Secure Two-Party Computation with Fairness - A Necessary Design Principle

Yehuda Lindell[1(✉)] and Tal Rabin[2]

[1] Deptartment of Computer Science, Bar-Ilan University, Ramat Gan, Israel
lindell@biu.ac.il
[2] IBM T.J. Watson Research Center, New York, USA
talr@us.ibm.com

Abstract. Protocols for secure two-party computation enable a pair of mutually distrustful parties to carry out a joint computation of their private inputs without revealing anything but the output. One important security property that has been considered is that of *fairness* which guarantees that if one party learns the output then so does the other. In the case of two-party computation, fairness is not always possible, and in particular two parties cannot fairly toss a coin (Cleve, 1986). Despite this, it is actually possible to securely compute many two-party functions with fairness (Gordon et al., 2008 and follow-up work). However, *all* known two-party protocols that achieve fairness have the unique property that the effective input of the corrupted party is determined at an arbitrary point in the protocol. This is in stark contrast to almost all other known protocols that have an explicit fixed round at which the inputs are *committed*.

In this paper, we ask whether or not the property of not having an input committal round is *inherent* for achieving fairness for two parties. In order to do so, we revisit the definition of security of Micali and Rogaway (Technical report, 1992), that explicitly requires the existence of such a committal round. We adapt the definition of Canetti in the two-party setting to incorporate the spirit of a committal round, and show that under such a definition, it is *impossible* to achieve fairness for any non-constant two-party function. This result deepens our understanding as to the type of protocol construction that is needed for achieving fairness. In addition, our result discovers a fundamental difference between the definition of security of Micali and Rogaway and that of Canetti (Journal of Cryptology, 2000) which has become the standard today. Specifically, many functions can be securely computed with fairness under the definition of Canetti but no non-constant function can be securely computed with fairness under the definition of Micali and Rogaway.

Keywords: Secure two-party computation · Fairness · Definitions of security

Y. Kalai and L. Reyzin (Eds.): TCC 2017, Part I, LNCS 10677, pp. 565–580, 2017.
https://doi.org/10.1007/978-3-319-70500-2_19

1 Introduction

In the setting of secure two-party computation, a pair of parties P_1 and P_2 wish to compute a joint function of their private inputs in a secure manner. The standard requirements for security are *privacy* (meaning that nothing but the output is revealed), *correctness* (meaning that the output is correctly computed) and *independence of inputs* (meaning that a corrupted party cannot make its input dependent on the honest party's input). An additional property that is highly desired in many applications is that of *fairness*, which guarantees that corrupted parties cannot receive output without the honest parties also receiving output. The fundamental question regarding the feasibility of achieving fairness has been studied since the late 1980 s starting with the seminal work of Cleve [14] who showed that it is *impossible* for two parties to securely toss an unbiased coin. Following this work, a folklore arose that assumed that essentially no interesting function can be securely computed with fairness. Intuitively, this makes sense since in order for two parties to compute the function by exchanging messages in turn, one must at some stage know more information than the other. As a result, *partial* notions of fairness were introduced, including gradual release [7] and the optimistic model that utilizes a trusted third party [2,22] (these models are not relevant to our work here that focuses on achieving full fairness).

However, over two decades later, it was shown by Gordon et al. [20] that it is in fact possible to securely compute some specific functions with (full) fairness. Later, it was shown that it is actually possible to compute *many* finite-domain Boolean functions securely with fairness in the presence of malicious adversaries [3–5,21]. These positive results demonstrate a specific methodology for protocol construction that achieves fair secure two-party computation. In contrast to these positive constructions, there has been very little work regarding *necessary conditions* for achieving fair secure two-party computation. In particular, there is no proof whether this methodology is in fact needed. In this paper, we prove that the central design principle used in all of [3–5,20,21] is in fact *necessary*.

Background. As we have mentioned, the classic definition of security for two-party computation guarantees central properties like privacy, correctness and independence of inputs. The actual security definition formalizes these security properties by comparing a real protocol execution to an ideal world in which an incorruptible trusted party computes the function for the parties [6,9,18,19,23]. In more detail, a protocol is proven secure by presenting an ideal-world simulator machine that interacts with a trusted party, sending the corrupted parties' inputs and receiving back their outputs. The requirement is then that for every real adversary attacking the protocol, the outputs of the adversary and honest parties in a real protocol execution is computationally indistinguishable from the output of the simulator and honest parties in the ideal model. Note that in the ideal model, the honest parties simply send their prescribed inputs to the trusted party and output whatever they receive back. This guarantees all the aforementioned security properties since the only thing that the simulator can do in the ideal

model is modify the corrupted parties' inputs. Since the outputs in the real and ideal executions are indistinguishable, the same holds also for the real secure protocol. Definitions of this type are said to follow the ideal-real model paradigm.

Classically, the literature considers two types of adversaries; semi-honest adversaries who follow the protocol specification but try to learn more than allowed by inspecting the transcript, and malicious adversaries who may follow any arbitrary polynomial-time attack strategy. In this paper, we consider the setting of malicious adversaries.

In the case of secure multiparty computation with an honest majority, the additional property of *fairness* is typically also required. This property guarantees that if the corrupted party receives an output then so does the honest party. However, in the case of two-party computation – where there is no honest majority and malicious adversaries – fairness is usually not required, since it has been shown that fairness cannot always be achieved. In particular, it was shown in [14] that it is *impossible* for two parties to securely toss an unbiased coin.

The protocol of Gordon et al. [20]. The protocol of [20] and its extensions in [3,4,21] have a very unique property that there is no specific round at which the parties' inputs are "determined". In order to explain this, let us consider for a moment the GMW protocol [17,18]. The GMW protocol begins with the parties running an "input commitment" phase, and then forcing the parties to use these inputs using zero-knowledge proofs. This paradigm of construction is not unique to [18], but rather is the norm in all known protocols.[1] In contrast, in the protocols of [3,4,20,21] which are the *only* protocols that achieve fairness without an honest majority, the corrupted parties' input is essentially determined by the point at which they halt, if they halt before the end. As a result, there is no input-commitment phase, and indeed no point whereby the input of a corrupted party is explicitly fixed. A very interesting question that arises from this is whether or not protocols that achieve fairness *must* work in this way, or if this is just one possible approach.

