Distributing the Setup in Universally Composable Multi-Party Computation

Jonathan Katz[*]
Dept. of Computer Science,
University of Maryland.
jkatz@cs.umd.edu

Aggelos Kiayias[†]
Dept. of Informatics and
Telecommunications,
University of Athens
aggelos@di.uoa.gr

Hong-Sheng Zhou[‡]
Dept. of Computer Science,
Virginia Commonwealth
University
hszhou@vcu.edu

Vassilis Zikas[§]
Dept. of Computer Science,
UCLA
vzikas@cs.ucla.edu

ABSTRACT

Universally composable (UC) protocols retain their security properties even when run concurrently alongside arbitrary other protocols. Unfortunately, it is known that UC multiparty computation (for general functionalities, and without assuming honest majority) is impossible without some form of setup. To circumvent this impossibility, various types of setup assumptions have been proposed.

With only a few notable exceptions, past work has viewed these setup assumptions as being implemented by some ideal, incorruptible entity. Any such entity is thus a *single point of failure*, and security fails catastrophically in case the setup entity is subverted by an adversary.

We propose here a clean, general, and generic approach for distributing trust among m arbitrary setups, by modeling potential corruption of setups within the UC framework, where such corruption might be fail-stop, passive, or arbitrary and is in addition to possible corruption of the parties themselves. We show several feasibility and impossibility results in this model, for different specifications of the corruptible sets. For example, we show that given m complete setups, up to t of which might be actively corrupted in an adaptive manner, general multiparty computation with no honest majority is possible if and only if $t < m/2$.

1. INTRODUCTION

Secure multi-party computation (MPC) is a quintessential cryptographic problem. The strongest and most desirable level of security that can be attained by such a protocol is universal composability (UC) [2]. Unfortunately, results by Canetti and Fischlin [5] and Canetti, Kushilevitz, and Lindell [6] show that unless a majority of the parties are honest, most "interesting" functionalities cannot be realized in the UC framework. These impossibility results can be circumvented if one is willing to assume a *trusted setup*: Canetti and Fischlin [5] showed that the existence of a (trusted) common reference string (CRS) allows for circumventing their impossibility result. Subsequently, Canetti et al. [7] showed that a CRS is *complete* in that it can be used for UC computation of arbitrary functionalities, for any number of corrupted parties. Similarly, oblivious transfer (OT) [24], a two-party primitive for sending one out of two inputs without learning which one was actually delivered, was proven to be complete for computing arbitrary functionalities with information-theoretic (aka unconditional or statistical) security [21]. Since then, researchers have suggested a variety of setups, e.g., [1, 18, 8, 20], and showed them to be complete; we refer to [3] for a (now slightly outdated) survey.

With only a few exceptions [15, 14, 12] (discussed below), prior work in this line of research has viewed setup assumptions as being implemented by some ideal, incorruptible entity. In reality, however, most setups would have to be implemented by some mechanism that could be subverted, or by some party that could be compromised. (As an example, NIST is planning to run a "random beacon" that could function as a CRS—cf. `https://beacon.nist.gov/home` and `http://www.nist.gov/itl/csd/ct/nist_beacon.cfm`; their implementation could be hacked, or contain unexpected design flaws.) To cope with such situations, recent works [15, 14, 12] have suggested distributing trust among more than one such (setup-)mechanisms, so that security is not compromised even when some of them fail.

[*] Work supported in part by NSF awards #1111599 and #1223623.

[†] Work supported by ERC grant CODAMODA.

[‡] Work supported in part by an NSF CI postdoctoral fellowship, and mainly done while at the University of Maryland.

[§] Work supported in part by the Swiss National Science Foundation (SNF), project no. 200020-132794, and mainly done while at the University of Maryland.

PODC'14, July 15–18, 2014, Paris, France.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-2944-6/14/07 ...$15.00.
http://dx.doi.org/10.1145/2611462.2611480.

1.1 Our Contribution

We propose here a new approach for modeling potential failure/corruption of setups within the UC framework that is more general and, arguably, more natural than prior attempts. Informally, for an arbitrary setup functionality $\mathcal{F}$ we consider three ways $\mathcal{F}$ might be compromised. In the *fail-stop* case, an adversary can cause $\mathcal{F}$ to halt at any point. *Passive* corruption allows the adversary to read the entire internal state of $\mathcal{F}$, but not change its behavior. Finally, in the *active* case the adversary gains complete control over $\mathcal{F}$ and can cause it to behave arbitrarily. This is in addition to any possible corruption of the parties running the protocol. Clearly, access to a single setup that can be actively corrupted is no different than having no setup at all. But our framework allows us to analyze the scenario where the parties have access to *multiple* setups, some subset of which might be corrupted.

