Universally Composable Multi-Party Computation with an Unreliable Common Reference String[*]

Vipul Goyal[†] Jonathan Katz[‡]

Abstract

Universally composable multi-party computation has been studied in two settings:

- When a majority of participants are honest, universally composable multi-party computation is known to be possible without any assumptions.
- When honest participants are *not* in the majority, universally composable multi-party computation is known to be impossible (under any cryptographic assumption) in the bare model. On the other hand, feasibility results have been obtained (under standard cryptographic assumptions) in various augmented models, the most popular of which posits the existence of a *common references string* (CRS) available to all parties who are executing the protocol.

In either of the above settings, some *assumption* regarding the protocol execution is made (i.e., that many parties are honest in the first case, or that a legitimately-chosen string is available in the second), and if this assumption is incorrect then all security is lost.

A natural question is whether it is possible to design protocols giving *some* assurance of security in case *either one* of these assumptions holds, i.e., a single protocol (that uses a CRS) which is secure if *either* at most s players are dishonest *or* if up to t players are dishonest (with $t > s$) but the CRS is chosen in the proscribed manner. We show that such protocols exist if and only if $s + t < n$.

1 Introduction

Protocols proven to satisfy the definition of *universal composability* [5] offer strong and desirable security guarantees. Informally speaking, such protocols remain secure even when executed concurrently with arbitrary other protocols running in some larger network, and can be used as sub-routines of larger protocols in a modular fashion.

Universally composable (UC) multi-party computation of arbitrary functionalities has been investigated in two settings. When a majority of the parties running a protocol are assumed to be honest, UC computation of arbitrary functionalities is possible without any cryptographic assumptions (as claimed in [5], building on [3, 17]). This result holds in the so-called "plain

[*]This work was done in part while the authors were visiting IPAM.

[†]Department of Computer Science, UCLA. Email: `vipul@cs.ucla.edu`. Research supported in part by NSF ITR and Cybertrust programs (including grants 0430254, 0627781, 0456717, and 0205594).

[‡]Department of Computer Science, University of Maryland. Email: `jkatz@cs.umd.edu`. Research supported in part by NSF Trusted Computing grant #0310499, NSF CAREER award #0447075, and US-Israel Binational Science Foundation grant #2004240.

model" which assumes only pairwise private and authenticated channels between each pair of parties. (A broadcast channel is not required [10], since fairness and output delivery are not guaranteed in the UC framework.)

In contrast, when the honest players *cannot* be assumed to be in the majority, it is known that UC computation of arbitrary functions is not possible in the plain model regardless of any cryptographic assumptions made. Canetti and Fischlin [7] showed the impossibility of two-party protocols for commitment and zero knowledge, and Canetti, Kushilevitz, and Lindell [8] ruled out UC two-party computation of a wide class of functionalities.

To circumvent these far-reaching impossibility results, researchers have investigated various *augmented* models in which UC computation without honest majority might be realizable [5, 7, 9, 1, 12, 6, 15]. The most widely-used of these augmented models is the one originally propounded by Canetti and Fischlin [7], in which a *common reference string* (CRS) is assumed to be available to all parties running a given execution of a protocol. (The use of a common reference string in cryptographic protocols has a long history that can be traced back to [4].) Canetti and Fischlin show that UC commitments and zero knowledge are possible in the two-party setting when a CRS is available, and later work of Canetti et al. [9] showed that (under suitable cryptographic assumptions) a CRS suffices for UC multi-party computation of arbitrary functionalities.

In summary, there are two types of what we might term "assumptions about the world" under which UC multi-party computation is known to be possible:

- When a strict minority of players are dishonest.
- When an arbitrary number of players may be dishonest, but a trusted CRS (or some other setup assumption) is available.

Our contribution. Known protocols designed under one of the assumptions listed above are *completely insecure* in case the assumption turns out not to hold. For example, the CLOS protocol [9] — that is secure in the presence of an arbitrary number of corrupted parties when a trusted CRS is available — is completely insecure in the presence of even a *single* corrupted party if the protocol is run using a CRS σ that is taken from the wrong distribution or, even worse, adversarially generated. Similarly, the BGW protocol [3] is completely insecure in case half or more of the parties are dishonest. Given this state of affairs, a natural question is whether it is possible to design a *single* protocol Π that uses a common reference string σ and simultaneously guarantees the following:

- If σ is generated "honestly" (i.e., by a trusted third party according to the expected distribution), then Π is secure as long as at most t parties are corrupted.
- *Regardless of how the CRS σ used by* Π *is generated* (and, in particular, even if σ is adversarially generated), Π is secure as long as at most s parties are corrupted.