The Micali-Rogaway (MR) definition of security. The definition of security for multiparty computation, formulated by Micali and Rogaway in [23], is based on the same ideal/real paradigm described above. One of the central differences between the definition of MR and Canetti [9] is the requirement that there exist an explicit committal round, which defines all parties' inputs. In order to understand why this requirement was included, we look back at the motivation provided by Micali and Rogaway for their definition. Micali and Rogaway articulate a number of key ideas for their notion of security (called "key choices" in [23, Sect. 1.6]). The three key choices are *blending privacy and correctness*, *adversarial awareness* and *tight mimicry*. The first two choices are common with the definition of Canetti (that also follows the ideal-real model paradigm), which has today become the standard for secure computation in the stand-alone

[1] In some cases, it is more subtle and the inputs are more implicitly committed; e.g., via oblivious transfer. However, this is still input commitment.

model. The requirement of *blending privacy and correctness* is important since there are examples showing that they cannot actually be separated (an attack on correctness can result in a breach of privacy), and the requirement of *adversarial awareness* means that the adversary's input is explicitly known (formulated by the simulator explicitly providing this input to the trusted party).

In contrast, the requirement of *tight mimicry* was not adopted by others to the same extent. This requirement is that the ideal-model simulation tightly mimics a real protocol execution. Micali-Rogaway observed that in all existing secure protocols, at the time, there was an explicit round whereby the parties commit to their inputs. This was true of the protocols of [8,13,18,24] and almost all protocols known today (with the exception of the fair protocols of [3,4,20,21] and the protocol of [1]), and they mimicked that. As a result they stated that it should be possible to know ***what inputs*** the adversary is using for the corrupted parties, ***when these inputs are determined***, and ***what output*** is received by the corrupted parties. The first and third of these requirements of "tight mimicry" do appear in [9,17], but the second does not. The second requirement is formalized in [23] by requiring the existence of a committal round so that the corrupted parties' inputs are *fully determined* by this round.

In order to understand the committal round requirement in more depth, we informally describe how it is formulated. The MR definition formalizes two different phases of simulation. In the first phase, the simulator simulates up to the committal round and then outputs the inputs of the corrupted parties, supposedly as would be used by the corrupted parties in a real execution. Next, the function is computed on the corrupted parties' inputs as output by the simulator and the honest parties' prescribed inputs. The simulator receives this function's output and continues in the simulation to the end. Note that the simulator interacts with the real adversary as one interacts with an external real party; in particular, this means that the simulator has only black-box access to the adversary and also cannot rewind it. Observe that the aforementioned phases are distinct since the simulation is "straight line".[2] We refer the reader to [23] for more details, and to [15] for a more concise version of the definition.

Importantly, Dodis and Micali [15] inherently utilize the existence of a committal round in order to prove their concurrent composition theorem. Thus, beyond accurately mimicking the way protocols work, this feature of the definition also has technical advantages in the context of composition.

Our results – fairness and committal rounds. Interestingly, the requirement of a committal round rules out the fair protocols of [3,4,20,21], and these protocols cannot be proven secure under any definition with such a requirement, like the MR definition. As we have stated, these are the only protocols that achieve

[2] This makes some aspects of the definition reminiscent of the much-later UC framework [10]; in particular, in [10] the adversarial environment is external to the simulator and the simulator can interact with it as with a real party (meaning, black box and no rewinding). Indeed, in [15] it was shown that protocols proven secure under the MR definition are secure under concurrent composition.

fairness and they all do not have a committal round. A very natural question that arises is therefore whether fair protocols must be designed so that there is no fixed point at which inputs are committed. In particular, we ask:

> *Is it possible to construct two-party protocols with fairness, with the property that the parties' inputs are committed at a fixed committal round?*

Answering this question will also shed light on whether the definition of Canetti is fundamentally different to the MR definition with respect to fairness.

In this paper, we show that the existence of a *committal round* does indeed result in a qualitatively different notion of security. In particular, it is impossible to securely compute any non-constant function in the two-party setting with fairness when there is a committal round. We prove the following theorem:

Theorem 1 (Main theorem – informally stated). *If f is a non-constant function, then it cannot be securely computed in the two-party setting with fairness using a definition that requires a committal round.*

In order to prove the theorem, we adapt the definition of Canetti in a seemingly *minimal way*, to include a committal round conceptually similar to that of MR (with two distinct phases of simulation). Our definition enables rewinding the adversary like Canetti, since otherwise security is not possible without an honest majority (or some secure setup).[3] Our definition suffices for defining security without fairness, and as evidence, *all* non-fair protocols that we know of can be securely computed under our adapted definition.

Our proof of the theorem demonstrates that the effective input of a corrupted party must depend on when it halts. In addition, we show that in a definition with a committal round, the simulator must determine the corrupted party's input at some point before the end of the protocol. This implies that the simulator must determine the corrupted party's input before knowing when it will halt, preventing it from correctly determining the effective input. Thus, simulation is not possible.

Conclusions. Our result deepens our understanding of the type of protocol design needed in order to obtain fairness. Specifically, it is essential that in any fair protocol the input of the corrupted party not be determined at any fixed point. Thus, any protocol that achieves fairness in secure two-party computation must follow the same construction paradigm as [20], at least with respect to the fact that a party's input is not committed at any fixed point.

In addition, our results show that the existence or non-existence of a *required* committal round is not at all inconsequential, and has actual ramifications on the feasibility of achieving security, particularly fairness. This in turn implies that there is actually a fundamental difference between the definitions of Micali-Rogaway and Canetti.