In most of our results, we allow the adversary to *adaptively* choose the parties and the setups to corrupt. Note that, similarly to party-corruption, adaptive corruption of setups is an important consideration: Consider, for example, the following mechanism used by two parties having access to several setups (some of which might be corrupted): in order to increase their probability of using uncorrupted setups, the parties start any protocol by agreeing on a random subset of setups to be used (e.g., by executing a coin-tossing protocol to determine this subset). Clearly adaptively choosing the corrupted setups (after the coins have been flipped and the decision of which setups to be used is taken) gives the adversary advantage in such a protocol over the adversary with static setup corruption.

As our main technical contribution, we study feasibility of general secure computation with no honest majority within the framework discussed above. We show, for example, that given m *arbitrary* complete setups, any t of which might be *actively* corrupted, secure computation of arbitrary (well-formed) functionalities is possible *if and only if* $t < m/2$. We also prove that this bound is tight even for *passive* corruption of setups. We further extend our characterization to the setting of mixed (i.e., active and passive) corruption: given m arbitrary complete setups, where any t_a might be *actively* corrupted and, simultaneously, any t_p might be passively corrupted, secure computation of arbitrary (well-formed) functionalities is possible *if and only if* $t_a + t_p < m/2$. We further generalize the above threshold results to the case of arbitrary "corruption structures" that enumerate the allowable sets of corruptible setups; there, we show that secure computation of arbitrary functionalities is possible if and only if no two corruptible sets cover the entire set of available setups.

Additionally, we show that for many setups considered in the literature, one can do better than the above generic bounds. (The negative results stated hold for arbitrary complete setups; for specific setups it may be possible to do better.) In particular, we introduce the class of *passively immune* setups (which includes a natural variant of the CRS functionality) we show that given m complete setups in this class, any t_a of which might be actively corrupted, secure computation of arbitrary functionalities is possible *if and only if* $t_a < m/2$ *even if all* m *setups can simultaneously be* passively *corrupted.*

All the above mentioned results hold for an adversary who, in addition to corrupting functionalities, *is allowed to actively corrupt arbitrary many parties.* In a slightly different direction, we give a "best-of-both worlds" result for the setting where there is either a corrupted majority of setups, or a corrupted majority of parties. Due to space limitation, some of the details of our results are deferred to the full version of the current extended abstract.

1.2 Prior Work on Corruptible Setups

Corruptible UC setups were first considered by Groth and Ostrovsky [15] for the special case of a Common Reference String (CRS). In particular, they studied the "multi-CRS" model where there are m common reference strings and the adversary is allowed to replace any t of them with strings of his own choice. Building on their work, Goyal and Katz [14] looked at the case where there is a *single* CRS, and either the CRS is compromised (as in [15]) but a majority of the parties are honest, or the CRS is not compromised but a majority of the parties is no longer honest. Finally, in TCC 2011 Garg et al. [12] initiated the study of feasibility of UC computation with *potentially different* setups that can "fail" in certain ways. Our work unifies, abstracts, and extends the above results, as we now explain.

Comparison to the work of Groth and Ostrovsky [15].

In contrast to [15], our corruption model is both more general and more broadly applicable. We consider active, passive, and fail-stop[1] corruption of setup functionalities, and in the active case we allow the adversary to *arbitrarily control* the setup (e.g., an actively corrupted CRS can send different strings of the adversary's choice to different parties, something not considered in [15]). Moreover, we consider corruption of general setup functionalities, not just a CRS, and give results for general corruption structures.

On a technical level, Groth and Ostrovsky [15, Theorem 4] prove impossibility of implementing non-interactive zero-knowledge for languages that are not in $\mathcal{P}/\mathsf{poly}$ when more than half the CRSs are faulty. In contrast, we show unconditionally that *any* set of arbitrary (even different) setup functionalities becomes *trivial* (i.e., implementable from secure channels without further assumptions) as soon as half of them are faulty.

Comparison to the work of Garg et al. [12].

