}=\frac{(3n-1)\gamma_{10}+(n+1)\gamma_{11}}{2n},$$

and we can then continue as in Lemma 16 to obtain the result. □

B.2 Utility-Balanced Fairness and the Cost of Corruption (cont'd)

As shown in [14], for any choice of the payoff vector $\vec{\gamma}$, there is a natural lower bound on the utility of the best adversary attacking any protocol, which is the utility of the best adversary attacking the "dummy" $\mathcal{F}_{\text{SFE}}$-hybrid protocol $\Phi^{\mathcal{F}_{\text{SFE}}}$ (i.e., the protocol that can invoke as a hybrid the trusted party $\mathcal{F}_{\text{SFE}}$ for SFE from [6]).[18] We refer to a protocol that performs at least as well as $\Phi^{\mathcal{F}_{\text{SFE}}}$ as *ideally fair*.

Definition 19. Let Π be a protocol and $\vec{\gamma}^{\mathcal{C}} \in \Gamma_{\text{fair}}$. We say that Π is *ideally $\vec{\gamma}^{\mathcal{C}}$-fair* if

$$\sup_{\mathcal{A}\in\mathtt{ITM}} u_{\mathtt{A}}(\Pi,\mathcal{A}) \quad \overset{\text{negl}}{\leq} \quad \sup_{\mathcal{A}\in\mathtt{ITM}} u_{\mathtt{A}}(\Phi^{\mathcal{F}_{\text{SFE}}},\mathcal{A}).$$

where $\Phi^{\mathcal{F}_{\text{SFE}}}$ is the "dummy" $\mathcal{F}_{\text{SFE}}$-hybrid protocol.

[18] We note that, usually, dummy protocols are only defined in the UC framework. However, [6] also allows for hybrid protocols, and it is natural to introduce it for the following definition.

Remark 2. We point out that depending on the actual values of the vector $\vec{\gamma}$ and the corruption-cost function, there might not exist an ideally fair protocol for computing certain functions. Nonetheless, as we show in Section 4.2, for a natural class of cost functions (which roughly require that the cost increases linearly with the number of corrupted parties) we are able to provide ideally fair multi-party protocols for computing an arbitrary function f (in the presence of an adversary who is allowed to corrupt arbitrarily many parties). In contrast, the existence of an optimally fair protocol is always guaranteed by the Minimax theorem, and the fact that such a protocol is a solution to the zero-sum (attack) game as defined in the RPD framework.

Moreover, we introduce the following definition which allows to compare cost functions. In Theorem 6, we show that utility-balanced fairness corresponds to security with the lowest-possible cost function.

Definition 20. Let $\mathcal{C}, \mathcal{C}' : [n] \to \mathbb{R}$ be arbitrary cost functions. We say that $\mathcal{C}$ *(weakly) dominates* $\mathcal{C}'$ if for every $t \in [n]$, $\mathcal{C}(t) \overset{\text{negl}}{\geq} \mathcal{C}'(t)$. We say that $\mathcal{C}$ *strictly dominates* $\mathcal{C}'$ if for every $t \in [n]$, $\mathcal{C}(t) \overset{\text{negl}}{>} \mathcal{C}'(t)$.

Most natural multi-party protocols are symmetric, such that the adversary's payoff will depend only on the number of corrupted parties (but not on the actual parties that are corrupted). Instead of indirectly expressing this fact via the cost of corrupting parties, we can take a more direct approach and measure the payoff for each number t of corrupted parties explicitly. Note that the notion of optimality transfers, but the comparison induces only a partial order.

Definition 21. Let $\vec{\gamma} \in \mathbb{R}^d \in \Gamma_{\text{fair}}$ be a payoff vector. For a function $\phi : [n-1] \to \mathbb{R}$, we say that a protocol Π is *ϕ-fair with respect to $\vec{\gamma}$* if for all adversaries $\mathcal{A}_t$ corrupting t parties,

$$u_{\mathtt{A}}(\Pi, \mathcal{A}_t) \overset{\text{negl}}{\leq} \phi(t).$$

Utility-balanced fairness can be defined in terms of the functions ϕ described in Definition 21: for a utility-balanced fair protocol, the term $\sum_{t=1}^{n-1} \phi(t)$ is minimized. In the following, we describe how the notion of being ϕ-fair corresponds to ideal $\vec{\gamma}^{\mathcal{C}}$-fairness with respect to a particular cost function $\mathcal{C}$. We generally write $s(t)$ for the payoff of the best adversary corrupting t parties when executing with the ideal functionality $\mathcal{F}$, i.e., $s(t) = \max_{\mathcal{A}_t} u_{\mathtt{A}}(\Phi^{\mathcal{F}}, \mathcal{A}_t)$ where we maximize over all adversaries $\mathcal{A}_t$ corrupting up to t parties.