In this case, we will call the protocol Π an "(s, t)-secure protocol". It follows from [7, 8] that (s, t)-security for general functionalities is only potentially achievable if $s < n/2$, where n is the total number of parties running the protocol. A priori, we might hope to achieve the "best possible" result that $(\lfloor (n-1)/2 \rfloor, n-1)$-secure protocols exist for arbitrary functionalities.

Here, we show matching positive and negative answers to the above question. First, we show that for any $s + t < n$ (and $s < n/2$) there exists an (s,t)-secure protocol realizing any functionality. We complement this with a result showing that this is, unfortunately, the best possible: if $s + t = n$ then there exists a large class of functionalities (inherited, in some sense, from [8]) for which no (s,t)-secure protocol exists. We remark that we prove security under adaptive corruptions for our positive result, while our negative result holds even for the case of non-adaptive corruptions.

The two extremes of our positive result (i.e., when $s = t = \lfloor (n-1)/2 \rfloor$, or when $s = 0$ and $t = n - 1$) correspond to, respectively, the known protocols that require honest majority (e.g., [3, 17]) or that tolerate an arbitrary number of malicious participants but require a CRS (e.g., [9]). Our results exhibit new protocols in between these two extremes. Choice of which protocol to use reflects a tradeoff between the level of confidence in the CRS and the number of corruptions that can be tolerated: e.g., choosing $s = 0$ represents full confidence in the CRS, while setting $s = \lfloor (n-1)/2 \rfloor$ (and ignoring the CRS) means that there is effectively no confidence in the CRS at all.

Related Work. Another suggestion for circumventing the impossibility results of [7, 8] has been to use a definition of security where the ideal-model simulator is allowed to run in *super-polynomial* time [16, 2]. This relaxation is sufficient to bypass the known impossibility results and leads to constructions of protocols for any functionality without setup assumptions. While these constructions seem to supply adequate security for certain applications, they require stronger (sub-exponential time) complexity assumptions and can be problematic when used as sub-routines within larger protocols.

Some other recent work has also considered the construction of protocols having "two tiers" of security. Barak, Canetti, Nielsen, and Pass [1] show a protocol relying on a key-registration authority: if the key-registration authority acts honestly the protocol is universally composable, while if this assumption is violated the protocol still remains secure in the stand-alone sense. Ishai et al. [13] and Katz [14], in the stand-alone setting, studied the question of whether there exist protocols that are "fully-secure" (i.e., guaranteeing privacy, correctness, and fairness) in the presence of a dishonest minority, yet still "secure-with-abort" otherwise. While the motivation in all these cases is similar, the problems are different and, in particular, a solution to our problem does not follow from (or rely on) any of these prior results.

Groth and Ostrovsky [11] recently introduced the *multi-CRS model* for universally composable multi-party computation. In this model, roughly speaking, the parties have access to a set of k common reference strings, some k' of which are guaranteed to have been chosen honestly (that is, by a trusted party). The remaining $k - k'$ strings can be chosen in an arbitrary manner. (Of course, it is not known which strings are "good" and which are"bad".) Groth and Ostrovsky explore conditions on k, k' under which UC multi-party computation is still possible. Although in both cases the question boils down to what security guarantees can be achieved if a supposedly "good" CRS turns out to be "bad", our end results are very different. In the work of Groth and Ostrovsky the number of corruptions to be tolerated is fixed and there are assumed to be some minimal number k' of "good" strings among the k available ones. In our work, in contrast, it is possible that *no* "good" CRS is available at all; even in this case, though, we would like to ensure security against some (necessarily) smaller set of corrupted parties. On the other hand, we do rely on the Groth-Ostrovsky result as a building block for our positive result.

2 Preliminaries

2.1 Review of the UC Framework

We give a brief overview of the UC framework, referring the reader to [5] for more details.

The UC framework allows for defining the security properties of cryptographic tasks so that security is maintained under general composition with an unbounded number of instances of arbitrary protocols running concurrently. In the UC framework, the security requirements of a given task are captured by specifying an ideal functionality run by a "trusted party" that obtains the inputs of the participants and provides them with the desired outputs. Informally, then, a protocol securely carries out a given task if running the protocol in the presence of a real-world adversary amounts to "emulating" the desired ideal functionality.