[3] The fact that no rewinding results in impossibility was shown in the framework of universal composability [10] which does not allow rewinding; see [11,12].

2 Defining Secure Two-Party Computation with a Committal Round

In this section, we present a version of the definition of Canetti [9,17] for the two-party case that is minimally adapted to include a committal round like that of MR. As in MR, we formalize the committal round by mandating two distinct phases for the simulation, but we allow rewinding in each phase (as needed for proving the security of two-party protocols). The first phase until the simulator provides the input of the corrupted party, and the second phase from the point that the simulator receives the output to the end of the protocol. As will become clear in the definition below, the simulator may rewind the adversary within each phase but not beyond it, in order to ensure that the phases are indeed distinct.

Our definition below requires fairness, since we aim to show impossibility of fairness when a committal round is included. However, as we will explain at the end of this section, an analogous definition which includes a committal round but *not* fairness is satisfied by almost all protocols that we are aware of; see Theorem 2. Thus, the existence of a committal round alone is not a barrier to achieving security (without fairness).

Preliminaries. We denote the security parameter by n. A function $\mu(\cdot)$ is negligible in n, or just negligible, if for every positive polynomial $p(\cdot)$ and all sufficiently large n's it holds that $\mu(n) < \frac{1}{p(n)}$. We say that two distribution ensembles $X = \{X(a,n)\}_{a\in\{0,1\}^*;n\in\mathbb{N}}$ and $Y = \{Y(a,n)\}_{a\in\{0,1\}^*;n\in\mathbb{N}}$ are computationally indistinguishable if for every non-uniform probabilistic polynomial-time distinguisher D there exists a negligible function $\mu(\cdot)$ such that for every $a \in \{0,1\}^*$ and every $n \in \mathbb{N}$,

$$\left|\Pr[D(X(a,n)=1)] - \Pr[D(Y(a,n)=1)]\right| \leq \mu(n).$$

The real model. In the real model, the two parties P_1 and P_2 interact directly running protocol π, exchanging messages with each other. To be concrete, we assume that in each round of the protocol, one party sends a message to the other party who waits to receive the message (this is the least restrictive and most general model). Each party P_i is given its input x_i and the security parameter n in unary form. We consider a malicious static adversary, $\mathcal{A}$, that controls one of the parties. We denote the corrupted party by P_i ($i \in \{1,2\}$) and denote the honest party by P_j ($j \in \{1,2\}$ with $j \neq i$). The adversary $\mathcal{A}$ is given the corrupted party's input x_i, an auxiliary input z, and the value 1^n (the security parameter in unary) and interacts directly with the honest party P_j. The honest party outputs whatever is prescribed by the protocol, and the corrupted party outputs its *view* in the execution. We denote by $\text{REAL}_{\pi,i,\mathcal{A}}(x_1,x_2,z,n)$ the output of the honest party and the view of the adversary in a real execution, where P_1 has input x_1, P_2 has input x_2, the security parameter is n, and the adversary is given auxiliary input z and controls party P_i.

The ideal model. In the ideal model, the parties do not interact with each other at all. Rather, they just send their input to an incorruptible trusted party who computes the output for them. We consider fairness and therefore the trusted party always sends output to both parties (if the corrupted party does not provide an input, then a default input is taken by the trusted party).[4] The honest party P_j always sends its prescribed input to the trusted party, whereas the corrupted party can send any input it desires or none at all. We denote the ideal-model adversary by $\mathcal{S}$.

Following the MR definition, we define a committal round, which we view as a "break point" in the simulation. Let CR be an integer that denotes the committal round. Our definition is *black-box*, and so the simulator $\mathcal{S}$ is given black-box (oracle) access to the real adversary $\mathcal{A}$ with its input x_i, auxiliary input z and uniformly-distributed random tape r. As formalized in [16, Sec. 4.5], such black-box access is modeled by $\mathcal{S}$ sending oracle queries of the form $q = (m_1, \ldots, m_\ell)$ and receiving back $\mathcal{A}(x_i, z, r; m_1, \ldots, m_\ell)$, where x_i, z, r are as stated and $m_1, \ldots, m_\ell$ is a series of incoming messages (we assume unique delimiters between each item in the query and so it is unambiguous). The response from $\mathcal{A}$ is its outgoing message (or output) when invoked on this input, random-tape and series of messages. We say that an oracle query q is of length ℓ if it contains ℓ incoming messages.

In our definition, $\mathcal{A}$ controls party P_i ($i \in \{1, 2\}$) and works in two distinct phases:

1. *Phase 1 – up to and including the committal round:* In this phase, $\mathcal{S}$ is allowed to send oracle queries of length at most CR only. At the end of this phase, $\mathcal{S}$ outputs a partial view of $\mathcal{A}$ up to and including CR – denoted by $\text{VIEW}^1_{\mathcal{S}^{\mathcal{A}}}(x_i, z, n)$ – and also $\mathcal{A}$'s input x'_i to be sent to the trusted party. The trusted party computes the function output from x'_i and the input received from the honest party, and sends the honest party its specified output.
2. *Phase 2 – post-committal round:* In this phase, $\mathcal{S}$ receives the corrupted party's output from the trusted party, and generates a partial view of $\mathcal{A}$ from the round after CR to the end of the protocol, denoted $\text{VIEW}^2_{\mathcal{S}^{\mathcal{A}}}(x_i, z, n)$.

Note that the "break point" of the simulation is the point *between* the committal round and the round following it. As we have mentioned, the definition of security is black-box (as is the original definition of MR); this seems inherent when formalizing a committal round and break-point.

The output of the ideal execution is the concatenation of $\text{VIEW}^1_{\mathcal{S}^{\mathcal{A}}}(x_i, z, n)$ with $\text{VIEW}^2_{\mathcal{S}^{\mathcal{A}}}(x_i, z, n)$, and the honest party's output. We stress that

[4] Our definition requires guaranteed output delivery (meaning that both parties always receive output), and not just fairness (meaning that if one receives an output then so does the other but it's possible that neither receive). In the setting of two-party computation, these properties are equivalent, since in the case of abort the honest party can compute the function on its own input and on a default input for the other party. We therefore arbitrarily chose the definition where parties always receive output.