Lemma 22. *Let $\phi : [n-1] \to [0,1]$ be a function and let Π be a protocol, and let $\vec{\gamma} = (\gamma_1, \ldots, \gamma_d) \in \Gamma_{\text{fair}}$ be a payoff vector. The following two statements are equivalent, with respect to* static[19] *adversaries:*

- *Π is ϕ-fair with respect to $\vec{\gamma}$.*
- *Π is ideally $\vec{\gamma}^{\mathcal{C}}$-fair, where $\vec{\gamma}^{\mathcal{C}} = (\gamma_1, \ldots, \gamma_d, \{-c(|\mathcal{I}|)\}_{\mathcal{I} \subseteq \mathcal{P}})$, i.e., we have the cost function $\mathcal{C}(\mathcal{I}) = c(|\mathcal{I}|) = \phi(|\mathcal{I}|) - s(|\mathcal{I}|)$.*

[19] The property of being ϕ-secure is defined only for adversaries which corrupt a predetermined number of parties; hence, the result actually holds for all adversaries where this number is fixed (but the choice of parties may be adaptive). Furthermore, one can define protocols in which the "optimal" number of parties to corrupt is determined only during the protocol run and becomes known to the adversary; in that case a ϕ-secure protocol may not be attack-payoff secure, with the described cost functions, against fully adaptive adversaries.

Proof. "$\Longrightarrow$": If Π is ϕ-fair, then by Definition 21, $u_{\mathtt{A}}(\Pi, \mathcal{A}_t) \overset{\text{negl}}{\leq} \phi(t)$ for all adversaries $\mathcal{A}_t$ corrupting t parties. If we then consider the vector $\vec{\gamma}^{\mathcal{C}} = (\gamma_0, \ldots, \gamma_d, \{-c(|\mathcal{I}|)\}_{\mathcal{I}\subseteq\mathcal{P}})$,

$$\hat{U}^{\Pi,\langle\mathcal{F}\rangle,\vec{\gamma}^{\mathcal{C}}}(\mathcal{A}_t) \;=\; \hat{U}^{\Pi,\langle\mathcal{F}\rangle,\vec{\gamma}}(\mathcal{A}_t) - c(t) \;\overset{\text{negl}}{\leq}\; \phi(t) - (\phi(t) - s(t)) \;\leq\; s(t) \;\overset{\text{negl}}{\approx}\; \max_{\mathcal{A}_t} \hat{U}^{\Phi^{\mathcal{F}},\langle\mathcal{F}\rangle,\vec{\gamma}}(\mathcal{A}_t),$$

which means that the payoff that the adversary achieves against the protocol is at most the same as against the ideal functionality, so Π is ideally $\vec{\gamma}^{\mathcal{C}}$-fair. Note that we used the notation from equation (2) because the adversary's utility is defined differently, which appears in the terms as $\vec{\gamma}^{\mathcal{C}}$ and $\vec{\gamma}$, respectively.
"$\Longleftarrow$": If Π is ideally $\vec{\gamma}^{\mathcal{C}}$-fair, then this means that $\hat{U}^{\Pi,\langle\mathcal{F}\rangle,\vec{\gamma}^{\mathcal{C}}}(\mathcal{A}_t) \leq s(t)$, or

$$\hat{U}^{\Pi,\langle\mathcal{F}\rangle,\vec{\gamma}}(\mathcal{A}_t) \;=\; \hat{U}^{\Pi,\langle\mathcal{F}\rangle,\vec{\gamma}^{\mathcal{C}}}(\mathcal{A}_t) + c(t) \;\overset{\text{negl}}{\leq}\; s(t) + c(t) \;=\; \phi(t),$$

which immediately means that Π is ϕ-fair. □

We are now ready to prove Theorem 6 from the main body, which we restate here.

