Bringing People of Different Beliefs Together to Do UC

Sanjam Garg[1], Vipul Goyal[2], Abhishek Jain[1], and Amit Sahai[1]

[1] UCLA
{sanjamg,abhishek,sahai}@cs.ucla.edu
[2] Microsoft Research, India
vipul@microsoft.com

Abstract. Known constructions of UC secure protocols are based on the premise that different parties collectively agree on some trusted setup. In this paper, we consider the following two intriguing questions: Is it possible to achieve UC if the parties do not want to put all their trust in *one* entity (or more generally, in one setup)? What if the parties have a difference of opinion about what they are willing to trust? The first question has been studied in only a limited way, while the second has never been considered before.

In this paper, we initiate a systematic study to answer the above questions. We consider a scenario with multiple setup instances where each party in the system has some individual *belief* (setup assumption in terms of the given setups). The belief of a party corresponds to what it is willing to trust and its security is guaranteed given that its belief "holds." The question considered is: "Given some setups and the (possibly) different beliefs of all the parties, when can UC security be achieved?" We present a general condition on the setups and the beliefs of all the parties under which UC security is possible. Surprisingly, we show that when parties have different beliefs, UC security can be achieved with a more limited "trust" than what is necessary in the traditional setting (where all parties have a common belief).

1 Introduction

Suppose Alice and Bob want to execute a UC-secure [4] protocol. They know that they will need to rely on some trust assumptions [6,7] in order to achieve UC security. Unfortunately, Alice and Bob have different beliefs about who is trustworthy: Suppose that Microsoft, Intel, Google, and Yahoo have all published common reference strings (CRS). Alice believes that either both Microsoft and Intel are trustworthy, or both Google and Yahoo are trustworthy. Bob, however, has a different view: Bob believes that either both Microsoft and Google are trustworthy, or both Intel and Yahoo are trustworthy. This seems like a horrible situation. Indeed, even if both Alice and Bob shared Alice's trust belief (or if they both shared Bob's trust belief), most UC-secure protocols would be impossible [13]. We show, surprisingly, that nevertheless in the situation above with

Y. Ishai (Ed.): TCC 2011, LNCS 6597, pp. 311–328, 2011.

asymmetric beliefs, Alice and Bob can execute a protocol that will guarantee UC security for *all parties whose trust beliefs turn out to be valid*. This paper is a systematic study of this problem – when can we guarantee UC security (in this sense) even when different parties have different trust beliefs.

Background. The last decade has seen a push towards obtaining secure computation protocols in the demanding network setting where there might be multiple concurrent protocol executions. The framework of universal composability (UC) was introduced by Canetti [4] to capture the security requirements in such a setting. However unfortunately, soon after the introduction of the UC framework, impossibility results were shown ruling out the existence of UC secure protocol for most functionalities of interest [6,7]. These results were further generalized [18,2] to rule out the existence of secure protocol even in various less demanding settings. These impossibility results refer to the "plain model" where the participating parties do not trust any external entity and they have no prior communication among themselves, etc.

To overcome these deep impossibility results and obtain secure protocols in the modern network setting, a number of different "setup assumptions" were introduced. A few examples follow. Canetti and Fischlin [6] and Canetti, Lindell, Ostrovsky and Sahai [8] consider the model where a trusted party publishes a "common reference string" (CRS). Canetti, Pass and Shelat [9] generalized these results by considering reference strings coming from an unknown distribution. Barak, Canetti, Nielsen and Pass [1] introduced the so called "registered public key" model; a variant of this model was considered by Canetti, Dodis, Pass and Walfish [5]. Katz [14] (and subsequently [10,11,19]) studied a model where the parties exchange tamper proof hardware tokens with each other. Under all of these settings, general positive results for all PPT computable functionalities have been shown in the UC framework. In addition, such positive results can also be obtained in the setting where a majority of the participants are assumed to be honest [3,4,15][1].

Now that there are a number of different options in what one might "assume about the world" to obtain UC protocols, the parties (or the system designer) would be forced to choose one. In some situation, such a choice would be natural from the environment where the protocols would be run. However in many other scenarios, it would be unclear which setup assumption is the right one: should the parties trust a CRS published by an authority or should they each register a public key with an authority? Should they assume that a majority of them are honest or should they not trust anyone and instead rely on tamper proof hardware tokens? Making such a choice can be non-trivial since a wrong choice could lead to a complete compromise of the system security.

In light of the above, we can consider the following two lines of thought. First, given the importance of making a correct choice, it is not unreasonable to imagine

[1] We slightly abuse the terminology and consider the honest majority setting as just another setup. Our study is targeted at assumptions which allow one to obtain UC protocols but can go wrong leading to a security compromise. Honest majority is one such assumption.

that in certain scenarios, the parties in the system cannot agree on a common choice, i.e., it is plausible that each party may have a different choice about what setup to use. For example, one party may want to use a CRS published by Microsoft while another may want to register keys with Google. Yet another party might want to place more trust in tamper-proofness of certain hardware tokens and not so much in any entity, and so on. In essence, each party may have a different trust assumption (referred to as the "belief" of the party) about the setups available. This gives rise to the following question:

Is it possible to construct UC protocols when parties have different beliefs about setups?

An orthogonal line of thought is that since a wrong choice could lead to a compromise of the system security, it might be desirable to diversify the risks involved and avoid a single point of failure. In other words, how about basing the protocol on a combination of setups rather than a single one? As an example, the parties might have access to a published CRS as well as have registered public keys with an authority. The protocol should retain its security even if one of these setup assumptions "breaks down" (for example, if the published CRS turned out to be adversarially chosen) but the other turned out to be "honestly chosen." Going a little further, there might be n instances of various setups and the protocol security should hold as long as (e.g.) either one of the first two or a majority of the rest were honestly done. More generally, we can consider the following question:

Is it possible to construct UC protocols using multiple setups when the parties share an arbitrary belief about the setups?

In this paper, we answer both the above questions in the affirmative. We remark that a general perception following the prior work on UC security is that a *common* trusted setup is necessary for achieving UC. As our results show, this is in fact *not* necessary. We further note that one would expect UC to be harder to achieve when parties may have asymmetric beliefs. On the contrary, we show that the level of trust needed to obtain UC can be significantly weakened in such a setting.