The notion of emulation in the UC framework is considerably stronger than that considered in previous models. As usual, the real-world model includes the parties running the protocol and an adversary $\mathcal{A}$ who controls their communication and potentially corrupts parties, while the ideal-world includes a simulator $\mathcal{S}$ who interacts with an ideal functionality $\mathcal{F}$ and dummy players who simply send input to/receive output from $\mathcal{F}$. In the UC framework, there is also an additional entity called the *environment* $\mathcal{Z}$. This environment generates the inputs to all parties, observes all their outputs, and interacts with the adversary in an arbitrary way throughout the computation. A protocol Π is said to *securely realize* an ideal functionality $\mathcal{F}$ if for any real-world adversary $\mathcal{A}$ that interacts with $\mathcal{Z}$ and real players running Π, there exists an ideal-world simulator $\mathcal{S}$ that interacts with $\mathcal{Z}$, the ideal functionality $\mathcal{F}$, and the "dummy" players communicating with $\mathcal{F}$, such that *no* poly-time environment $\mathcal{Z}$ can distinguish whether it is interacting with $\mathcal{A}$ (in the real world) or $\mathcal{S}$ (in the ideal world). $\mathcal{Z}$ thus serves as an "interactive distinguisher" between a real-world execution of the protocol Π and an ideal execution of functionality $\mathcal{F}$. A key point is that $\mathcal{Z}$ cannot be re-wound by $\mathcal{S}$; in other words, $\mathcal{S}$ must provide a so-called "straight-line" simulation.

The following *universal composition theorem* is proven in [5]. Consider a protocol Π that operates in the $\mathcal{F}$-hybrid model, where parties can communicate as usual and in addition have ideal access to an unbounded number of *copies* of the functionality $\mathcal{F}$. Let ρ be a protocol that securely realizes $\mathcal{F}$ as sketched above, and let Π^ρ be identical to Π with the exception that the interaction with *each copy* of $\mathcal{F}$ is replaced with an interaction with a *separate instance* of ρ. Then, Π and Π^ρ have essentially the same input/output behavior. In particular, if Π securely realizes some functionality $\mathcal{G}$ in the $\mathcal{F}$-hybrid model then Π^ρ securely realizes $\mathcal{G}$ in the standard model (i.e., without access to any functionality).

2.2 Definitions Specific to Our Setting

We would like to model a single protocol Π that uses a CRS σ, where σ either comes from a trusted functionality $\mathcal{F}_{CRS}$ (where this functionality is as in [7] and all subsequent work on UC computation in the CRS model) or is chosen in an arbitrary manner by the environment $\mathcal{Z}$. A minor technical detail is that parties running Π can trivially "tell" where σ comes from depending on which incoming communication tape σ is written on (since an ideal functionality would write inputs to a different tape than $\mathcal{Z}$ would). Because this does not correspond to what we are attempting to model in the real world, we need to effectively "rule out" protocols that utilize this additional knowledge. The simplest way to do this is to define a "malicious CRS" functionality $\mathcal{F}_{mCRS}$ that we now informally describe. Functionality $\mathcal{F}_{mCRS}$ takes input σ from the adversary $\mathcal{A}$ and then, when activated by any party P_i, sends σ to that party. The overall effect of this is that $\mathcal{A}$ (and hence $\mathcal{Z}$) can set the CRS to any value of its choice; however, it is forced to provide the *same* value to all parties running protocol Π. When the parties interact with $\mathcal{F}_{CRS}$, this (intuitively) means that the CRS is "good"; when they interact with $\mathcal{F}_{mCRS}$ the CRS is "bad". We refer to this setting, where parties interact with either $\mathcal{F}_{CRS}$ or $\mathcal{F}_{mCRS}$ but do not know which, as the *mixed CRS model.* We can now define an (s, t)-secure protocol.

Definition 1 *We say a protocol* Π (s, t)-securely realizes a functionality $\mathcal{F}$ in the mixed CRS model *if*

(a) Π *securely realizes* $\mathcal{F}$ *in the* $\mathcal{F}_{mCRS}$*-hybrid model when at most* s *parties are corrupted.*

(b) Π *securely realizes* $\mathcal{F}$ *in the* $\mathcal{F}_{CRS}$*-hybrid model when at most* t *parties are corrupted.*

We stress that Π itself does not "know" in which of the two hybrid models it is being run. The ideal-world adversary $\mathcal{S}$, however, may have this information hard-wired in. (More concretely: although Π is a fixed protocol, two different ideal-world adversaries $\mathcal{S}$, $\mathcal{S}'$ may be used in proving each part of the definition above.)

3 Positive Result for $s + t < n$

We begin by showing our positive result: if $s+t < n$ and $s < n/2$ (where n is the total number of parties running the protocol), then essentially any functionality $\mathcal{F}$ can be (s, t)-securely realized in the mixed CRS model. This is subject to two minor technical conditions [9] we discuss briefly now.

Non-trivial protocols. The ideal process does not require the ideal-process adversary to deliver the messages that are sent between the ideal functionality and the parties. A corollary of the above fact is that a protocol that "hangs", never sends any messages and never generates output, securely realizes any ideal functionality. However, such a protocol is uninteresting. Therefore, we stick to the notion of a non-trivial protocol [9], for which if the real-life adversary delivers all messages, then the ideal-process adversary delivers all messages from the ideal functionality to the parties. This captures the minimal property that if all parties are honest and adversary does not prevent the delivery of any message then all parties receive output.