$\text{VIEW}^1_{\mathcal{S}^{\mathcal{A}}}(x_i, z, n)$ must contain exactly CR incoming messages; otherwise the output of the ideal execution will be $\perp$. We denote this output by $\text{IDEAL}^{CR}_{f,i,\mathcal{S}^{\mathcal{A}}}(x_1, x_2, z, n)$.

We are now ready to define security:

Definition 1. *Let $f : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^* \times \{0,1\}^*$ be a two-party functionality. A protocol π* securely computes *f* with a committal round and fairness *if there exists a specific round CR and a probabilistic non-uniform polynomial-time simulator $\mathcal{S}$ for the ideal model such that for every probabilistic non-uniform polynomial-time real adversary $\mathcal{A}$ controlling party P_i with $i \in \{1,2\}$:*

$$\left\{\text{IDEAL}^{CR}_{f,i,\mathcal{S}^{\mathcal{A}}}(x_1, x_2, z, n)\right\}_{x_1,x_2,z\in\{0,1\}^*;n\in\mathbb{N}} \stackrel{\text{c}}{\equiv} \left\{\text{REAL}_{\pi,i,\mathcal{A}}(x_1, x_2, z, n)\right\}_{x_1,x_2,z\in\{0,1\}^*;n\in\mathbb{N}}$$

Feasibility of achieving an analogous definition without fairness. We conclude this section by showing that a committal round in itself is not a barrier to achieving security, even for the case of no honest majority, as long as fairness is not also required. In order to see this, consider a modified version of Definition 1 which is exactly the same except that fairness is not guaranteed. In particular, the only difference is that in phase 2, $\mathcal{S}$ receives the corrupted party's output first. Then, if $\mathcal{S}$ sends abort to the trusted party, then the honest party does not receive the output (but rather $\perp$). In contrast, if $\mathcal{S}$ sends continue to the trusted party, then the honest party does receive the actual output. We say that a protocol that achieves this definition is secure with a committal round but without fairness.

By observation of the simulator of the GMW two-party protocol [18] as described in [17, Chap. 7], we have that the simulator indeed can be separated into working in these two different phases. The first simulator works in the "input commitment" phase which essentially consists of each party committing to its input and proving a zero-knowledge proof of knowledge of the committed value. The simulator in this phase extracts the corrupted party's input from the proof of knowledge. Then, the second simulator simulates the rest of the protocol. We therefore have:

Theorem 2. *For every probabilistic polynomial-time two-party functionality f, there exists a protocol π that securely computes f with a committal round but without fairness. In particular, π can be taken as the protocol of [17, 18].*

We remark that the protocol of [17,18] is the rule and not the exception. Indeed, we do not know of *any* protocol that is not secure under this analogous definition that requires a committal round but does not require fairness (with the exception of the *fair* protocols of [20] and its extensions in [3,4,21] since they do not meet the committal round requirement).

3 Proof of Impossibility of Fairness

Theorem 3. *Let f be a non-constant two-party function with a finite domain. Then there does not exist a protocol that securely computes f with a committal round and fairness, as in Definition 1.*

Proof. We will use the notion of a protocol being "honest-correct" in the proof. We stress that this definition of correctness is very different to – and much weaker than – the standard notion. Specifically, when we say that a protocol is honest-correct, we mean that two *honest* parties running the protocol receive the correct output (the function computed on their prescribed inputs), and this does not say anything about the output of the honest party when one of the parties is malicious or halts before the end of the protocol.

Definition 2. *A protocol π for computing a function f is* honest-correct *if for every two inputs x_1, x_2 written on the input tapes of P_1 and P_2, respectively, the output of honest P_1 and honest P_2 running π is $f(x_1, x_2)$, except with negligible probability.*

The proof of the theorem follows immediately from the following two lemmas. The first lemma states that in order to correctly compute the function, then there must be at least one round of communication after the committal round (formally, if there is no round after the committal round, then the protocol cannot be honest-correct). In contrast, the second lemma states that any protocol that is secure with a committal round and fairness can be truncated to the committal round and will still maintain honest-correctness. We therefore derive a contradiction.

Lemma 1. *Let f be a non-constant two-party function, let π be a protocol that securely computes f with a committal round and fairness, and let CR be the index of the committal round. Then, the protocol obtained by truncating π to exactly CR rounds is not honest-correct.*

Proof. This lemma relies on the assumption that f is non-constant, since a constant function can be securely computed without any interaction. We prove the lemma by showing that since the simulator $\mathcal{S}$ receives the output only in the post-committal round phase, and after it outputs the view of $\mathcal{A}$ up to round CR, the view of the adversary in all rounds up to and including the CR is independent of the output. Thus, there must be at least one additional round in the protocol beyond the CR in order to obtain the correct output. Since the function being computed is non-constant, this implies either that the simulation is distinguishable from a real execution (which contradicts the assumed security) or that the protocol is not honest-correct (does not always provide even honest parties with the correct output based on their input). We now prove this formally.

Let π' be the protocol π truncated to round CR (and including round CR). Assume, by contradiction, that π' is honest-correct, as in Definition 2. If f is non-constant then either there exist inputs $x_1, x_2, \tilde{x}_2$ such that $f(x_1, x_2) \neq f(x_1, \tilde{x}_2)$, or there exist inputs $x_1, \tilde{x}_1, x_2$ such that $f(x_1, x_2) \neq f(\tilde{x}_1, x_2)$. This holds since if f is non-constant then there must be either a "row" or "column" in its function matrix with different values. Without loss of generality, assume that there exist $x_1, x_2, \tilde{x}_2$ such that $f(x_1, x_2) \neq f(x_1, \tilde{x}_2)$. Let $\mathcal{A}$ be an adversary attacking the non-truncated protocol π, who controls P_1 and runs P_1 honestly on input x_1,

with the exception that it halts at round CR and outputs whatever the protocol specifies it to output (as if the other party halted). By the contradicting assumption, $\mathcal{A}$ receives correct output by this round (where correct is defined by the honest party's input and by $\mathcal{A}$'s input; this is well defined since $\mathcal{A}$ behaves like an honest party in the truncated protocol π').