Related Work. There have been two previous works in the direction of basing UC secure protocols based on multiple setups. However the works have been much narrower in scope. Groth and Ostrovsky [13] consider the question of basing cryptography on multiple CRS. They showed how to construct UC protocols under the assumption that a majority of the given reference strings were honestly generated. Subsequently, Goyal and Katz [12] considered basing UC secure protocol under a combination of a CRS and the honest majority assumption.

These questions can be viewed as two special cases of our general question (in particular, the case of common belief). Our work subsumes these works and provides answers to a host of other such interesting questions.

Our Results. In this paper, we initiate a systematic study of the question of basing UC protocols on multiple different setups in a setting where each party has its own belief (setup assumption in terms of the given setups) about these setups. An informal problem statement is as follows. Consider a system with multiple setups and parties. Each party has a belief expressed as an arbitrary monotonic formula expressed in disjunctive normal form in terms of the setups, e.g. $(A \wedge B) \vee (C \wedge D)$ where A, B, C and D are some setups. We interpret this belief as– "Either setup A and setup B hold, or setup C and setup D hold." We ask the following question: "Given these setups and the different beliefs of all the parties, when can UC security be achieved?" In answering this question, we give a very general condition on the setups and the different beliefs of all the parties under which UC security is possible. This result is presented in Section 5.

Towards the goal of answering the above question, we first look at a simpler scenario in which all the parties have a "common belief," i.e., all the parties share the same belief about the setups in the system. We give a very general condition on the setups and the common belief of all the parties under which UC security is possible. This result is presented in Section 4.

As we discuss later in Section 2, a setup may be "corruptible" in multiple ways; as such, parties might be willing to put trust in the "extent" of corruption of a particular setup. Furthermore, different parties might be willing to put different levels of trust in the same setup. The results of Section 4 and 5 handle this case to some extent, but this scenario is handled in its full generality in Appendix B. We advise the reader to look at some of the interesting examples presented there.

In order to argue about security of a system with arbitrary setups, we abstract formal properties that essentially capture what "extra powers" a setup provides to a simulator over an adversary. These abstract properties allow us to categorize setups and argue UC security of protocols that can be constructed from them. We note that all known setups fit these definitions very well. We further note that our results generalize the previously known tight results of Groth and Ostrovsky [13] and Goyal and Katz [12]. Finally, we leave it as an open problem to study the tightness of our results in the general case.

Overview of Main Ideas. In past, UC protocols for different setups have been designed with very different techniques. Here, we wish to design a single protocol that simultaneously uses a combination of a number of different setups. Our starting point is the recent work of Lin, Pass and Venkitasubramaniam [16] which puts forward a unified framework for designing UC secure protocols. In the UC framework, to obtain positive results, the simulator is required to obtain some "extra power" over the adversary. Lin, Pass and Venkitasubramaniam observe that a general technique for constructing UC secure protocols is to have the simulator obtain a "trapdoor string" which is hard to compute for the adversary. This is formalized in the form of (two party) UC-puzzle protocols that enable the simulator to obtain such a trapdoor string (but prevent the adversary from doing so). Such a trapdoor string is already available to the simulator in the CRS

model and hence designing a UC-puzzle is trivial in that case. However, even in case of other setup assumptions, Lin, Pass and Venkitasubramaniam show that generally it is possible to easily design such a UC-puzzle at the end of which the simulator obtains a trapdoor string (but the adversary does not).

Once we have a unified construction for UC protocols under different setup assumptions, we come to our main question: how do we fruitfully use multiple setups in a single protocol? Different setups might compose very differently with each others. A priori, it might seem that each pair of setups may compose in a unique way. Hence, considering the general question that we wish to study, it seems unclear how to proceed at all.

A key conceptual contribution of our work is a classification of the setup assumptions used to construct UC protocols. We observe that almost all setups can be classified among three types. To understand these different types, recall that in this work, we are concerned with unreliable setup assumptions that may actually turn out to be false. Coming back to the framework of Lin et al [16], our simulator would obtain a trapdoor string which would be hard to compute for the adversary (this is of course if the setup was "honest"). However what happens if the setup assumption was actually false (i.e., setup was "malicious")? We could have any one of the following three cases: (I) the adversary is able to obtain the trapdoor string (associated with the UC-puzzle) but not the simulator, (II) none of them are able to obtain the trapdoor string, and, (III) both the adversary and the simulator are able to obtain the trapdoor string[2]. Intuitively, the first case corresponds to *complete* corruption of the setup, while the other two cases correspond to *partial* corruption of the setup.

We are able to show that the above classification of setups solely decides the composability properties of different setups. In general, Type II and Type III setups have better composability properties than the Type I ones. For instance, in the special case where we have multiple instances of the same setup, a majority of them should be "honest" if the setup is of Type I. However if it is either of Type II or Type III, it is sufficient to have a *single* "honest" setup.

Going further, we note that following the work of Lin et al [16], the task of constructing UC secure protocols from any setup assumption reduces to the task of constructing a UC-puzzle (in the hybrid model of the corresponding setup). Then, in a scenario where all the parties share a common belief about the setups in the system, the task of constructing UC protocols reduces to task of constructing a UC-puzzle in the hybrid model of the multiple setups in the system. But what of the scenario where the parties have different beliefs about the setups? In this case, we show how to construct a *family* of UC-puzzles that in turn can be used to construct a family of "concurrent simulation-sound" zero knowledge protocols with "UC-simulation" property. For reasons discussed later in section 3 and 5, this is sufficient to construct UC protocols in such a setting.

Organization. We start by describing our model in Section 2. In Section 2.1, we recall the notion of UC-puzzles [16] and give their classification into various

[2] The fourth case is uninteresting as we argue later.

types. In Section 2.2, we give a classification of setups into types. We then present our positive result for the case where parties share a common belief in Section 4. Next, in Section 5, we present our positive result for the case where parties have different beliefs. Finally, in Appendix A and B, we extend our results to cover some more involved settings.

2 Our Model

Traditionally, protocols that utilize a single instance of a setup have been constructed, and proven to be universally composable as long as the setup is "honest" (i.e., the setup assumption holds). We, however, consider settings where a protocol may utilize more than one setup; in such a setting, one or more setups may in fact be "corrupted" (i.e., the setup assumption corresponding to a setup no longer holds because of possible control of the setup by an adversary). We wish to investigate when UC-security can be realized in such settings. To this end, we first consider an augmented modeling for setups (that could either be "honest" or "corrupted").