Lin et al. [22] introduced the notion of a UC puzzle, and showed that UC puzzles are complete for secure computation. (We sketch the definition of UC puzzles in Appendix A.) In the work of Garg et al., corruption of setups is not modeled "directly"; rather, three types of "faulty" UC puzzles are introduced, and corruption of a setup $\mathcal{F}$ is defined by what type of faulty UC puzzle can be realized using $\mathcal{F}$. A bit more specifically, given a setup $\mathcal{F}$ that can be used to construct a UC puzzle, and the assumption that an adversary can corrupt $\mathcal{F}$ to obtain the faulty setup $\mathcal{F}'$, the "type" of this corruption model is defined to be exactly the "type" of faulty UC puzzle that results from using $\mathcal{F}'$.

[1] The non-triviality of dealing with fail-stop corruption stems from the fact that we aim for secure computation with *unanimous* abort [13]. Thus, it is not sufficient for a single party to disqualify a setup when it observes a failure; rather, there needs to be consensus that a failure occurred. And running a consensus protocol is not an option as we allow arbitrary many corrupted parties.

From a conceptual point of view we find our modeling more appealing, as it speaks *directly* of the different ways (fail-stop, passive, active) in which arbitrary setups can be compromised, rather than indirectly in terms of what sort of puzzle can be constructed. Moreover, the framework of Garg et al. allows the faulty version $\mathcal{F}'$ of a setup $\mathcal{F}$ to be *arbitrary*, even having no relation to $\mathcal{F}$. (E.g., $\mathcal{F}$ could be a CRS, while $\mathcal{F}'$ implements oblivious transfer. Our modeling, instead, uses well-defined "wrappers" which ensure a particular relationship between $\mathcal{F}$ and $\mathcal{F}'$. Although we only deal with fail-stop, passive, and active faults in this paper, our framework could easily be extended to deal with other faults, e.g., dropped messages.

Our choice of modeling has technical consequences as well, which are reflected both on our results and the corresponding proofs (all our proofs here follow a different approach than [12]). First, note that the framework of Garg et al. *only* applies to setups $\mathcal{F}$ that can be used to construct UC puzzles in the first place. In contrast, our framework applies to arbitrary, even incomplete, setups. Even restricting attention to complete setups (which is the more interesting case), it is not known whether all complete setups can be used to construct UC puzzles: Indeed, even for a UC complete setup $\mathcal{R}$, the standard approach of replacing, in an $\mathcal{F}$-hybrid UC puzzle, calls to $\mathcal{F}$ by an $\mathcal{R}$-hybrid protocol computationally implementing $\mathcal{F}$ does not yield a $\mathcal{R}$-hybrid UC puzzle as it does not satisfy the *statistical* simulatability property [22]. Second, observe that applying the results of Garg et al. to some specific $\mathcal{F}$ (that is known to yield a UC puzzle) and its faulty version $\mathcal{F}'$, would require the protocol designer to know (or somehow figure out) what type of faulty UC puzzle can be constructed from $\mathcal{F}'$. It is not clear, in general, how (or even if) this can be done. Finally, we note that although Type 1 setups (as defined in [12]) can be shown to correspond to the active corruptions we consider here (at least for setups that can be handled by [12]), such a correspondence does not seem to hold for the other corruption types considered in our work.

With regard to differences in the results (as opposed to differences in the models), we first note that the positive results from [12] all rely on the construction of general multi-party computation from UC puzzles from [22]. As a consequence, (1) their protocols can *only* be *computationally* secure, even when the assumed setups are complete for information-theoretic secure computation; (2) their work only handled static adversaries.[2] In contrast, our possibility results rely on constructions of MPC from oblivious transfer (OT), yielding *information-theoretic* constructions secure against *adaptive adversaries.*

Finally, we mention that *proving tightness of the positive results* was left as the main open question in [12]. Although our results do not directly answer this question in their model (since we use a different model), we are able to prove tightness of all the positive results shown in this work.

Combiners for coping with malicious setups.

In other work similar in spirit to our own, researchers have studied the notion of *combiners* [16, 23, 26]. The goal of that work is to construct a single primitive securely realizing some task (e.g., oblivious transfer) from a collection of m such primitives, some t of which may fail. Our work is more general in several ways, in particular because we work in the UC framework, and do not require that each of our m setup functionalities realizes the same primitive. Rather, our impossibility results apply to arbitrary, even different setups. Additionally, we specify different ways in which setups might fail and provide characterizations for each of them individually as well as their combinations. In fact, a corollary of our impossibility result for passive corruption of setups (Lemma 6) gives a simpler proof of impossibility for third-party black-box OT combiners when half (or more) of the OT candidates are faulty [16, 23]. On the other hand, at the technical level, our main positive results are obtained by reduction to OT-combiners.