Well-formed functionalities. A well-formed functionality is oblivious of the corruptions of parties, runs in polynomial time, and reveals the internal randomness used by the functionality to the ideal process adversary in case all parties are corrupted [9]. This class contains all functionalities we can hope to securely realize from a non-trivial protocol, as discussed in [9].

We can now formally state the result of this section:

Theorem 1 *Assume that enhanced trapdoor permutations, augmented non-committing encryption schemes, and dense cryptosystems exist. Fix s, t, n with $s + t < n$ and $s < n/2$. Then for every well-formed n-party functionality $\mathcal{F}$, there exists a non-trivial protocol Π which (s, t)-securely realizes $\mathcal{F}$ in the mixed CRS model.*

The cryptographic assumptions of the theorem are inherited directly from [9], and we refer the reader there for formal definitions of each of these. Weaker assumptions suffice to achieve security against static corruptions; see [9].

To prove the above theorem, we rely on the results of Groth and Ostrovsky regarding the multi-CRS model [11]. Very informally speaking, they show the following as one of their results: Let there be parties $P_1, \ldots, P_n$ having access to $k \geq 1$ strings $\sigma_1, \ldots, \sigma_k$. As long as $k' > k/2$ of these strings are honestly generated according to some specified distribution $\mathcal{D}$ (and assuming the same cryptographic assumptions of the theorem stated above), then for every well-formed functionality $\mathcal{F}$ there exists a non-trivial protocol Π securely realizing $\mathcal{F}$. We stress that the remaining $k - k'$ strings can be generated arbitrarily (i.e., adversarially), even possibly depending on the k' honestly-generated strings.

Building on the above result, we now describe our construction. We assume there are n parties $P_1, \ldots, P_n$ who wish to run a protocol to realize a (well-formed) functionality $\mathcal{F}$. Construct a protocol Π as follows:

1. All parties begin with the same string σ^* provided as input. (Recall the parties do not know whether this is a "good" CRS or a "bad" CRS.) $P_1, \ldots, P_n$ first "amplify" the given string σ^* to m CRSs $\sigma_1^*, \ldots, \sigma_m^*$, where m is a parameter which is defined later on. The requirements here are simply that if σ^* is "good", then each of $\sigma_1^*, \ldots, \sigma_m^*$ should be "good" also. (If σ^* is "bad" — i.e., chosen by $\mathcal{F}_{mCRS}$ — then there is no guarantee on $\sigma_1^*, \ldots, \sigma_m^*$.)

   The above can be accomplished by using the CLOS protocol [9] as follows. Define an ideal functionality $\mathcal{F}_{m\_new\_CRS}$ which generates m new CRSs from the appropriate distribution $\mathcal{D}$ (where $\mathcal{D}$ refers to the the distribution used in the Groth-Ostrovsky result mentioned above) and outputs these to all parties. When running the CLOS protocol, use the given string σ^* as the CRS.

   Note that when σ^* was produced by $\mathcal{F}_{CRS}$, security of the CLOS protocol guarantees that the m resulting CRSs are all chosen appropriately. On the other hand, there are *no* guarantees in case σ^* was produced by $\mathcal{F}_{mCRS}$, but recall that we do not require anything in that case anyway.

2. Following the above, each party P_i chooses a string σ_i according to distribution $\mathcal{D}$ (where, again, $\mathcal{D}$ is the distribution used in the Groth-Ostrovsky result mentioned above), and broadcasts σ_i to all other parties.[1]

3. Each party receives $\sigma_1, \ldots, \sigma_n$, and sets $\sigma_{n+i} = \sigma_i^*$ for $i = 1$ to m.

4. All parties now have $n + m$ strings $\sigma_1, \ldots, \sigma_{n+m}$. These strings are used to run the Groth-Ostrovsky protocol for $\mathcal{F}$.

We claim that for any s, t satisfying the conditions of Theorem 1, it is possible to set m so as to obtain a protocol Π that (s, t)-securely realizes $\mathcal{F}$. The conditions we need to satisfy are as follows:

- When Π is run in the F_{CRS}-hybrid model, σ^* is a "good" CRS and so the strings $\sigma_1^*, \ldots, \sigma_m^*$ are also "good". The $n-t$ honest parties contribute another $n-t$ "good" strings in step 2, above, for a total of $n-t+m$ "good" strings in the set of strings $\sigma_1, \ldots, \sigma_{n+m}$. At most t of the strings in this set (namely, those contributed by the t malicious parties) can be "bad". For the Groth-Ostrovsky result to apply, we need $n - t + m > t$ or
$$m > 2t - n. \tag{1}$$

- When Π is run in the F_{mCRS}-hybrid model, σ^* is adversarially-chosen and so we must assume that the strings $\sigma_1^*, \ldots, \sigma_m^*$ are also "bad". In step 2, the malicious parties contribute another s "bad" strings (for a total of $m + s$ "bad" strings), while the $n - s$ honest parties contribute $n - s$ "good" strings. For the Groth-Ostrovsky result to apply, we now need $n - s > m + s$ or
$$m < n - 2s. \tag{2}$$

Since m, t, n are all integers, Equations (1) and (2) imply
$$2t - n \leq n - 2s - 2$$
or $s + t \leq n - 1$. When this condition holds, the equations can be simultaneously satisfied by setting $m = n - 2s - 1$, which gives a positive solution if $s < n/2$.