Modeling Setup Failure. Typically, a setup is modeled as a "trusted" ideal functionality that interacts with the parties in the system. Let $\mathcal{G}$ denote such an ideal functionality. In order to account for the scenario that a setup could in fact be corrupted by an adversary, we augment this model by considering another ideal functionality $m\mathcal{G}$ that represents a "malicious" version of $\mathcal{G}$. We refer to $m\mathcal{G}$ as a *failure mode* of $\mathcal{G}$. Then, we will model a setup as a pair $(\mathcal{G}, m\mathcal{G})$ of ideal functionalities, where $\mathcal{G}$ represents to the honest version of the setup while $m\mathcal{G}$ represents its failure mode.

For example, consider the common reference string (CRS) setup [6,8]. In previous works, the CRS setup is modeled as a trusted ideal functionality that samples a CRS from a specified distribution. A party in the system can query the ideal functionality, who in response, will return the CRS to the party. In our setting, we will model the CRS setup as a pair $(\mathcal{G}_{CRS}, m\mathcal{G}_{CRS})$ of ideal functionalities. Here $\mathcal{G}_{CRS}$ corresponds to the case where the CRS is generated honestly by the ideal functionality, such that no adversary can obtain any "trapdoor" information for the CRS. On the other hand, $m\mathcal{G}_{CRS}$ corresponds to the case where the functionality returns an adversarially chosen CRS; in particular, an adversary may be able to obtain some "trapdoor" information for the CRS.

It should be implicit that we only consider setups that are "sufficient" to realize UC-secure protocols. That is, we assume that given any setup $(\mathcal{G}, m\mathcal{G})$, it is possible to construct UC-secure protocols in the $\mathcal{G}$-hybrid model. We further assume that constructing UC-secure protocols is impossible in the $m\mathcal{G}$-hybrid model.

Multiple Failure Modes. The above modeling of setups is not quite complete, in that it is too restrictive to imagine that a setup may only have a single failure mode. Specifically, one could imagine a setup failing in multiple ways depending upon how it is corrupted by an adversary. For instance, let us consider the

tamper-proof hardware token setup of Katz [14]. In the model of Katz, it is assumed that (a) parties can exchange tamper-proof hardware tokens with each other (b) a token "creator" cannot send messages to the token after giving the token to another party. This is modeled in the form of a wrapper functionality that takes a program code as an input from a party and then uses that code to "emulate" a token that interacts with the intended receiver. In this case, one can consider different possible corruptions of the wrapper functionality (each corresponding to a different failure mode $m\mathcal{G}$) where either or both of the above assumptions fail. In general, for a given honest version $\mathcal{G}$ of a setup, there may be multiple failure modes $m\mathcal{G}_1, m\mathcal{G}_2, \ldots, m\mathcal{G}_k$.

In the sequel, for simplicity of exposition, we will first restrict ourselves to the case where a setup only has a single failure mode. The results presented in Section 4 and 5 are obtained under this restriction. Later in Appendix B, we discuss how to extend our modeling to incorporate multiple failure modes and then explain how our results can be extended to this case.

Setup Types. Now recall that one of the main goals of this paper is to provide a way to construct UC secure protocols that utilize multiple different setups. A priori, it might seem that different setups may compose very differently with each other *depending upon their specific properties*, and that in the worst case, each pair of setups might compose in a unique way. However, we show that this is not the case; specifically, we give a classification of setups into different "Types," and then show that the Type of any setup solely decides the composability properties of that setup. We then study composition of setup types and give positive results for the feasibility of constructing UC-secure protocols in the presence of multiple setups (of possibly different Types).

Our classification of a setup into Types is based on the feasibility of constructing a specific primitive called "UC-puzzle" (in the hybrid model of the setup). We first recall the notion of UC puzzles in Section 2.1. Later, in Section 2.2, we give a classification of setups into Types.

2.1 UC Puzzles and Their Classification

Lin et al [16] introduced the notion of UC-puzzles, where, informally speaking, a UC-puzzle is a two-party protocol in a $\mathcal{G}$-hybrid model (where $\mathcal{G}$ is a trusted ideal functionality)[3] such that no real world adversary can complete the puzzle and also obtain a "trapdoor," while there exists a simulator that can "simulate" a puzzle execution (with the "correct" distribution) and also obtain a trapdoor. Looking ahead, our positive results rely crucially on UC-puzzles; therefore, we discuss them below in detail.

Recall that in our setting, setups are "corruptible." For instance, in the above example, the setup functionality in the system may in fact be $m\mathcal{G}$ (instead of $\mathcal{G}$) which is controlled by an adversary. Clearly, there may be no guarantee that

[3] We note that in the original definition of Lin et al [16], the existence of a an ideal functionality $\mathcal{G}$ is not compulsory. However, in such cases, one can imagine $\mathcal{G}$ to be an empty functionality.

the aforementioned properties of a UC-puzzle still hold in this case; as such, the original definition of [16] does not suffice for our purposes. To this end, we extend the original definition of UC-puzzles to account for such a scenario. We give an informal definition of a UC-puzzle as follows. Part of the definition below is taken almost verbatim from [16].

UC-puzzle. Let $(\mathcal{G}, m\mathcal{G})$ be a setup. A UC-puzzle is a pair $(\langle S, R\rangle, \mathcal{R})$, where $\langle S, R\rangle$ is a protocol between two parties—a sender S, and a receiver R—in the $\mathcal{F}$-hybrid model (where $\mathcal{F}$ is either $\mathcal{G}$ or $m\mathcal{G}$), and $\mathcal{R} \subseteq \{0,1\}^* \times \{0,1\}^*$ is an associated PPT computable relation.

I. If $\mathcal{F}$ is the "honest" ideal functionality $\mathcal{G}$, i.e., the puzzle is in the $\mathcal{G}$-hybrid model, then it must satisfy the following two properties.

Soundness. No PPT adversarial receiver R^* after an execution with an honest sender S can find (except with negligible probability) a trapdoor $\sigma \in \mathcal{R}(\texttt{trans})$, where `trans` is the transcript of the puzzle execution.

Statistical Simulatability. Let $\mathcal{A}$ be a real world adversary (in an environment $\mathcal{Z}$) that participates as a sender in multiple concurrent executions of a UC-puzzle. Then, for every such $\mathcal{A}$, there exists a simulator $\mathcal{S}$ interacting only with $\mathcal{Z}$ such that no (possibly unbounded) $\mathcal{Z}$ can distinguish between an execution with $\mathcal{A}$ from an execution with $\mathcal{S}$, except with negligible probability. Further, for every completed puzzle execution, except with negligible probability, $\mathcal{S}$ outputs a trapdoor $\sigma \in \mathcal{R}(\texttt{trans})$, where `trans` is the transcript of that puzzle execution.