Theorem 6. *Let $\vec{\gamma} = (\gamma_{00}, \gamma_{01}, \gamma_{10}, \gamma_{11}) \in \Gamma^{+}_{\text{fair}}$. 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 $\vec{\gamma}^{\mathcal{C}} = (\gamma_{00}, \gamma_{01}, \gamma_{10}, \gamma_{11}, -\{\mathcal{C}(\mathcal{I})\}_{\mathcal{I}\subseteq\mathcal{P}}) \in \Gamma^{+\mathcal{C}}_{\text{fair}}$ for the function $c(t) = u_{\mathtt{A}}(\Pi, \mathcal{A}_t)$, where $\mathcal{A}_t$ is the best adversary strategy corrupting up to t parties.*
2. *The cost function $\mathcal{C}$ 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 $\mathcal{C}$ according to Definition 20.*

Proof. The first statement actually holds for arbitrary protocols Π and is a direct consequence of Lemma 22. The second statement can be seen as follows: Let Π' be a protocol and $\mathcal{C}'$ be a cost function that is strictly dominated by $\mathcal{C}$. Then, again by Lemma 22, we obtain that $u_{\mathtt{A}}(\Pi', \mathcal{A}'_t) \overset{\text{negl}}{\leq} u_{\mathtt{A}}(\Pi, \mathcal{A}_t)$, where the inequality is strict for at least one $t \in [n-1]$. But then, the sum of these terms over all such t is strictly smaller for Π', which contradicts the optimality of Π according to Definition 5. □

C Utility-based vs. Partial Fairness (cont'd)

A notion that is related to our fairness definition is the concept of $1/p$-security introduced by Gordon and Katz [18] for the two-party setting. Roughly speaking, the notion allows a noticeable error of at most $1/p$ (for a polynomial p) in the security statement for a protocol.

In this section, we weaken our definition by modifying the functionality $\mathcal{F}^{f,\perp}_{\text{SFE}}$ to allow for a correctness error; specifically, functionality $\mathcal{F}^{f,\$}_{\text{SFE}}$ allows the adversary to replace the honest party's output by a randomly chosen one. Intuitively, the ability of the simulator to use this capability with probability $1/p$ corresponds to contradicting the full security definition in the manner the protocols described in [18] do.

We then show that even this weakened notion is *strictly* stronger than $1/p$-security—even if the latter notion is strengthened by additionally requiring "full privacy" as suggested in [18]—by providing a protocol that leaks honest inputs, but can still be proven to achieve both conditions of [18]. We show, however, that the protocols described in [18] do achieve the weakened notion of security introduced in this section, which strengthens the result of [18].

C.1 Description of $1/p$-Security

The fundamental idea underlying $1/p$-security can be described easily: If the implemented functionality outputs a value to a (dishonest) party such that the set of all possible outputs is polynomial, then a value chosen uniformly at random from that set will meet the actual output with noticeable probability. Hence, if the party receives a long sequence of random values (this sequence will contain the actual output by chance already several times), and have this sequence switch to be constant with the actual output after a random number of steps, then the dishonest party will not be able to recognize the switch at the exact point (but only later). This "delay" is exploited to also provide the correct value to the other party.

The protocols described in [18] follow this intuition: The two parties compute a functionality (using an unfair protocol) by which they obtain a sequence of commitments to the above described values—for each round of the following protocol, there is one commitment per party, such that always the other party has the opening information. One party is chosen to always go first and send the opening information to the other party, which opens the commitment and responds with the corresponding opening information. As soon as one party aborts, the other party outputs the last value obtained by opening a commitment.

The actual security definition is two-fold:

- $1/p$-secure computation, which means

$$\forall \mathcal{A}\ \exists \mathcal{S}:\ \left\{\mathrm{IDEAL}_{\mathcal{F},\mathcal{S}(\mathsf{aux})}(x,y,n)\right\}_{(x,y)\in X\times y,\mathsf{aux}\in\{0,1\}^*,n\in\mathbb{N}} \overset{1/p}{\approx} \left\{\mathrm{REAL}_{\Pi,\mathcal{A}(\mathsf{aux})}(x,y,n\right\}_{(x,y)\in X\times y,\mathsf{aux}\in\{0,1\}^*,n\in\mathbb{N}}.$$

- privacy, which is defined as

$$\forall \mathcal{A}\ \exists \mathcal{S}:\ \left\{\mathrm{OUT}^{\mathcal{S}}_{\mathcal{F},\mathcal{S}(\mathsf{aux})}\right\}_{(x,y)\in X\times Y,\mathsf{aux}\in\{0,1\}^*,n\in\mathbb{N}} \equiv \left\{\mathrm{VIEW}^{\mathcal{A}}_{\Pi,\mathcal{A}(\mathsf{aux})}(x,y,n)\right\}_{(x,y)\in X\times y,\mathsf{aux}\in\{0,1\}^*,n\in\mathbb{N}}.$$

The two main differences with our definition show up in that the second condition does not include correctness (so this is only guaranteed with noticeable error), and in that the simulators in the two conditions can be chosen independently, which relaxes the privacy condition for certain classes of functions.