Consider a real execution between $\mathcal{A}$ and P_2, where P_2 has input x_2. By the security of the non-truncated protocol π, we have

$$\left\{\text{IDEAL}^{CR}_{f,1,\mathcal{S}^{\mathcal{A}}}(x_1, x_2, z, n)\right\}_{n\in\mathbb{N}} \stackrel{\text{c}}{\equiv} \left\{\text{REAL}_{\pi,1,\mathcal{A}}(x_1, x_2, z, n)\right\}_{n\in\mathbb{N}}.$$

Likewise, in a real execution where P_2 has input $\tilde{x}_2$, the security of the protocol guarantees that

$$\left\{\text{IDEAL}^{CR}_{f,1,\mathcal{S}^{\mathcal{A}}}(x_1, \tilde{x}_2, z, n)\right\}_{n\in\mathbb{N}} \stackrel{\text{c}}{\equiv} \left\{\text{REAL}_{\pi,1,\mathcal{A}}(x_1, \tilde{x}_2, z, n)\right\}_{n\in\mathbb{N}}.$$

Consider now the truncation of the above distributions to include only the view of the adversary up until and including round CR. The truncation of these *ideal* distributions yields $\text{VIEW}^1_{\mathcal{S}^{\mathcal{A}}}(x_1, z, n)$ in both cases, and so are identical. This is due to the fact that $\mathcal{S}$'s view is identical in both cases because the output is received only after this part of the view is fixed. Denote by $\text{VIEW}_{\mathcal{A},\pi}(x_1, x_2, z, n)$ the view of $\mathcal{A}$ alone in the execution. Since $\mathcal{A}$ halts at round CR, we have

$$\left\{\text{VIEW}^1_{\mathcal{S}^{\mathcal{A}}}(x_1, z, n)\right\}_{n\in\mathbb{N}} \stackrel{\text{c}}{\equiv} \left\{\text{VIEW}_{\mathcal{A},\pi}(x_1, x_2, z, n)\right\}_{n\in\mathbb{N}}$$

and

$$\left\{\text{VIEW}^1_{\mathcal{S}^{\mathcal{A}}}(x_1, z, n)\right\}_{n\in\mathbb{N}} \stackrel{\text{c}}{\equiv} \left\{\text{VIEW}_{\mathcal{A},\pi}(x_1, \tilde{x}_2, z, n)\right\}_{n\in\mathbb{N}}.$$

Combining the above, we have

$$\left\{\text{VIEW}_{\mathcal{A},\pi}(x_1, x_2, z, n)\right\}_{n\in\mathbb{N}} \stackrel{\text{c}}{\equiv} \left\{\text{VIEW}_{\mathcal{A},\pi}(x_1, \tilde{x}_2, z, n)\right\}_{n\in\mathbb{N}}.$$

However, by the contradicting assumption, $\mathcal{A}$ receives correct output by round CR, and its view defines its output. Thus, the view with input x_1, x_2 defines the output $f(x_1, x_2)$ for $\mathcal{A}$, while the view with input $x_1, \tilde{x}_2$ defines the output $f(x_1, \tilde{x}_2)$ for $\mathcal{A}$. Since $f(x_1, x_2) \neq f(x_1, \tilde{x}_2)$, the distributions are easily distinguishable, in contradiction. This completes the proof.

We now proceed to prove the second lemma that states that a protocol that is secure with a committal round and fairness can actually be truncated to the committal round and remain honest-correct. Intuitively, this holds since in the ideal model the simulator must provide the input used by the corrupted party by the committal round. Now, since the output is determined at this point and cannot change, this implies that the honest party must always output the function computed on its own input and the input provided by the simulator (it can also never abort since the corrupted party already learned the output at the

committal round). This in turn implies that the honest party must always output the same correct output in a real protocol execution, irrespective of where the corrupted party halts. In particular, it must hold even if the corrupted party halts immediately after the committal round. Formally, we prove this by showing that if correctness does not hold at the committal round, then there exists a specific round where it transitions from not holding to holding (clearly correctness holds for the full protocol π). Then, we show that a distinguisher can distinguish the real and ideal executions with an adversary that halts either at the round before the transition or at the transition. Note that the lemma does not hold for the case of multiparty computation with an honest majority; this is explained after the proof of the lemma.

Lemma 2. *Let f be a two-party function with a finite domain, let π be a protocol that securely computes f with a committal round and fairness, and let CR be the index of the committal round. Then, the protocol obtained by truncating π to exactly CR rounds is honest-correct.*

Proof. Denote by π_0 the protocol π truncated to round CR, and denote by π_ℓ the protocol π truncated to ℓ rounds *after* round CR. Let m be the number of rounds in π *after* the committal round and so $\pi_m = \pi$ (note that the total number of rounds in the protocol equals $CR + m$); clearly, m is polynomial in the security parameter n. Recall that by our definition of the real model, in each round of interaction, exactly one party sends a message and the other waits to receive it. Without loss of generality, we assume that the first message after round CR is from P_1 to P_2 (likewise all odd messages), and the second message after round CR is from P_2 to P_1 (likewise all even messages). In addition, we assume that m is even (if this is not true then just add a dummy message to π). In more detail, in protocol π_ℓ, the parties output what π specifies them to output in the event that the other party halts at this point. For example, if P_1 sends the last message in π_ℓ, then P_2's output in π_ℓ is the same as it would in π in the case of an adversarial P_1 who halts after sending the ℓth message after CR. Observe that in this example, P_1's output is the same in π_ℓ and $\pi_{\ell-1}$ since in both cases its last message received is from P_2 in the $(\ell-1)$th round after CR. In contrast, P_2's output *may* be different in these cases since its view is different.