II. Otherwise, if $\mathcal{F}$ is the "malicious" functionality $m\mathcal{G}$, i.e., the puzzle is in the $m\mathcal{G}$-hybrid model, then we consider three sub-cases and define different "types" for UC-puzzles. In order to describe these cases, we first define two properties that can be seen as "strong" negations of the two aforementioned properties of UC-puzzles[4].

Unsound. A UC-puzzle $(\langle S, R\rangle, \mathcal{R})$ is said to be Unsound in the $m\mathcal{G}$-hybrid model if there exists a PPT adversarial receiver R^* that can find (except with negligible probability) a trapdoor $\sigma \in \mathcal{R}(\texttt{trans})$, for a puzzle execution (with a transcript `trans`) with an honest sender S.

Unsimulatable. Further, we say that the UC-puzzle is Unsimulatable in the $m\mathcal{G}$-hybrid model if there exists an adversarial sender $\mathcal{A}$ such that no simulator $\mathcal{S}$ can obtain (except with negligible probability) a trapdoor $\sigma \in \mathcal{R}(\texttt{trans})$ for a completed puzzle execution (where `trans` is the transcript of the puzzle execution).

[4] An intuitive explanation of the two new properties can be seen as follows. Informally speaking, we wish to say that a UC-puzzle in the $m\mathcal{G}$-hybrid model is Unsound if an adversary in the $m\mathcal{G}$-hybrid model enjoys the same power as any simulator in the $\mathcal{G}$-hybrid model. Similarly, a UC-puzzle in the $m\mathcal{G}$-hybrid model is Unsimulatable if a simulator in the $m\mathcal{G}$-hybrid model enjoys no extra power as compared to an adversary in the $\mathcal{G}$-hybrid model.

We now consider the following cases.

Type I. We say that a UC-puzzle is of Type I if it is Unsound and Unsimulatable in the $m\mathcal{G}$-hybrid model.

Type II. We say that a UC-puzzle is of Type II if it satisfies Soundness but is Unsimulatable in the $m\mathcal{G}$-hybrid model.

Type III. We say that a UC-puzzle is of Type III if it is Unsound but satisfies Statistical Simulatability in the $m\mathcal{G}$-hybrid model.

Finally, we can consider the case where both Soundness and Statistical Simulatability are satisfied in the $m\mathcal{G}$-hybrid model. We discard this case for the following reasons. Note that this case implies that we can construct UC-puzzles even in the $m\mathcal{G}$-hybrid model. Then, from the result of [16], it would follow that we can construct UC-secure protocols even in the $m\mathcal{G}$-hybrid model. Informally speaking, this means that the setup was not corrupted in any "interesting" way.

This completes our definition of UC-puzzles and their classification into Types.

2.2 Classification of Setups

Having defined different Types of UC-puzzles, we are now ready to define the Type of a setup based on the feasibility of constructing UC-puzzles in the hybrid model of that setup.

Definition 1 (Setup Types). *A setup $(\mathcal{G}, m\mathcal{G})$ is said to be of Type X if it is possible to construct a UC-puzzle of Type X in the $\mathcal{F}$-hybrid model, where $\mathcal{F}$ is either $\mathcal{G}$ or $m\mathcal{G}$, and $X \in \{I, II, III\}$.*

Setups with multiple Types. Note that it may be possible to construct multiple UC-puzzles of different Types in the hybrid model of a setup $(\mathcal{G}, m\mathcal{G})$; as such, the above definition allows a setup $(\mathcal{G}, m\mathcal{G})$ to have multiple Types. For simplicity of exposition, in the sequel, we will first assume that each setup has a unique Type. The results presented in Section 4 and 5 are obtained under this restriction. Later, in Appendix A, we give an example of a custom setup that has multiple types, and then explain how to extend our results to incorporate setups with multiple Types.

Classification of known setups. We now briefly discuss some known setups such as CRS [CF01, CLOS02, CPS07], tamper-proof hardware [Kat07], and key registration [BCNP04]. We first note that the only "natural" failure mode for the CRS setup corresponds to "complete corruption," where the CRS is chosen by the adversary. Then, it is not difficult to see that the CRS setup is of Type I. In contrast, the key registration setup and the hardware-token setup naturally allow "partial corruption". Let us first consider the key registration setup. Recall that in the key registration setup, it is assumed that parties can register their public keys in such a way that: (a) the public keys of the honest parties are "safe" (in the sense that the secret keys were chosen at random and kept secret from the adversary), and (b) the public keys of the corrupted parties are 'well-formed" (in the sense that the functionality has seen the corresponding secret

keys). Then, an adversary may be able to corrupt the setup in multiple ways (each corresponding to a different failure mode) such that either or both of these assumptions are violated. The first failure mode corresponds to the complete corruption of the setup (i.e., both the above assumptions fail); in this case, the setup is of Type I. The second failure mode corresponds to the case where the public keys of corrupt parties may not be "well-formed"; however, the secret keys of the honest parties are still "safe." In this case, the setup is of Type II. Finally, in the third failure mode, the secret keys of honest parties may not be "safe"; however, the public keys of corrupted parties are still "well-formed." In this case, the setup is of Type III. In a similar way, one can consider different failure modes for the hardware token setup, each leading to a different Type. We refer the reader to the full version of the paper for details.

3 UC Security via UC-Puzzles

As observed by Lin et al [16], it is implicit from prior works [8,17,21,20] that the task of constructing UC-secure protocols for any well-formed[5] functionality reduces to the task of constructing a "concurrent simulation-sound" zero knowledge protocol (ssZK) with "UC simulation" property[6]. Very informally, these properties can be described as follows (the text is taken almost verbatim from [16]):

UC simulation: For every PPT adversary $\mathcal{A}$ receiving "honest" proofs of statements x using witness w, where (x, w) are chosen by the environment $\mathcal{Z}$, there exists a simulator $\mathcal{S}$ (that only gets statements x as input) such that no $\mathcal{Z}$ can distinguish (except with negligible probability) whether it is interacting with $\mathcal{A}$ or $\mathcal{S}$.