C.2 The Functionality with Randomized Abort

The protocols introduced in [18] intuitively achieve the following guarantee: with probability (at least) $1-1/p$, the protocol execution is fair in the sense that either both parties obtain the correct output or both parties obtain a random output, and with the remaining probability, the honest party obtains a random output while the corrupted party obtains the correct output. This is formalized with the help of the functionality $\mathcal{F}^{f,\$}_{\mathrm{SFE}}$ described in Figure 1.

C.3 Analyzing the Gordon-Katz Protocols

Gordon and Katz [18] propose two different protocols: One for functions that have (at least) one domain of (at most) polynomial size, and one for functions in which both domains might be large, but (at least) one range is at most polynomial. We first extend the proof of the protocol for

Functionality $\mathcal{F}_{\text{SFE}}^{f,\$}$ (Computation with random abort, two-party case)

The functionality is parametrized by two families of distributions $Y_i(\cdot)$ on $\mathcal{Y}_i$ with parameter in $\mathcal{X}_i$, for $i \in \{1,2\}$

- Obtain $(\texttt{input}, x_i)$ from p_i (for $i \in \{1,2\}$), and send $(\texttt{input}, p_i)$ to $\mathcal{A}$.
- After both p_1 and p_2 have provided input, compute $(y_1, y_2) \leftarrow f(x_1, x_2)$ and provide y_1 as a private delayed output to p_1, and y_2 as a private delayed output to p_2.
- Upon receiving $(\texttt{corrupt}, p_i)$ from $\mathcal{A}$, send x_i to $\mathcal{A}$.
- If p_i is corrupted and $\mathcal{A}$ inputs $(\texttt{abort})$ while $y_{\neg i}$ has not yet been delivered, set $y_{\neg i} \xleftarrow{\$} Y_{\neg i}(x_{\neg i})$.

Figure 1: The functionality with randomized abort

functions with polynomial domain (see [18, Section 3.2]) to our model. In particular, we prove that the protocol realizes the functionality $\mathcal{F}_{\text{SFE}}^{f,\$}$ formally described in Figure 1 on page 27. Intuitively, this functionality states that in case of an "unfair" adversarial abort, the adversary's input is replaced by a value chosen at random from some distribution that only depends on the honest party's input.

We prove the strengthened result for the protocols with polynomial domain. In that case, the distribution $Y_i(x_i)$ is defined via $Y_1(x_1) := f(x_1, X_2)$ (resp. $Y_2(x_2) := f(X_1, x_2)$) with X_2 (resp. X_1) uniformly random.

Theorem 23. *Let $f = \{f_n : X_n \times Y_n \to Z_n\}$ be a (randomized) functionality where $|Y_n| = \mathsf{poly}(n)$. For any polynomial p there is an $\mathcal{O}(p \cdot |Y_n|)$-round $\mathsf{ShareGen}$-hybrid protocol implementing $\mathcal{F}_{\text{SFE}}^{f,\$}$ such that $\bar{u}_{\mathtt{A}}(\Pi, \mathcal{A}) \overset{\text{negl}}{\leq} 1/p$.*

Proof sketch. We begin by describing a simulator $\mathcal{S}$ such that $\text{REAL}_{\Pi,\mathcal{A},\mathcal{Z}} \overset{\text{negl}}{\approx} \text{IDEAL}_{\mathcal{F}_{\text{SFE}}^{f,\$},\mathcal{S},\mathcal{Z}}$ and $U_I^{\mathcal{F}_{\text{SFE}}^{f,\$},\vec{\gamma}}(\mathcal{S}, \mathcal{Z}) \leq 1/p$ for all $\mathcal{Z}$ analogously to the simulator described in the proof of [18, Theorem 3]. $\mathcal{S}$ initially simulates the messages sent by the parties alternatingly: $\mathcal{S}$ starts by choosing the round $1 \leq i^* \leq r$ and draws the MAC keys k_a and k_b uniformly at random, the simulated messages are pairs of random shares and MAC tags. As soon as the adversary corrupts a party, $\mathcal{S}$ sends a corresponding message to $\mathcal{F}_{\text{SFE}}^{f,\$}$ and obtains that party's input. Given these values, $\mathcal{S}$ can easily simulate the internal state of the corrupted party similarly to the messages, as described in [18, Theorem 3].