4 Impossibility Result for $s + t \geq n$

In this section, we state and prove our main impossibility result which shows that the results of the previous section are tight.

[1]The "broadcast" used here is the universally-composable broadcast protocol from [10] (which achieves a weaker definition than "standard" broadcast, but suffices for constructing protocols in the UC framework).

Theorem 2 *Let n, t, s be such that $t + s \geq n$. Then there exists a well-formed deterministic functionality for which no non-trivial n-party protocol exists that (s, t)-securely realizes $\mathcal{F}$ in the mixed CRS model.*

We in fact show that the above theorem holds for a large class of functionalities. That is, there exists a large class of functionalities for which no such non-trivial protocol exists.

The proof of Theorem 2 relies on ideas from the impossibility result of Canetti, Kushilevitz, and Lindell [8] that applies to 2-party protocols in the plain model. Since ours is inherently a multi-party scenario, our proof proceeds in two stages. In the first stage of our proof, we transform any n-party protocol Π that securely computes a function f in the mixed CRS model, into a two-party protocol Σ that computes a related function g (derived from f) called the *t-division* of the function f. Protocol Σ guarantees security for one party in the $\mathcal{F}_{CRS}$-hybrid model, and security for the other party in the *plain model.* In the second stage of our proof, we show that one of the parties running Σ can run a successful *split simulator strategy* [8] against the other. As in [8], the existence of a split simulator strategy means that the class of functionalities that can be securely realized by the two-party protocol Σ is severely restricted. This also restricts the class of functionalities f which can be realized using the original n-party protocol.

We now give the details. Let $x \| y$ denote the concatenation of x and y. We first define the *t-division* of a function f.

Definition 2 *Let $f = (f_1, \ldots, f_n)$ be a function taking n inputs $x_1, \ldots, x_n$ and returning n (possibly different) outputs. Define the two-input/two-output function $g = (g_1, g_2)$, the t-*division of f *via:*

$$\begin{aligned} g_1\Big(\overbrace{(x_1\|\cdots\|x_t)}^{I_1},\ \overbrace{(x_{t+1}\|\cdots\|x_n)}^{I_2}\Big) &= f_1(x_1,\ldots,x_n)\|\cdots\|f_t(x_1,\ldots,x_n) \\ g_2\big((x_1\|\cdots\|x_t),\ (x_{t+1}\|\cdots\|x_n)\big) &= f_{t+1}(x_1,\ldots,x_n)\|\cdots\|f_n(x_1,\ldots,x_n). \end{aligned}$$

Lemma 1 (Reduction to Two Party) *Let n, t, s be such that $t + s = n$ and $s < n/2$. Let there be an n-party (s, t)-secure protocol Π using which parties $P_1, \ldots, P_n$ holding inputs $x_1, \ldots, x_n$ can evaluate a function $f(x_1, \ldots, x_n)$. Then there exists a two party protocol Σ using which parties p_1, p_2 holding inputs $I_1 = x_1 \| \ldots \| x_t, I_2 = x_{t+1} \| \ldots \| x_n$ can evaluate the t-division function $g(I_1, I_2)$. Further Σ is fully secure in the $\mathcal{F}_{CRS}$-hybrid model and secure against a dishonest p_2 in the $\mathcal{F}_{mCRS}$-hybrid model.*

Proof We construct the protocol Σ using the protocol Π. The basic idea is as follows. The parties p_1 and p_2 break their input I_1, I_2 into several parts and start emulating n parties running the protocol Π to compute f on those input parts. Some of these parties in Π are controlled and emulated by p_1 and others by p_2. Finally when Π finishes, p_1 and p_2 get several outputs f_i meant for parties controlled by them. Using these outputs, p_1 and p_2 then individually reconstruct their final output g_1 and g_2. More details follow.