Concurrent simulation-soundness: An adversary who receives an unbounded number of concurrent *simulated* proofs, of statements chosen by $\mathcal{Z}$, cannot prove any false statements (except with negligible probability).

Lin et al [16] gave a unified framework for constructing UC-secure protocols from known setup assumptions like CRS [6,8], tamper-proof hardware tokens [14], key registration [1], etc. Specifically, Lin et al [16] gave a construction for an ssZK protocol from a UC-puzzle in a $\mathcal{G}$-hybrid model (where $\mathcal{G}$ is a "trusted" ideal functionality that may correspond to, for instance, a CRS setup), and a strongly non-malleable witness indistinguishable (SNMWI) argument of knowledge (see [16] for details).

We note that following the work of [16], the task of constructing UC secure protocols from any setup assumption reduces to the task of constructing a UC-puzzle (in the hybrid model of the corresponding setup)[7]. Then, looking ahead, the positive results in this paper crucially rely on the framework of [16].

[5] We refer the reader to [8] for a definition of "well-formed" functionalities.

[6] Formally, this can be modeled as implementing a specific "zero knowledge proof of membership" functionality.

[7] Note that [16] gave a construction of an SNMWI protocol based on one-way functions.

4 Common Belief about the Setups

We first consider the setting where all the parties in the system share a *common*, though arbitrary, belief about the setups present in the system. For instance, consider an example where there are three CRS setups in the system. In this case, all the parties may share a common belief that either (a) the first CRS is honestly generated, or (b) both the second and the third CRSs are honestly generated. Ideally, given any function f, we would like to construct a protocol Π (in the hybrid model of the three CRSs) that securely realizes f when either of the above two cases is actually true. In this section, we investigate the possibility of constructing such protocols. In particular, we will give a condition that takes into account the setups present in the system and the common belief shared by the parties; we then show that constructing secure protocols (where security is defined in the above sense) is possible if our condition is satisfied. We first introduce some notation and definitions.

Belief Set. In the above example, we can express the common belief of the parties in the form of a DNF formula written as CRS_1 OR (CRS_2 AND CRS_3), where CRS_i denotes the i^{th} CRS. The DNF formula, in turn, can be represented as a set $\Sigma = \{T_1, T_2\}$ where T_1 is a set that consists of CRS_1, and T_2 is a set that consists of the CRS_2 and CRS_3.

We generalize this in the following manner. Let U be the set of all the setups present in the system. Then, we will represent the common belief of the parties as a set $\Sigma = \{T_1, \ldots, T_k\}$ (where k is an arbitrary, possibly exponential, value), where each member T_i is a subset of U. The common belief of the parties is expressed as follows: $\exists T_i \in \Sigma$ such each setup in T_i holds[8]. We will refer to this set Σ as the *belief set*.

Σ-secure protocols. We would like to construct secure computation protocols that realize UC security if the common belief of the parties about the setups in the system is actually true. We formalize this in following definition of *Σ-secure* protocols.

Definition 2 (Σ-secure protocols). *Let there be n setups in the system denoted by $(\mathcal{G}_1, m\mathcal{G}_1), \ldots, (\mathcal{G}_n, m\mathcal{G}_n)$. Let $\mathcal{F}_1, \ldots, \mathcal{F}_n$ be n ideal functionalities, where $\forall i, \mathcal{F}_i$ is either $\mathcal{G}_i$ or $m\mathcal{G}_i$. Let $\Sigma = \{T_1, \ldots, T_k\}$ be a belief set over the n setups that represents the common belief of the parties. We say that a protocol Π Σ-securely realizes a functionality f in the $(\mathcal{F}_1, \ldots, \mathcal{F}_n)$-hybrid model if Π UC-securely realizes f in the $(\mathcal{F}_1, \ldots, \mathcal{F}_n)$-hybrid model whenever $\exists T_i \in \Sigma$ such that each setup in T_i holds (i.e., $\mathcal{F}_j$ is the ideal functionality $\mathcal{G}_j$, for each setup $(\mathcal{G}_j, m\mathcal{G}_j) \in T_i$).*

Remark. We note that our security definition does not immediately comply with the traditional UC framework where setups are "incorruptible". To this

[8] Here, and throughout the text, it should be implicit that whenever we say a setup X holds, what we actually mean is that the *setup assumption* corresponding to setup X holds.

end, we consider a simple modification to the UC framework where the adversary is allowed to choose (before the start of the protocol) the setups that she wishes to corrupt. Of course, this information is not known to the protocol designer, since in that case, achieving UC security is easy – the parties could simply ignore the corrupt setups and use only the honest ones.

UC-compatible Belief Sets. Before we discuss our positive result on constructing Σ-secure protocols, we first define the notion of a *UC-compatible* belief set, that is central to our result.

Definition 3 (UC-compatible belief set). *A belief set $\Sigma = \{T_1, T_2 \dots T_k\}$, where $\forall i \in [k]$, $T_i \neq \emptyset$, is said to be UC-compatible if $\forall i, j \in [k]$, at least one of the following conditions hold:*

- *$T_i \cap T_j \neq \emptyset$.*
- *Both T_i and T_j contain at least one (not necessarily the same) Type II setup.*
- *Both T_i and T_j contain at least one (not necessarily the same) Type III setup.*
- *Either T_i or T_j contains at least one Type II setup as well as at least one Type III setup.*

We are now ready to state our main result for the common belief case.

Theorem 1 (Main result for common belief case). *Let there be n setups in the system denoted by $(\mathcal{G}_1, m\mathcal{G}_1), \dots, (\mathcal{G}_n, m\mathcal{G}_n)$. Let $\mathcal{F}_1, \dots, \mathcal{F}_n$ be n ideal functionalities, where $\forall i, \mathcal{F}_i$ is either $\mathcal{G}_i$ or $m\mathcal{G}_i$. Let Σ be a belief set over the n setups that represents the common belief of the parties. If Σ is UC-compatible, then for every well-formed functionality f, there exists a non-trivial protocol Π that Σ-securely realizes f in the $(\mathcal{F}_1, \dots, \mathcal{F}_n)$-hybrid model.*

Proof (Sketch). As noted in Section 3, it follows from the work of [16] that the task of constructing UC secure protocols from any setup assumption reduces to the task of constructing a UC-puzzle (in the hybrid model of the corresponding setup). Then, we note that in order to prove Theorem 1, it suffices to construct a puzzle $\langle S, R \rangle$ with an associated relation $\mathcal{R}$ such that $(\langle S, R \rangle, \mathcal{R})$ is a UC-puzzle in the $(\mathcal{F}_1, \dots, \mathcal{F}_n)$-hybrid model if the belief set Σ is UC-compatible. We now briefly explain the puzzle construction.