If the adversary aborts in round $i < i^*$, then $\mathcal{S}$ sends $(\texttt{abort})$ to $\mathcal{F}_{\text{SFE}}^{f,\$}$, essentially replacing the dishonest party's input with a randomly chosen one. Here, $\mathcal{S}$ does not retrieve the dishonest party's output. If the adversary aborts in round $i > i^*$, the adversary tells $\mathcal{F}_{\text{SFE}}^{f,\$}$ to deliver the correct output (and also retrieves the correct output of the dishonest party). If, however, p_1 is corrupted and the adversary aborts in step i^*, then $\mathcal{S}$ sends $(\texttt{abort})$ to $\mathcal{F}_{\text{SFE}}^{f,\$}$, thereby replacing the dishonest party's input with a randomly chosen one, and *does* retrieve the dishonest party's output. The distinguishing advantage between the real and the ideal setting is at most negligible and comprises the probability of forging a MAC; all other values can be simulated perfectly.

By [18, Lemma 2], we can conclude $U_I^{\mathcal{F}_{\text{SFE}}^{f,\$},\vec{\gamma}}(\mathcal{S}, \mathcal{Z}) \leq 1/p$ exactly as in [18, Theorem 3]: $\mathcal{S}$ will trigger the event E_{10} only if $\mathcal{Z}$ makes the protocol abort in round i^* while p_1 is corrupted, but an environment $\mathcal{Z}$ that guesses this round correctly can be used as an adversary winning the game in the lemma. □

The protocol for functionalities with polynomial size range is different, we adapt the following theorem from [18, Section 3.3]. The proof is adapted from the original one [18, Theorem 6] exactly as in the case of Theorem 23.

Theorem 24. *Let $f : \{f_n : X_n \times Y_n \to Z_n^1 \times Z_n^2\}$ be a (randomized) functionality, where $|Z_n^1| = \mathsf{poly}(n)$. For any polynomial p there is an $\mathcal{O}(p^2 \cdot |Z_n^1|)$-round $\mathsf{ShareGen}$-hybrid protocol implementing $\mathcal{F}_{\text{SFE}}^{f,\ell}$ such that $\bar{u}_{\mathtt{A}}(\Pi, \mathcal{A}) \overset{\text{negl}}{\leq} 1/p$.*

C.4 Utility-based Fairness Implies $1/p$-Security (cont'd)

For an appropriately chosen vector $\vec{\gamma}$, we can show that a security statement in our model implies the notion of $1/p$-security. In particular, we choose $\vec{\gamma} = (0, 0, 1, 0)$ to show the following lemma.

Lemma 25. *If for a protocol Π it holds that for $\vec{\gamma} = (0, 0, 1, 0)$ and for any adversary $\mathcal{A}$,*

$$\bar{u}_{\mathtt{A}}(\Pi, \mathcal{A}) \quad \leq \quad 1/p,$$

then there is a polynomial p' (such that $1/p' \overset{\text{negl}}{\leq} 1/p$) and Π is $1/p'$-secure and private.

Proof. We show the two conditions independently.

We first use the assumption that $\bar{u}_{\mathtt{A}}(\Pi, \mathcal{A}) \leq 1/p$, which means that there is a simulator $\mathcal{S}$ such that $\text{REAL}_{\Pi,\mathcal{A},\mathcal{Z}} \overset{\text{negl}}{\approx} \text{IDEAL}_{\mathcal{F}_{\text{SFE}}^{f,\$},\mathcal{S},\mathcal{Z}}$ and $U_I^{\mathcal{F}_{\text{SFE}}^{f,\$},\vec{\gamma}}(\mathcal{S}, \mathcal{Z}) \leq 1/p$ for all $\mathcal{Z}$. From $\mathcal{S}$, we obtain a simulator $\mathcal{S}'$ for the fully fair functionality as follows: $\mathcal{S}'$ behaves exactly as $\mathcal{S}$ except for the (`abort`)-queries and the potentially following ones to replace the output (at the functionality $\mathcal{F}_{\text{SFE}}^{f,\$}$). $\mathcal{S}'$ first ignores these queries, but stops once $\mathcal{S}$ both requested from $\mathcal{F}_{\text{SFE}}^{f,\$}$ the dishonest party's output and issued an (`abort`)-query. (The above condition makes sure that this happens with probability at most $1/p$, as it would imply the event E_{10}.) We conclude $\text{IDEAL}_{\mathcal{F}_{\text{SFE}}^{f,\$},\mathcal{S}',\mathcal{Z}} \equiv \text{IDEAL}_{\mathcal{F}_{\text{SFE}}^{f},\mathcal{S}',\mathcal{Z}}$ since $\mathcal{S}'$ never aborts $\mathcal{F}_{\text{SFE}}^{f,\$}$ and