The parties p_1, p_2 hold inputs $I_1 = x_1 \| \ldots \| x_t, I_2 = x_{t+1} \| \ldots \| x_n$ and wish to compute the function g. p_1 internally starts emulating parties $P_1, \ldots, P_t$ on inputs $x_1, \ldots, x_t$ respectively to compute the function f. Similarly, p_2 starts emulating parties $P_{t+1}, \dots, P_n$ on inputs $x_{t+1}, \dots, x_n$. Whenever P_i sends a message M to P_j in the protocol Π, it is handled in Σ as follows. If $i, j \leq t$, p_1 internally delivers M from P_i to P_j. Similarly, if $i, j > t$, p_2 internally delivers M from P_i to P_j. Otherwise if $i \leq t, j > t$, p_1 sends a message (i, j, M) to p_2 who then internally delivers M to P_j as if it was received from P_i. The case $i > t, j \leq t$ is also handled similarly. After Π finishes, $P_1, \dots, P_t$ halt outputting $f_1, \dots, f_t$. Hence, p_1 obtains $g_1 = f_1 \| \dots \| f_t$. Similarly, p_2 obtains $g_2 = f_{t+1} \| \dots \| f_n$.

To see the security, recall that Π is t-secure in the $\mathcal{F}_{\text{CRS}}$-hybrid model. This means that Π securely computes f in the presence of any coalition of up to t corrupted parties. This in particular means that Π remains secure if all of $P_1, \dots, P_t$ are corrupted. Thus, Σ remains secure against a dishonest p_1 (who controls $P_1, \dots, P_t$). Also since $s \leq t$ (because $s < n/2$), Π also remains secure if all of $P_{t+1}, \dots, P_n$ are corrupted and hence so does Σ against a dishonest p_2. Thus we get that Σ securely computes g in the $\mathcal{F}_{\text{CRS}}$-hybrid model. Now recall that Π is s-secure in the $\mathcal{F}_{mCRS}$-hybrid model. This means that Π remains secure if all of $P_{t+1}, \dots, P_n$ are corrupted. Hence we get that Σ is secure against a dishonest p_2 in the $\mathcal{F}_{mCRS}$-hybrid model. In other words, Σ securely computes g as long as p_1 remains honest (irrespective of whether the CRS is adversarially chosen or comes from the right distribution). ■

In the second stage, we now show that a malicious p_2 can run a *successful split simulator strategy* against an honest p_1 in the protocol Σ in the $\mathcal{F}_{mCRS}$-hybrid model. Thus even if p_1 remains honest, this shows that a large class of functionalities cannot be securely realized by Σ.[2] This, in turn, will finally show the existence of a class of functionalities which cannot be (s, t)-securely realized by the protocol Π (as long as $t + s \geq n$).

Showing the existence of a successful split simulator strategy for p_2 amounts to reproving the *main technical lemma* for our setting. We start by first recalling a few definitions and notations from [9, 8]. Part of it is taken almost verbatim from [8].

Notation. Let $g : D \times D \to \{0,1\}^* \times \{0,1\}^*$ be a deterministic, polynomial time computable function, where $D \subseteq \{0,1\}^*$ is an arbitrary, possibly infinite, domain of inputs. Although we assume that both inputs to g are from same domain D, it is only for notational convenience and our results hold even if two inputs come from different domains. Function g is denoted as $g = (g_1, g_2)$ where g_1 and g_2 denote the outputs of p_1 and p_2 respectively.

Definition 3 (Split Adversarial Strategy, [8]) *Let Σ be a protocol to securely compute the function g. Let $D_2 \subseteq D$ be a polynomial-size subset of inputs (i.e., $|D_2| = \text{poly}(\kappa)$, where κ is a security parameter), and let $I_2 \in D_2$. Then a corrupted party p_2 is said to run a* split adversarial strategy *if it consists of machines p_2^a and p_2^b such that:*

1. *Upon input (D_2, I_2), party p_2 internally gives the machines p_2^b the input pair (D_2, I_2).*

2. *An execution between (an honest) p_1 running Σ and $p_2 = (p_2^a, p_2^b)$ works as follows:*

   (a) p_2^a interacts with p_1 according to some specified strategy.

[2]Note that in [8], it was shown that *both* the parties p_1 and p_2 can run a split simulator strategy against each other. In our case, we can only show that p_2 can do so against p_1. Hence, the class of functionalities which we prove are impossible to realize is somewhat smaller than that in [8].

(b) At some stage of the execution p_2^a hands p_2^b a value I'_1.

(c) When p_2^b receives I'_1 from p_2^a, it computes $J'_1 = g_1(I'_1, I'_2)$ for some $I'_2 \in D_2$ of its choice.

(d) p_2^b hands p_2^a the value J'_1, and p_2^a continues interacting with p_1.