Recall that by Definition 2, a protocol is honest-correct, if for *every* pair of inputs x_1, x_2 written on the parties' input tapes, their output when honestly running the protocol is $f(x_1, x_2)$, except with negligible probability. Observe that protocols $\pi_0, \ldots, \pi_m$ are fully specified and that we only consider executions of pairs of honest parties in these protocols. Thus, the notion of honest-correctness is well defined with respect to each π_ℓ (meaning that each of these protocols is either honest-correct or not honest-correct, and this is a property of the protocol alone).

We prove that π_0 is honest-correct. In order to see this, observe that π_m is honest-correct since $\pi_m = \pi$ and π is a secure protocol (since we consider here the case that both parties behave honestly, security in Definition 1 implies

honest-correctness as in Definition 2). By contradiction, assume that π_0 is not honest-correct, meaning that there exist inputs so that at least one of the parties outputs an incorrect output in π_0 with non-negligible probability. Then, there exists a *maximal* index ℓ ($1 \leq \ell \leq m$) such that π_ℓ is honest-correct, but $\pi_{\ell-1}$ is not honest-correct.

Without loss of generality, let P_1 be the party who sends the message in the ℓth round. This implies that P_1's view in π_ℓ and $\pi_{\ell-1}$ is identical, and thus its output is identical. However, since the protocol $\pi_{\ell-1}$ is not honest-correct, this in turn implies that P_2's output is *correct* in π_ℓ (except with negligible probability) but *incorrect* in $\pi_{\ell-1}$ with non-negligible probability. By definition, $\pi_{\ell-1}$ being incorrect means that there exist some inputs x_1, x_2 such that in an execution of $\pi_{\ell-1}$ on these inputs, P_2 receives some output value $y' \neq f(x_1, x_2)$ with non-negligible probability. Recall that honest-correctness applies to *all* inputs, and thus its negation may apply only to a specific pair of inputs. Let x_1, x_2 be inputs for which $\pi_{\ell-1}$ is not honest-correct; concretely, this means that with non-negligible probability P_2 outputs $y' \neq f(x_1, x_2)$.

We first prove that in $\pi_{\ell-1}$, except with negligible probability, the output received by P_2 must be $f(\tilde{x}_1, x_2)$ for some $\tilde{x}_1$, where x_2 is the input written on P_2's input tape. Intuitively this follows from the standard correctness property of secure protocols. Formally, in order to see this, we construct an adversary $\mathcal{A}$ who controls P_1 and interacts with an honest P_2 running π. $\mathcal{A}$ runs the protocol honestly with the exception that it halts after the $(\ell-1)$th round, and in particular, does not send its message in the ℓth round. By the security of π, simulator $\mathcal{S}_1$ when run on adversary $\mathcal{A}$ outputs some $\tilde{x}_1$ as P_1's input and it holds that the honest party's output in a real execution is indistinguishable from $f(\tilde{x}_1, x_2)$. Thus, P_2 must output $f(\tilde{x}_1, x_2)$ for some $\tilde{x}_1$.[5] (If this does not hold then the distinguisher can always distinguish since the function has a finite domain and so it can try all possible inputs for P_1 and see if P_2's output is $f(\tilde{x}_1, x_2)$ for some $\tilde{x}_1$.)

By what we have shown so far, when P_1 and P_2 run on inputs x_1 and x_2, respectively, we have that P_2 outputs $f(x_1, x_2)$ in π_ℓ, but with non-negligible probability outputs $f(\tilde{x}_1, x_2) \neq f(x_1, x_2)$ for some $\tilde{x}_1 \neq x_1$ in $\pi_{\ell-1}$. In contrast, in both π_ℓ and $\pi_{\ell-1}$, party P_1 has an identical view and thus has the same output. Since we know that π_ℓ is honest-correct, this implies that P_1 outputs $f(x_1, x_2)$ in both π_ℓ and $\pi_{\ell-1}$. We now show that this yields a contradiction. Before proceeding, we claim that there exist *specific* $x_1^*, x_2^*, \tilde{x}_1^*$ for which the above holds for infinitely many n's. That is, we claim that there exist $x_1^*, x_2^*, \tilde{x}_1^*$, an infinite set of integers $N \subseteq \mathbb{N}$ and a polynomial $p(\cdot)$, such that when given inputs x_1^*, x_2^*, respectively, P_1 and P_2 output $f(x_1^*, x_2^*)$ in π_ℓ except with negligible probability and in particular with probability greater than $1 - \frac{1}{2p(n)}$. In contrast, P_2 outputs $f(\tilde{x}_1^*, x_2^*) \neq f(x_1^*, x_2^*)$ in $\pi_{\ell-1}$ with probability at least $\frac{1}{p(n)}$. This holds since f

[5] Note that this actually implies that f is non-constant, since $f(\tilde{x}_1, x_2) = y' \neq f(x_1, x_2)$. Nevertheless, we do not need to assume this to prove this lemma (unlike Lemma 1), since this follows from the contradicting assumption.

has a finite domain: if f had an infinite domain then it would be possible that for every n there would exist a different pair of inputs for which the claim holds.[6]

Let $\mathcal{A}'$ be an adversary who controls P_1 and interacts with an honest P_2 with input x_2^* in a real protocol execution of (the untruncated) protocol π. $\mathcal{A}'$ runs the honest party's instructions with input x_1^* until the $(\ell-1)$th round after CR. Then, $\mathcal{A}'$ applies a pseudorandom function (with a randomly chosen key taken from its random tape) to its view up to round CR to determine if it sends the ℓth message. If the pseudorandom function's output is 0, then $\mathcal{A}'$ sends the $(CR+\ell)$th message to P_2 and halts; if the pseudorandom function's output is 1 then $\mathcal{A}'$ halts immediately in round $CR+\ell-1$ and before it sends the $(CR+\ell)$th message. We stress that $\mathcal{S}$ has only black-box access to $\mathcal{A}'$, and so cannot influence its input, auxiliary input and random-tape.[7]

We claim that $\mathcal{S}$ fails in the simulation of $\mathcal{A}'$. In order to see this, we first replace the pseudorandom function used by $\mathcal{A}'$ by a truly random function. By a straightforward reduction, the output of $\mathcal{S}$ with $\mathcal{A}'$ using a truly random function is computationally indistinguishable from when $\mathcal{A}'$ uses a pseudorandom function.