$$\left|\Pr\left[\text{IDEAL}_{\mathcal{F}_{\text{SFE}}^{f,\$},\mathcal{S}',\mathcal{Z}} = 1\right] - \Pr\left[\text{IDEAL}_{\mathcal{F}_{\text{SFE}}^{f,\$},\mathcal{S},\mathcal{Z}} = 1\right]\right| \quad \leq \quad U_I^{\mathcal{F}_{\text{SFE}}^{f,\$},\vec{\gamma}}(\mathcal{S}, \mathcal{Z})$$

since the behavior of $\mathcal{S}$ and $\mathcal{S}'$ only differs only if $\mathcal{S}$ both issues an (`abort`) query and requests the dishonest party's output, which corresponds to the event E_{10}.

Assume (toward a contradiction) that for all p' with $1/p' \overset{\text{negl}}{\leq} 1/p$ it holds that Π is *not* $1/p'$-secure. In particular, there exist inputs $x \in X$ and $y \in Y$, auxiliary inputs $\mathsf{aux} = (\mathsf{aux}_n)_{n\geq 1}$, $\mathsf{aux}_n \in \{0,1\}^*$, an adversary $\mathcal{A}$, such that for all simulators $\mathcal{S}$, the random variable ensembles $\left\{\text{IDEAL}_{\mathcal{F},\mathcal{S}(\mathsf{aux})}(x, y, n)\right\}_{n\in\mathbb{N}}$ and $\left\{\text{REAL}_{\Pi,\mathcal{A}(\mathsf{aux})}(x, y, n)\right\}_{n\in\mathbb{N}}$ are distinguished by some distinguisher $\mathcal{D}$ with advantage $\varepsilon \geq 1/p + 1/q_{\mathcal{S}}$ for some polynomial $q_{\mathcal{S}}$.

We now use the environment $\mathcal{Z}_{x,y,\mathsf{aux},\mathcal{D}}$ that inputs x at the interface of p_1, y at the interface of p_2, and then admits the execution following the instruction of $\mathcal{A}(\mathsf{aux})$ until it finishes, and records the outputs of the party. Then it compiles the tuple of outputs as expected by $\mathcal{D}$, runs $\mathcal{D}$, and uses its output. By the assumption on $\mathcal{D}$,

$$\left|\Pr\left[\text{REAL}_{\Pi,\mathcal{A},\mathcal{Z}_{x,y,\mathsf{aux},\mathcal{D}}} = 1\right] - \Pr\left[\text{IDEAL}_{\mathcal{F}_{\text{SFE}}^{f,\$},\mathcal{S}',\mathcal{Z}_{x,y,\mathsf{aux},\mathcal{D}}} = 1\right]\right| \quad \geq \quad 1/p + 1/q_{\mathcal{S}'}.$$

But, on the other hand,

$$\left|\Pr\left[\text{IDEAL}_{\mathcal{F}_{\text{SFE}}^{f,\$},\mathcal{S}',\mathcal{Z}_{x,y,,\mathsf{aux},\mathcal{D}}} = 1\right] - \Pr\left[\text{IDEAL}_{\mathcal{F}_{\text{SFE}}^{f,\$},\mathcal{S},\mathcal{Z}_{x,y,\mathsf{aux},\mathcal{D}}} = 1\right]\right| \quad \leq \quad 1/p$$

by the assumption on $\mathcal{S}$.

Using the triangle inequality and the above two bounds, we obtain

$$\left|\Pr\left[\text{REAL}_{\Pi,\mathcal{A},\mathcal{Z}_{x,y,\mathsf{aux},\mathcal{D}}}=1\right]-\Pr\left[\text{IDEAL}_{\mathcal{F}_{\text{SFE}}^{f,\$},\mathcal{S},\mathcal{Z}_{x,y,\mathsf{aux},\mathcal{D}}}=1\right]\right|\geq 1/p'-1/p \quad\geq\quad 1/q_{\mathcal{S}'},$$

in contradiction to $\mathcal{S}\in\texttt{SIM}_{\mathcal{A}}$.