Definition 4 (Successful Strategies, [8]) *Let g, Σ, κ be as defined in Definition 2. Let $\mathcal{Z}$ be the environment who hands input I_1 to p_1 and a pair (D_2, I_2) to p_2 where $D_2 \subseteq D$, $|D_2| = \text{poly}(\kappa)$, and $I_2 \in_r D_2$. Then a split adversarial strategy for p_2 is said to be successful if for every $\mathcal{Z}$ as above and every input z to $\mathcal{Z}$, the following conditions hold in a real execution of p_2 with $\mathcal{Z}$ and honest p_1:*

1. *The value I'_1 output by p_2^a in step 2b of Definition 2 is such that for every $I_2 \in D_2, g_2(I'_1, I_2) = g_2(I_1, I_2)$.*

2. *p_1 outputs $g_1(I_1, I'_2)$, where I'_2 is the value chosen by p_2^a in step 2c of Definition 2.*

Lemma 2 (Main Technical Lemma) *Let Σ be a non-trivial protocol using which parties p_1, p_2 can realize the ideal functionality $\mathcal{G}$ for a polynomial time computable function g as defined above. Let protocol Σ be fully secure in the $\mathcal{F}_{CRS}$-hybrid model and secure against a dishonest p_2 in the $\mathcal{F}_{mCRS}$-hybrid model. Then there exists a machine p_2^a such that for every machine p_2^b of the form described in Definition 2, the split adversarial strategy $p_2 = (p_2^a, p_2^b)$ is* successful *in the plain model, except with negligible probability.*

Proof The proof in our setting is very similar to the proof of main technical lemma in the plain model [8]. Here we only give a sketch of our proof highlighting the main difference from the one for plain model. We refer the reader to [8] for complete details.

Recall that in the proof for plain model, we first consider the real world process where the party p_1 is controlled by the environment $\mathcal{Z}$ through a dummy adversary $\mathcal{A}_D$ who simply forwards messages received from the environment to party p_2 and answers received from p_2 to the environment. Parties p_1 and p_2 have inputs I_1 and I_2 respectively and execute Σ; we assume that Σ securely realizes $\mathcal{G}$. Thus, there exists a simulator $\mathcal{S}$ that interacts with the ideal process such that $\mathcal{Z}$ cannot distinguish an execution of a real world process from an execution of the ideal process. Notice that in the ideal world, $\mathcal{S}$ must send an input I'_1 to the functionality $\mathcal{G}$ and receive an output J'_1 from $\mathcal{G}$ such that I'_1 and J'_1 are functionally equivalent to I_1 and $g_1(I_1, I'_2)$ respectively. Here I'_2 is chosen by p_2. This implies that if $\mathcal{Z}$ simply runs an honest p_1 inside, $\mathcal{S}$ is able to *extract the inputs of the honest player* p_1 and also force its output to be J'_1.

In our setting, in the $\mathcal{F}_{\text{CRS}}$-hybrid model (i.e., if the string σ is an honestly generated CRS), the protocol Σ is fully secure. This means that there exists a simulator $\mathcal{S}_\tau$ which, when given access to the trapdoor information τ on the CRS σ, is able to extract the input of the honest player p_1 in the $\mathcal{F}_{\text{CRS}}$-hybrid model.

Now consider the case of $\mathcal{F}_{mCRS}$-hybrid model, i.e., when Σ was compiled with an adversarially-generated string σ. In this case, a malicious p_2 in the possession of the trapdoor information τ is able to initialize $\mathcal{S}_\tau$ and have it interact with the honest p_1. At a high level, the machine p_2^a

just consists of running the simulator $\mathcal{S}_\tau$ with honest p_1. Machine p_2^a forwards every message that it receives from p_1 to $\mathcal{S}_\tau$ as if it came from $\mathcal{Z}$. Similarly every message that $\mathcal{S}_\tau$ sends to $\mathcal{Z}$, p_2^a forwards to p_1 in the real execution. When $\mathcal{S}_\tau$ outputs a value I_1' that it intends to send to $\mathcal{G}$, p_2^a gives it to p_2^b. Later, when p_2^b gives a value J_1' to p_2^a, p_2^a gives it to $\mathcal{S}_\tau$ as if it came from $\mathcal{G}$. Hence, a malicious p_2 in the possession of τ is able to use the simulator $\mathcal{S}_\tau$ to do whatever the simulator $\mathcal{S}$ was doing in the plain model. This in particular means that p_2 is able to extract the input of the honest p_1 and run a *successful* split simulator strategy. This completes our proof sketch. ■

Completing the proof of theorem 2. As shown by [8], the existence of a successful split simulator strategy for p_2 against an honest p_1 rules out realization of several interesting well formed functionalities g. This in turn rules out several n-input functionalities f whose secure computation implies secure computation of g. We give a concrete example in the following.