Next, observe that $\mathcal{S}$ must send the corrupted P_1's input to the trusted party in round CR and thus before it can see whether $\mathcal{A}'$ sends its message in the $(CR+\ell)$th round or not. However, this determines whether P_2 outputs $f(x_1^*, x_2^*)$ or $f(\tilde{x}_1^*, x_2^*)$ in the real model. Thus, $\mathcal{S}$ cannot know whether it should send x_1^* or $\tilde{x}_1^*$ to the trusted party. We now formally prove this argument.

Let D be a distinguisher who receives the output (including $\mathcal{A}'$'s view and P_2's output) and runs $\mathcal{A}'$ on its view to see if $\mathcal{A}'$ aborts at round $CR+\ell$ or $CR+\ell-1$. If $\mathcal{A}'$ aborts at round $CR+\ell$ and P_2's output is $f(x_1^*, x_2^*)$ or if $\mathcal{A}'$ aborts at round $CR+\ell-1$ and P_2's output is $f(\tilde{x}_1^*, x_2^*)$ then D outputs 1. Else, D outputs 0. We now analyze the probability that D outputs 1 in the real and ideal executions. Fix $n \in N$, where N is the infinite set of integers specified above.

– *Real execution:* Recall that if $\mathcal{A}'$ proceeds to round $CR+\ell$ then P_2 outputs $f(x_1^*, x_2^*)$ with probability greater than $1-1/2p(n)$, whereas if $\mathcal{A}'$ halts at round $CR+\ell-1$ then P_2 outputs $f(\tilde{x}_1^*, x_2^*)$ with probability at least $1/p(n)$. Furthermore, $\mathcal{A}'$ proceeds with probability 1/2. We therefore have that for every $n \in N$:

$$\Pr[D \text{ outputs } 1] \geq \frac{1}{2} \cdot \left(1 - \frac{1}{2p(n)}\right) + \frac{1}{2} \cdot \frac{1}{p(n)} = \frac{1}{2} + \frac{1}{4p(n)}.$$

[6] We believe that the proof would still hold for the case of infinite domain by providing the inputs for which the claim holds as non-uniform advice to the adversary and distinguisher. However, this would needlessly complicate things.

[7] One could define a weaker type of black-box access where the simulator can provide these values as part of its query. However, this would make no difference since we would then define $\mathcal{A}$' to ignore the input, auxiliary input and randomness and use hardwired values only.

– *Ideal execution:* The main observation here is that the probability that P_2 outputs $f(x_1^*, x_2^*)$ or $f(\tilde{x}_1^*, x_2^*)$ is *independent* of whether or not $\mathcal{A}'$ proceeds to round $CR + \ell$ or halts at $CR + \ell - 1$. This holds because P_2's output is defined by the input provided by $\mathcal{S}$ and this is provided before $\mathcal{S}$ can know if $\mathcal{A}'$ halts in round $CR + \ell$ or $CR + \ell - 1$ since $\mathcal{S}$ can only send queries of length CR to $\mathcal{A}'$ before sending the input. (Note that if P_2 outputs anything else then D will output 0 and so this is not included in the calculation below.) Thus:

$$\begin{aligned}
\Pr[D \text{ outputs } 1] &= \Pr[\mathcal{A}' \text{ halts at } CR+\ell \textit{ and } P_2 \text{ outputs } f(x_1^*, x_2^*)] \\
&\quad + \Pr[\mathcal{A}' \text{ halts at } CR+\ell-1 \textit{ and } P_2 \text{ outputs } f(\tilde{x}_1^*, x_2^*)] \\
&= \Pr[\mathcal{A}' \text{ halts at } CR+\ell] \cdot \Pr[P_2 \text{ outputs } f(x_1^*, x_2^*)] \\
&\quad + \Pr[\mathcal{A}' \text{ halts at } CR+\ell-1] \cdot \Pr[P_2 \text{ outputs } f(\tilde{x}_1^*, x_2^*)] \\
&= \frac{1}{2} \cdot \Pr[P_2 \text{ outputs } f(x_1^*, x_2^*)] + \frac{1}{2} \cdot \Pr[P_2 \text{ outputs } f(\tilde{x}_1^*, x_2^*)] \\
&= \frac{1}{2} \cdot \Big(\Pr[P_2 \text{ outputs } f(x_1^*, x_2^*)] + \Pr[P_2 \text{ outputs } f(\tilde{x}_1^*, x_2^*)] \Big) \\
&\leq \frac{1}{2}
\end{aligned}$$

where the second equality is by the *independence* of probabilities explained above. We remark that in the last step, it is not equality since P_2 may output something else.

We have shown that for infinitely many n's (for every $n \in N$), distinguisher D distinguishes between the real and ideal executions with probability at least $1/4p(n)$. Thus, D distinguishes with non-negligible probability, in contradiction to the assumed security of the protocol.

Lemmas 1 and 2 contradict each other therefore completing the proof of Theorem 3.