Secondly, we assume that Π does not achieve privacy which means that there is a pair of inputs $(x,y)\in X\times Y$, and auxiliary inputs $\mathsf{aux}=(\mathsf{aux}_n)_{n\geq 1}$, $\mathsf{aux}_n\in\{0,1\}^*$, an adversary $\mathcal{A}$, such that for all simulators $\mathcal{S}$ the probability ensembles $\left\{\text{OUT}^{\mathcal{S}}_{\mathcal{F},\mathcal{S}(\mathsf{aux})}(x,y,n)\right\}_{n\in\mathbb{N}}$ and $\left\{\text{VIEW}^{\mathcal{A}}_{\Pi,\mathcal{A}(\mathsf{aux})}(x,y,n)\right\}_{n\in\mathbb{N}}$ are distinguished by some distinguisher $\mathcal{D}$ with advantage $\varepsilon\geq 1/q_{\mathcal{S}}$ for some polynomial $q_{\mathcal{S}}$.

We use the environment $\mathcal{Z}'_{x,y,\mathsf{aux},\mathcal{D}}$ constructed as above (with the only difference that the input to $\mathcal{D}$ is constructed differently). Let $\mathcal{A}$ be an adversary. From a simulator $\mathcal{S}$ in the setting with $\mathcal{F}_{\text{SFE}}^{f,\$}$ we define a simulator $\mathcal{S}''$ that behaves exactly as $\mathcal{S}$ except for blocking all queries that either abort the functionality or replace the honest party's output. Note that we have

$$\text{OUT}^{\mathcal{S}}_{\mathcal{F}_{\text{SFE}}^{f,\$},\mathcal{S}(\mathsf{aux})} \quad\equiv\quad \text{OUT}^{\mathcal{S}'}_{\mathcal{F}_{\text{SFE}}^{f,\$},\mathcal{S}'(\mathsf{aux})} \quad\equiv\quad \text{OUT}^{\mathcal{S}'}_{\mathcal{F}_{\text{SFE}}^{f},\mathcal{S}'(\mathsf{aux})},$$

which means that

$$\begin{aligned}&\left|\Pr\left[\text{REAL}_{\Pi,\mathcal{A},\mathcal{Z}'_{x,y,\mathsf{aux},\mathcal{D}}}=1\right]-\Pr\left[\text{IDEAL}_{\mathcal{F}_{\text{SFE}}^{f,\$},\mathcal{S},\mathcal{Z}'_{x,y,\mathsf{aux},\mathcal{D}}}=1\right]\right|\\ &\quad=\quad\left|\Pr\left[\mathcal{D}\left(\text{VIEW}^{\mathcal{A}}_{\Pi,\mathcal{A}(\mathsf{aux})}(x,y)\right)=1\right]-\Pr\left[\mathcal{D}\left(\text{OUT}^{\mathcal{S}}_{\mathcal{F}_{\text{SFE}}^{f},\mathcal{S}'(\mathsf{aux})}(x,y)\right)=1\right]\right|\quad\geq\quad 1/q_{\mathcal{S}'},\end{aligned}$$

which in particular means $\mathcal{C}_{\mathcal{A}}=\emptyset$ in contradiction to the assumption. □

C.5 Separating Utility-based Fairness from $1/p$-Security

In the opposite direction, we construct a "leaky" protocol $\tilde{\Pi}$ that is insecure with respect to the definition based on $\mathcal{F}_{\text{SFE}}^{f,\$}$, but still satisfies the definition of $1/p$-security. We consider the particular case of computing the logical "and": $\wedge:\{0,1\}\times\{0,1\}\to\{0,1\}$, and set $p=2$. The protocol is described as follows, with inputs x_1 and x_2 for p_1 and p_2, respectively:

- The first message is a 0-bit that is sent from p_2 to p_1.
- Yet, 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 the standard $\frac{1}{4}$-secure protocol to compute $x_1\wedge x_2$.

In particular, we show that the protocol does not even realize the "relaxed" functionality.

Lemma 26. *The protocol does not implement $\mathcal{F}_{\text{SFE}}^{f,\$}$.*

Proof sketch. We consider the straightforward environments $\mathcal{Z}_1$ and $\mathcal{Z}_2$ such that at least one distinguishes with good probability as follows: $\mathcal{Z}_i$ chooses the input $x_1\in\{0,1\}$ uniformly at random, immediately corrupts p_2 and sends a 1-bit. Afterward, $\mathcal{Z}_i$ admits the complete protocol execution (running the honest protocol with input $x_2=0$ for p_2) and obtains the output z_1.
The output of $\mathcal{Z}_1$ is computed as follows:

- If (in the first round) p_1 sends the correct input x_1 to p_2, and $z_1=0$, then $\mathcal{Z}_1$ outputs 1.
- If p_1 sends $1-x_1$, or does not send its input at all, or $z_1=1$, then $\mathcal{Z}_1$ outputs 0.