Consider the n setups in the system. Recall that if a setup is of type t_i, then there exists a UC-puzzle (in the hybrid model of that setup) of type t_i. Then, our new puzzle protocol $\langle S, R \rangle$ is simply a sequential composition of the n puzzle protocols obtained from the n setups. Defining the associated relation $\mathcal{R}$ (and hence the trapdoor) is more tricky. Let $\Sigma = \{T_1, \dots, T_k\}$, where Σ is the belief set. Recall that as per definition 2, our (final) protocol should be UC-secure whenever there exists $T_i \in \Sigma$ such that each setup in T_i holds (in this case, we say that set T_i is good). To this end, we define k trapdoors σ_i, one corresponding to each set T_i; here, the main requirement is that if a set T_i is good, then (a) the simulator can obtain the trapdoor σ_i, but (b) no adversary can obtain any of the k trapdoors. We make the following observations: (a) the simulator can obtain the trapdoor for the UC-puzzle corresponding to each setup present in a good

set (but the adversary cannot), (b) the simulator can obtain the trapdoor for a Type III UC-puzzle even if the corresponding setup does *not* hold (hence, these trapdoors are for "free"). In light of these observations, we define σ_i to contain the trapdoor for the UC-puzzle corresponding to each setup present in T_i; additionally, σ_i contains the trapdoor for each Type III UC-puzzle.

Now suppose that $\exists i \in [k]$ such that T_i is good. Further, (in the worst case) suppose that each $T_j \neq T_i$ is such that none of the setups in T_j holds (we will call such a set T_j to be bad). By definition, the simulator can obtain the trapdoor σ_i. Further, no adversary can obtain the trapdoor σ_i. In order to argue that no adversary can obtain any of the remaining $k-1$ trapdoors (corresponding to the bad sets), we make use of the fact that Σ is UC-compatible. That is, since Σ is UC-compatible, for each bad set T_j, at least one of the four conditions (c.f. Definition 3) must hold. The proof follows by a case analysis. Here, in addition to the earlier observations, we use the fact that no adversary can obtain a trapdoor for a Type II UC-puzzle (even if the corresponding setup does *not* hold). Due to lack of space, we defer the details to the full version.

5 Different Beliefs about the Setups

In this section, we consider the setting where the parties in the system have different and independent beliefs about the setups in the system. Note that in such a scenario, depending upon the *reality* of how all the setups are implemented (for e.g., a third party that publishes a CRS may or may not be honest), the beliefs of some parties about the setups may turn out to be true, while that of other parties may turn out to be false. Then let us consider what would be an appropriate security definition for secure computation protocols in such a setting. Ignoring for a moment whether the definition is actually realizable, note that an acceptable security definition must provide standard security guarantees for at least those parties whose beliefs about the setups turn out to be true. But what of the parties whose belief about the setups turns out to be false? Note that a party would not expect the protocol to be secure if its belief about the setups turns out to be false. In light of this observation, below we consider a security definition that provides security for only those honest parties whose belief about the setups turns out to be true; any other party is considered to be adversarial, and therefore, no security is provided for such a party. As we show later, our definition is actually realizable when the beliefs of the parties about the setups satisfy a specific property (see below for more details). We now give more details.

Let $P_1, \ldots, P_m$ denote the parties in the system. We will use the notion of a $\mathsf{belief\ set}$ (as defined in Section 4) to represent the independent belief of each party about the setups in the system. Specifically, let $\Sigma_i = \{T_{i,1}, \ldots, T_{i,k_i}\}$ denote the belief set of P_i.

$(\Sigma_1, \ldots, \Sigma_m)$-secure protocols. As mentioned above, we would like to construct protocols that realize UC security with the (natural) condition that security is provided only for those (honest) parties whose beliefs about the setups turn out to be true. To formally capture the fact that no security provided for a specific set of

parties, we consider a minor modification in the standard model of UC security [4]. Specifically, let $U = \{P_1, \ldots, P_m\}$ denote the set of parties in the system. Let $H \subseteq U$ be a set of parties for whom we wish to provide standard security guarantees. We stress that H is not an a-priori fixed set of parties. Specifically, in our setting, H is determined once the adversary decides which setups in the system it wishes to corrupt. Then, in the modified UC framework, at the beginning of the protocol, the adversary $\mathcal{A}$ is required to corrupt each party $P_i \in U \setminus H$. The adversary can then further corrupt parties in H depending upon the corruption model (static or adaptive). We wish to provide security for the *honest* parties in H. We are now ready to define $(\Sigma_1, \ldots, \Sigma_m)$-*secure protocols.*

Definition 4 ($(\Sigma_1, \ldots, \Sigma_m)$-secure protocols). *Let there be n setups in the system denoted by $(\mathcal{G}_1, m\mathcal{G}_1)$, ..., $(\mathcal{G}_n, m\mathcal{G}_n)$. Let $\mathcal{F}_1, \ldots, \mathcal{F}_n$ be n functionalities, where $\forall i, \mathcal{F}_i$ is either $\mathcal{G}_i$ or $m\mathcal{G}_i$. Let $U = \{P_1, \ldots, P_m\}$ denote the set of parties in the system. For every i, let $\Sigma_i = \{T_{i,1}, \ldots, T_{i,k_i}\}$ be a belief set over the n setups that represents the independent belief of P_i. We say that a protocol Π $(\Sigma_1, \ldots, \Sigma_m)$-securely realizes a functionality f in the $(\mathcal{F}_1, \ldots, \mathcal{F}_n)$-hybrid model if $\forall H \subseteq \{P_1, \ldots, P_m\}$, Π UC-securely realizes f in the $(\mathcal{F}_1, \ldots, \mathcal{F}_n)$-hybrid model against all adversaries that initially corrupt all parties in $U \setminus H$, when $\forall P_i \in H$, $\exists j \in [k_i]$ such that each setup in $T_{i,j} \in \Sigma_i$ holds (i.e., $\mathcal{F}_\ell = \mathcal{G}_\ell$ for each setup $(\mathcal{G}_\ell, m\mathcal{G}_\ell) \in T_{i,j}$).*

UC-compatibility for Collection of Belief Sets. The notion of UC-compatibility (as defined in Section 4) is central to our results. Here, we extend this notion to a *collection* of belief sets.