We consider *single-input functions which are not efficiently invertible* [8]. The definition of efficiently invertible functions is given as [8]:

Definition 5 *A polynomial-time function $g : D \to \{0,1\}^*$ is efficiently invertible if there exists a PPT invertible machine M such that for every distribution $\hat{D} = \{\hat{D}_k\}$ over D that is samplable in PPT by a non-uniform Turing machine, every polynomial $p(\cdot)$ and all sufficiently large k's*

$$Pr_{x \leftarrow \hat{D}_k}[M(1^k, g(x)) \in g^{-1}(g(x)] > 1 - \frac{1}{p(k)}$$

Let t, s, n be such that $t + s = n$ and $s < n/2$. Let there be n-parties $P_1, \ldots, P_n$. We consider the following functionality $\mathcal{F}$. Let parties $P_1, \ldots, P_t$ hold inputs $x_1, \ldots, x_t$ while $P_{t+1}, \ldots, P_n$ have no inputs. The output of $P_1, \ldots, P_t$ is $\perp$ while the output of $P_{t+1}, \ldots, P_n$ is $f(x_1, \ldots, x_t)$. $f(x_1, \ldots, x_t)$ is defined as $g(x_1 \| \ldots \| x_t)$ where g is a single input function which is not efficiently invertible (see definition above). Let there exist a protocol Π which (s,t)-securely realizes $\mathcal{F}$.

Now we define the functionality $\mathcal{G}$ as follows. Let party p_1 hold input $I_1 = x_1 \| \ldots \| x_t$ while p_2 has no input. The output of p_1 is $\perp$ while the output of p_2 is $g(I_1)$. Given Π, Lemma 1 implies the existence of a two party protocol Σ to realize $\mathcal{G}$ which is fully secure in the $\mathcal{F}_{\text{CRS}}$-hybrid model and secure against a malicious p_2 in the $\mathcal{F}_{mCRS}$-hybrid model. However, Lemma 2 implies that p_2 can run a successful split simulator strategy and extract an input I_1' such that $g(I_1) = g(I_1')$. Since all the information learnt by p_2 during the execution of Σ should follow from its output $g(I_1)$ alone, it follows that a value I_1' such that $g(I_1) = g(I_1')$ is computable given $g(I_1)$. This contradicts the non-invertibility of function g.

Hence, we conclude that there does not exist such a protocol Π to evaluate the functionality $\mathcal{F}$. This proves theorem 2. This impossibility result can be extended to include a large class of functionalities as in [8].

References

[1] B. Barak, R. Canetti, J.B. Nielsen, and R. Pass. Universally composable protocols with relaxed set-up assumptions. In *Proc. 45th FOCS*, pages 186–195, 2004.

[2] B. Barak and A. Sahai. How to play almost any mental game over the net — concurrent composition using super-polynomial simulation. In *Proc. 46th FOCS*, 2005.

[3] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *STOC*, 1988.

[4] M. Blum, P. Feldman, and S. Micali. Non-interactive zero-knowledge and its applications. In *STOC*, 1988.

[5] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proc. 42nd FOCS*, pages 136–147, 2001. Preliminary full version available as Cryptology ePrint Archive Report 2000/067.

[6] R. Canetti, Y. Dodis, R. Pass, and S. Walfish. Universally composable security with global setup. In *TCC*, 2007.

[7] R. Canetti and M. Fischlin. Universally composable commitments. Report 2001/055, Cryptology ePrint Archive, July 2001. Extended abstract appeared in CRYPTO 2001.

[8] R. Canetti, E. Kushilevitz, and Y. Lindell. On the limitations of universally composable two-party computation without set-up assumptions. *J. Cryptology*, 19(2):135–167, 2006.

[9] R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. Universally composable two-party and multi-party secure computation. In *Proc. 34th STOC*, pages 494–503, 2002.

[10] S. Goldwasser and Y. Lindell. Secure multi-party computation without agreement. *J. Cryptology*, 18(3):247–287, 2005.

[11] J. Groth and R. Ostrovsky. Cryptography in the multi-string model. In *Crypto 2007.*

[12] D. Hofheinz, J. Müller-Quade, and D. Unruh. Universally composable zero-knowledge arguments and commitments from signature cards. In *Proc. 5th Central European Conference on Cryptology*, 2005.

[13] Y. Ishai, E. Kushilevitz, Y. Lindell, and E. Petrank. On combining privacy with guaranteed output delivery in secure multiparty computation. In *Crypto 2006.*

[14] J. Katz. On achieving the "best of both worlds" in secure multiparty computation. In *STOC 2007.*

[15] J. Katz. Universally composable multi-party computation using tamper-proof hardware. In *Eurocrypt 2007.*

[16] M. Prabhakaran and A. Sahai. New notions of security: Achieving universal composability without trusted setup. In *Proc. 36th STOC*, pages 242–251, 2004.

[17] T. Rabin and M. Ben-Or. Verifiable secret sharing and multi-party protocols with honest majority. In *STOC*, 1989.