The output of $\mathcal{Z}_2$ is computed as follows:

- If (in the first round) p_1 sends some non-empty message $\tilde{x} \in \{0,1\}$ to p_2, then $\mathcal{Z}_2$ outputs 1.
- If p_1 sends an empty message, then $\mathcal{Z}_2$ outputs 0.

It is easy to compute

$$\Pr\left[\text{REAL}_{\tilde{\Pi},\mathcal{A},\mathcal{Z}_1} = 1\right] \quad = \quad \Pr\left[\text{REAL}_{\tilde{\Pi},\mathcal{A},\mathcal{Z}_2} = 1\right] = 1/4,$$

and we also show

$$\Pr\left[\text{IDEAL}_{\mathcal{F}_{\text{SFE}}^{\wedge,\$},\mathcal{S},\mathcal{Z}_1} = 1\right] \quad \leq \quad \frac{3}{4} \cdot \Pr\left[\text{IDEAL}_{\mathcal{F}_{\text{SFE}}^{\wedge,\$},\mathcal{S},\mathcal{Z}_2} = 1\right].$$

This holds as $\mathcal{S}$ cannot observe the correct value x_1 *without* potentially allowing for $z_1 = 1$. For the following arguments, we only consider those runs where $\mathcal{S}$ indeed simulates the "first round reply" of p_1 (which must happen in approximately a $\frac{1}{4}$-fraction of the cases, as otherwise $\mathcal{Z}_2$ distinguishes).

- For those cases where $\mathcal{S}$ inputs $x_2 = 0$ (or does not obtain p_2's output before simulating the "first round reply"), the message $\tilde{x}$ is statistically independent of the uniformly random input x_1, so $\mathcal{Z}_2$ would output 1 "about twice as often as" $\mathcal{Z}_1$ (which does so only if $\tilde{x} = x_1$).
- For those cases where $\mathcal{S}$ inputs $x_2 = 1$, the output that $\mathcal{S}$ obtains would be $y_1 = y_2 = 1$ in approximately half the cases (namely if $x_1 = 1$). In that case, the best $\mathcal{S}$ can do is abort and re-randomize y_1, but as $x_1 = 1$ this still means that $y_1 = 1$ in a quarter of the cases where $\mathcal{S}$ inputs $x_2 = 1$. Overall, $\mathcal{Z}_2$ would output 1 "about $\frac{4}{3}$ times as often as" $\mathcal{Z}_1$ (which does so only of $y_1 = 0$).

Since there is a non-negligible gap between the probabilities w.r.t $\mathcal{Z}_1$ and $\mathcal{Z}_2$ in the ideal case, while the probabilities are the same in the real case, at least one of the environments $\mathcal{Z}_1$ and $\mathcal{Z}_2$ must be successful. □

In the following lemma, we show that the protocol is both $\frac{1}{2}$-secure and private in the sense of [18, Definition 1] and [18, Definition 14].

Lemma 27. *The protocol $\tilde{\Pi}$ is both $\frac{1}{2}$-secure and private.*

Proof sketch. We first sketch that the protocol achieves $\frac{1}{2}$-security. (The only "interesting" case is where p_2 is corrupted and $\mathcal{A}$ sends as the first message a 1-bit; otherwise the protocol is even $\frac{1}{4}$-secure.) For this, we describe a simulator $\mathcal{S}$ that first draws a biased random bit C with $\Pr[C = 1] = \frac{1}{4}$, and simulates a uniformly random bit in the name of p_1 if $C = 1$. Then, it uses the "standard" $\frac{1}{4}$-security simulator for the second stage of the protocol. The distinguishing advantage for the second stage alone is bounded by $\frac{1}{4}$, and with probability at least $\frac{7}{8}$ the simulator for the first stage is perfect, which concludes the first part of the proof sketch.

Second, we show that the protocol is private. Again, the only interesting case is where p_2 is corrupted and $\mathcal{A}$ initiates the protocol with a 1-bit. In this case, the simulator can replace p_2's input to be $y' = 1$ and can obtain the input x of p_1. Then, it can faithfully simulate the protocol run using the input x of the honest party. □