The case of an honest majority. In the setting of multiparty computation with an honest majority, our proof does not hold. This is due to the fact that our proof relies inherently on the fact that when the adversary halts, the honest party can receive no more information towards obtaining its output. Rather, its view until that halting point is all that it receives. (Formally, this can be seen in the proof of Lemma 2 where we say that P_2's output changes if $\mathcal{A}$ halts in round $CR+\ell-1$ or halts in round $CR + \ell$.) In contrast, when there is an honest majority, the honest parties may continue to exchange messages even if all corrupted parties halt.

Semi-trivial functions. Our proof holds for *all* non-constant functions, including functions f that can be singlehandedly determined by one of the parties. In particular, consider a function f such that for every x_1 and all $x_2, \tilde{x}_2$ it holds that $f(x_1, x_2) = f(x_1, \tilde{x}_2)$, meaning that P_2's input is meaningless. Such a function *cannot* be securely computed with fairness under our definition with a committal round. However, observe that *all* such functions (with a polynomial-size domain)

can be securely computed with fairness under Canetti's definition using a *trivial protocol* (in particular, the protocol of [20] is not required). Specifically, party P_1 can simply compute the output itself and send it to P_2. This protocol is fair since if P_1 does not send the output then P_2 can compute the function on its real input and a default input for P_1. (Note that P_2 must also check that the output is valid in that there exists such a value in the domain of f, and otherwise should also compute a default output. Since we consider finite-domain functions only here, P_2 can always do this.) More formally, a simulator under the definition of Canetti can obtain the value sent by a corrupted P_1 and simply find an input that leads to such an output (this is possible since the domain is polynomial-size). Furthermore, when P_2 is corrupted, the simulator just receives the output and simulates P_1 sending that value. Note that this protocol is not secure under our definition with a committal round: if the committal round is before P_1 sends the message then the simulator in the case that P_1 is corrupted cannot send the input to the trusted party, and if the committal round is after P_1 sends the message then the simulator in the case that P_2 is corrupted cannot simulate the first phase.

References

1. Aggarwal, G., Mishra, N., Pinkas, B.: Secure computation of the k^{th}-ranked element. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 40–55. Springer, Heidelberg (2004). doi:10.1007/978-3-540-24676-3_3
2. Asokan, N., Schunter, M., Waidner, M.: Optimistic protocols for fair exchange. In: The 4th ACM Conference on Computer and Communications Security, pp. 8–17 (1997)
3. Asharov, G.: Towards characterizing complete fairness in secure two-party computation. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 291–316. Springer, Heidelberg (2014). doi:10.1007/978-3-642-54242-8_13
4. Asharov, G., Beimel, A., Makriyannis, N., Omri, E.: Complete characterization of fairness in secure two-party computation of boolean functions. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015. LNCS, vol. 9014, pp. 199–228. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46494-6_10
5. Asharov, G., Lindell, Y., Rabin, T.: A full characterization of functions that imply fair coin tossing and ramifications to fairness. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 243–262. Springer, Heidelberg (2013). doi:10.1007/978-3-642-36594-2_14
6. Beaver, D.: Foundations of Secure Interactive Computing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 377–391. Springer, Heidelberg (1992). doi:10.1007/3-540-46766-1_31
7. Beaver, D., Goldwasser, S.: Multiparty computation with faulty majority. In: 30th FOCS, pp. 468–473 (1989)
8. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: 20th STOC, pp. 1–10 (1988)
9. Canetti, R.: Security and composition of multiparty cryptographic protocols. J. Cryptol. **13**(1), 143–202 (2000)

10. Canetti, R.: Universally Composable Security: A new paradigm for cryptographic protocols. In: 42nd FOCS, pp. 136–145 (2001). Full version http://eprint.iacr.org/2000/067
11. Canetti, R., Fischlin, M.: Universally composable commitments. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 19–40. Springer, Heidelberg (2001). doi:10.1007/3-540-44647-8_2
12. Canetti, R., Kushilevitz, E., Lindell, Y.: On the limitations of universal composable two-party computation without set-up assumptions. J. Crypt. **19**(2), 135–167 (2006)
13. Chaum, D., Crépeau, C., Damgård, I.: Multi-party unconditionally secure protocols. In: 20th STOC, pp. 11–19 (1988)
14. Cleve, R.: Limits on the security of coin flips when half the processors are faulty. In: 18th STOC, pp. 364–369 (1986)
15. Dodis, Y., Micali, S.: Parallel reducibility for information-theoretically secure computation. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 74–92. Springer, Heidelberg (2000). doi:10.1007/3-540-44598-6_5
16. Goldreich, O.: Foundations of Cryptography: Basic Tools, vol. 1. Cambridge University Press, Cambridge (2001)
17. Goldreich, O.: Foundations of Cryptography: Basic Applications, vol. 2. Cambridge University Press, Cambridge (2004)
18. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game - a completeness theorem for protocols with honest majority. In: 19th STOC, pp. 218–229 (1987) For details see [17, Chap. 7]
19. Goldwasser, S., Levin, L.: Fair computation of general functions in presence of immoral majority. In: Menezes, A.J., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 77–93. Springer, Heidelberg (1991). doi:10.1007/3-540-38424-3_6
20. Gordon, S.D., Hazay, C., Katz, J., Lindell, Y.: Complete fairness in secure two-party computation. J. ACM **58**(6), 24 (2011). An extended abstract appeared at the 40th STOC, pp. 413–422 (2008)
21. Gordon, S.D., Katz, J.: Complete fairness in multi-party computation without an honest majority. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 19–35. Springer, Heidelberg (2009). doi:10.1007/978-3-642-00457-5_2
22. Micali, S.: Simple and fast optimistic protocols for fair electronic exchange. In: The 22nd PODC, pp. 12–19 (2003)
23. Micali, S., Rogaway, P.: Secure computation. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 392–404. Springer, Heidelberg (1992). doi:10.1007/3-540-46766-1_32
24. Rabin, T., Ben-Or, M.: Verifiable secret sharing and multiparty protocols with honest majority. In: The 21st STOC, pp. 73–85 (1989)