Definition 5 (UC-compatible collection of belief sets). *A collection of belief sets $\Sigma_1, \ldots, \Sigma_m$ where $\Sigma_i = \{T_{i,1}, \ldots, T_{i,k_i}\}$ and $T_{i,\ell} \neq \emptyset$ $\forall i \in [m], \ell \in [k_i]$, is said to be UC-compatible if $\forall i, j \in [m]$ and $\forall \ell \in [k_i], \hat{\ell} \in [k_j]$, at least one of the following conditions hold:*

- $T_{i,\ell} \cap T_{j,\hat{\ell}} \neq \emptyset$.
- *Both $T_{i,\ell}$ and $T_{j,\hat{\ell}}$ contain at least one (not necessarily the same) Type II setup.*
- *Both $T_{i,\ell}$ and $T_{j,\hat{\ell}}$ contain at least one (not necessarily the same) Type III setup.*
- *Either $T_{i,\ell}$ or $T_{j,\hat{\ell}}$ contains at least one Type II setup as well as at least one Type III setup.*

Remark. Note that it is possible that given two belief sets Σ_1, Σ_2, neither of them is UC-compatible but the collection $\{\Sigma_1, \Sigma_2\}$ is UC-compatible.

We are now ready to state our main result for the different beliefs case.

Theorem 2 (Main result for different beliefs case). *Let there be n setups in the system denoted by $(\mathcal{G}_1, m\mathcal{G}_1), \ldots, (\mathcal{G}_n, m\mathcal{G}_n)$. Let $\mathcal{F}_1, \ldots, \mathcal{F}_n$ be n ideal functionalities, where $\forall i, \mathcal{F}_i$ is either $\mathcal{G}_i$ or $m\mathcal{G}_i$. Let $P_1, \ldots, P_m$ be m parties. For every i, let Σ_i be a belief set over the n setups that represents the independent*

belief of P_i. If the collection of belief sets $\Sigma_1, \ldots, \Sigma_m$ is UC-compatible, then for every well-formed functionality f, there exists a non-trivial protocol Π that $(\Sigma_1, \ldots, \Sigma_m)$-securely realizes f in the $(\mathcal{F}_1, \ldots, \mathcal{F}_n)$-hybrid model.

Proof (Idea). As noted earlier in Section 3, the task of constructing UC-secure protocols for any well-formed functionality reduces to the task of constructing an ssZK protocol. Intuitively, this is because given a functionality f, we can start with a semi-honest secure computation protocol Π for f, and then "compile" Π with an ssZK protocol to obtain a UC-secure protocol against active adversaries. Furthermore, following the result of [16], given any setup assumption (such as CRS), the above task is further reduced to the task of constructing a UC-puzzle in the hybrid model of the corresponding setup.

Now recall that in light of the above, we proved our positive result in the common belief case by simply constructing a UC-puzzle if the belief set (that represents the common belief of the parties) is UC-compatible. In the different beliefs case, however, instead of constructing a single UC-puzzle, we will construct a *family* of UC-puzzles if the collection of belief sets $\Sigma_1, \ldots, \Sigma_m$ (where Σ_i represents the belief of party P_i) is UC-compatible. Specifically, for each pair of parties P_i and P_j, we will construct two different UC-puzzles, (a) one where P_i (resp., P_j) acts as the sender (resp., receiver) and (b) the other where the roles of P_i and P_j are reversed. Then, given such a family of UC-puzzles, we can construct a family of ssZK protocols where the protocols in the family are concurrent simulation-sound with respect to each other. Specifically, for each pair of parties P_i and P_j, we can construct two different ssZK protocols, (a) one where P_i (resp., P_j) acts as the prover (resp., verifier), and (b) the other, where the roles of P_i and P_j are reversed. Finally, in order to construct a UC-secure protocol for any well-formed functionality f, we can start with a semi-honest protocol Π for f, and then "compile" Π with the above family of ssZK protocols in the following manner. Whenever a party P_i sends a protocol message to P_j, it proves that it has "behaved honestly so far in the protocol" by running an execution of the "appropriate" ssZK protocol (i.e., where P_i and P_j play the roles of the prover and verifier respectively) from the above family.

Due to lack of space, the details of the proof are deferred to the full version.

References

1. Barak, B., Canetti, R., Nielsen, J., Pass, R.: Universally composable protocols with relaxed set-up assumptions. In: FOCS (2004)
2. Barak, B., Prabhakaran, M., Sahai, A.: Concurrent non-malleable zero knowledge. In: FOCS (2006)
3. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: STOC (1988)
4. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: FOCS (2001)
5. Canetti, R., Dodis, Y., Pass, R., Walfish, S.: Universally composable security with global setup. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 61–85. Springer, Heidelberg (2007)

6. Canetti, R., Fischlin, M.: Universally composable commitments. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 19–40. Springer, Heidelberg (2001)
7. Canetti, R., Kushilevitz, E., Lindell, Y.: On the limitations of universally composable two-party computation without set-up assumptions. J. Cryptology 19 (2006)
8. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: STOC (2002)
9. Canetti, R., Pass, R., Shelat, A.: Cryptography from sunspots: How to use an imperfect reference string. In: FOCS (2007)
10. Chandran, N., Goyal, V., Sahai, A.: New constructions for UC secure computation using tamper-proof hardware. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 545–562. Springer, Heidelberg (2008)
11. Damgård, I., Nielsen, J.B., Wichs, D.: Isolated proofs of knowledge and isolated zero knowledge. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 509–526. Springer, Heidelberg (2008)
12. Goyal, V., Katz, J.: Universally composable multi-party computation with an unreliable common reference string. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 142–154. Springer, Heidelberg (2008)
13. Groth, J., Ostrovsky, R.: Cryptography in the multi-string model. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 323–341. Springer, Heidelberg (2007)
14. Katz, J.: Universally composable multi-party computation using tamper-proof hardware. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 115–128. Springer, Heidelberg (2007)
15. Kushilevitz, E., Lindell, Y., Rabin, T.: Information-theoretically secure protocols and security under composition. In: STOC (2006)
16. Lin, H., Pass, R., Venkitasubramaniam, M.: A unified framework for concurrent security: universal composability from stand-alone non-malleability. In: STOC (2009)
17. Lindell, Y.: Bounded-concurrent secure two-party computation without setup assumptions. In: STOC (2003)
18. Lindell, Y.: Lower bounds for concurrent self composition. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 203–222. Springer, Heidelberg (2004)
19. Moran, T., Segev, G.: David and goliath commitments: UC computation for asymmetric parties using tamper-proof hardware. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 527–544. Springer, Heidelberg (2008)
20. Pass, R.: Bounded-concurrent secure multi-party computation with a dishonest majority. In: STOC (2004)
21. Pass, R., Rosen, A.: Bounded-concurrent secure two-party computation in a constant number of rounds. In: FOCS (2003)

A Setups with Multiple Types

In the preceding text, for simplicity of exposition, we first restricted ourselves to the setting where each setup has a unique type. In this section, we discuss how our results can be extended to incorporate setups with multiple Types. Due to lack of space, we only provide an informal treatment of the results here. We refer the reader to the full version for more details.

Extending our results of Section 4 and 5. For simplicity of exposition, we will only consider the case where a setup is of all three types – Type I, Type II and Type III (other cases can be handled in a similar manner). The main

idea that allows to handle setups with multiple types is that we can think of a setup $(\mathcal{G}, m\mathcal{G})$ as three separate setups of unique type. If $(\mathcal{G}, m\mathcal{G})$ "holds", then each of the three setups must "hold," while if it is "corrupt", then each of the three setups are "corrupt." Then, roughly speaking, simply replacing $(\mathcal{G}, m\mathcal{G})$ with these three setups of unique types allows us to directly use the results of Section 4 and Section 5.

More specifically, recall that a belief set (as defined in Section 4) is expressed in terms of setups with unique types. If, instead, a belief set has a setup with multiple types, then we can modify the belief set and express it in terms of setups with unique types by following the above trick. Once we reduce all the setups in the system into setups of unique types and the belief set of parties are expressed in terms of these setups, then we can directly apply Theorem 1 to obtain a possibility result in the common belief case. In case parties have different belief sets as in Section 5, then we will need to express belief sets of all parties in terms of setups of unique types. Then we can directly apply Theorem 2 to obtain a possibility result for the distinct belief case.

B Setups with Multiple Failure Modes

In the preceding text, for simplicity of exposition, we first restricted ourselves to the setting where a setup has only a single failure mode. We now briefly discuss how to handle setups with multiple failure modes. Due to lack of space, our discussion will be informal and brief. We refer the reader to the full version for details.

Example. We first discuss a motivating example to highlight the importance of handling setups with multiple failure modes. Consider a system with three instances of the key registration setup. Recall that a key registration setup is based on the following two assumptions: (1) the secret keys of honest parties are "safe", and (2) the public keys of corrupted parties are "well-formed." Then, the parties in the system may have the following common belief: either

- the first key registration functionality is "honest", but in case it fails, then either (1) or (2) still holds (i.e., it never fails completely), or
- the second and third key registration functionalities are "honest", but in case the second functionality fails, then (1) still holds, while if the third functionality fails, then (2) still holds.

We stress that the above example is very "natural", and that in a real-world scenario, the parties may be willing to put trust into the "extent" of corruption of a setup. (In some scenarios, this may be due to some physical constraints imposed upon an adversary because of how the setup is done.) It is interesting to note that UC is indeed possible in the above example. We now briefly explain how to extend our model and our earlier results to accommodate setups with multiple failure modes.

Extending our model to accommodate multiple failure modes. Consider a setup modeled by an ideal functionality $\mathcal{G}$ with failure modes $m\mathcal{G}_1, m\mathcal{G}_2, \ldots,$

$m\mathcal{G}_\ell$. For lack of a better terminology, we will refer to each pair $(\mathcal{G}, m\mathcal{G}_i)$ (that was originally referred to as a setup) as a *setup mode*. Then, we can define Types for a setup mode in the same manner as we defined Types as in Definition 1. A setup mode $(\mathcal{G}, m\mathcal{G})$ is said to be of Type X if it is possible to construct a UC-puzzle of Type X in $(\mathcal{G}, m\mathcal{G})$ hybrid model. Note that the above definition allows a setup mode to have multiple Types. For simplicity of exposition, in this subsection, we will restrict ourselves to the case where a setup mode has only a single Type. We stress that this restriction can be easily removed by using techniques from Appendix A.

Extending our results of Section 4 and Section 5. Due you lack of space, we informally explain how our results in the common belief case can be extended. We refer the reader to the full version of the paper for details on the different beliefs case.

We note that the key issue that arises because of multiplicity of failure modes is in the definition of UC-compatible belief set (c.f. Definition 3). We start by giving a more general definition of UC-compatible belief set (see below) which takes into account the multiplicity of failure modes for setups. We then briefly argue that given this more general definition of UC-compatible belief set, Theorem 1 is still applicable.

Definition 6 (Generalization of Definition 3). *A belief set* $\Sigma = \{T_1, T_2 \dots T_k\}$*, where* $\forall i \in [k]$*,* $T_i \neq \emptyset$*, is said to be UC-compatible if* $\forall i, j \in [k]$*, at least one of the following conditions hold:*

- *There exists a setup* $\mathcal{G}$ *such that both* T_i *and* T_j *contain a setup mode* $(\mathcal{G}, \star)$*.*
- *Both* T_i *and* T_j *contain at least one (not necessarily the same) Type II setup mode.*
- *Both* T_i *and* T_j *contain at least one (not necessarily the same) Type III setup mode.*
- *Either* T_i *or* T_j *contains at least one Type II setup mode as well as at least one Type III setup mode.*

Definition 6 differs from Definition 3 in two ways: (a) The first condition of Definition 6 ignores the failure mode of the setups when taking intersection. However, we note that this is not really a fundamental difference because the failure modes of the setups were irrelevant in the first condition of Definition 3 as well. This is because we earlier restricted setups to exhibit only a single failure mode. Intuitively, this condition captures the case when the sets T_i and T_j share a setup and that setup *holds*. Hence, the failure mode is of no consequences. (b) The remaining three conditions in Definition 6 differ from the corresponding conditions in Definition 3 in the usage of setup modes instead of setups. We note, however, that these conditions in Definition 3 (resp., Definition 6) rely only on the types of the setup (resp., setup mode), and disregard the actual setups themselves. Hence, intuitively, the fact the same setup is leading to different types is of no consequences. Due to the above reasons, the proof of Theorem 1 easily extends to the general case with multiple failure modes.