2. THE MODEL

We lay down the specifics of our model and fix some notation and conventions. We assume the reader has some familiarity with the Universal Composition (UC) framework [2]. We use the following notation: for $m \in \mathbb{Z}$ we denote by $[m]$ the set $[m] = \{1, \ldots, m\}$. We consider both the two-party and the multi-party case, where the parties can communicate over a complete network of bilateral authenticated channels. We denote by $\mathcal{P} = \{p_1, \ldots, p_n\}$ the player set ($\mathcal{P} = \{p_1, p_2\}$ for the two-party case). The adversary, denoted by $\mathcal{A}$, corrupts parties adaptively, i.e., might corrupt parties during the execution of the protocol. We consider standard (i.e., active) party corruption as defined in [2].

As usually in the UC literature we model setups as UC functionalities. The *setup-functionalities* $\mathcal{F}$ considered in this work have the following properties: **(1)** Whenever some input is received from some honest party, the functionality $\mathcal{F}$ notifies the adversary *(input notification)*, and **(2)** all outputs of the functionality are issued in a delayed manner as specified in [2], i.e., whenever $\mathcal{F}$ has an output to be sent to some party p_i, it notifies the adversary about this fact (the notification includes the PID of p_i) and delivers the input only when the adversary permits it *(delayed output)*. Furthermore, we only deal with setup-functionalities that are *efficiently computable*, where we assume that the inputs and outputs of the setup-functionalities are strings of a fixed length. Whenever the term "setup-functionality" is used in this work it refers to a functionality that satisfies the above properties.[3]

Consistently with the UC framework, assuming a (setup) functionality $\mathcal{F}$ implies assuming that every $\mathcal{F}$-hybrid protocol can instantiate its own (independent) instance(s) of $\mathcal{F}$.[4] Hence, assuming that the setup $\mathcal{F}$ fails (in our terminology, becomes corrupted) implies that its instances fail across all protocols using $\mathcal{F}$ as a hybrid. As pointed out in [2, 9], a side effect of this modeling choice is that in order to prove security of multiple protocols using the same (stateful) setup, e.g., the same public key infrastructure (PKI), one needs to

[2] Recent work of Dachman-Soled et al. [11] shows how to construct adaptively secure MPC from UC puzzles. Plugging this protocol into the feasibility result of [12] strengthens their result to hold for adaptive corruption of parties (but still static failure of setups). But the construction from [11] relies on even stronger (non-standard) complexity assumptions than those used in [12].

[3] At times we might refer to setup-functionalities as *setups*.

[4] For example, when one speaks of UC completeness of Oblivious Transfer (OT) functionality, it is implicitly assumed that multiple independent instances of the OT functionality can be invoked with different sender and receivers.

use the *joint state composition theorem* (JUC) [9]. An alternative approach would be to model setups directly in the *UC with global setups* (GUC) model of Canetti et al [4]. Our modeling of corrupted functionalities can be easily extended to capture corrupted (global) setups in GUC. However, our proofs do not go through in that model; it is an interesting open question to prove bounds for corrupted setups in the GUC model.

2.1 Modeling Corruption of Functionalities

To capture the fact that a setup might be, at least partially, controlled by the adversary we introduce the notion of corruptible functionalities. *Corruptible functionalities* are modeled as being enclosed into a *corruption wrapper* which behaves as follows: any message sent to/from the functionality is forwarded by the wrapper to its recipient (the wrapper also records all these messages and can access the random tape of the wrapped functionality). In addition to the standard communication with the functionality, the wrapper might accept from the adversary $\mathcal{A}$ messages of the form $(\texttt{fcorrupt}, \textit{type})$—where *type* specifies the way in which $\mathcal{A}$ wishes to corrupt the underlying functionality—and modify its behavior accordingly. Consistently with the multi-party computation (MPC) literature, in this work we focus on the following three corruption types

Active: Upon receiving a message $(\texttt{fcorrupt}, \texttt{active})$ from $\mathcal{A}$, the corruption wrapper sends $\mathcal{A}$ all the messages it has seen so far as well as the internal randomness of the wrapped functionality; from then on, the wrapper behaves as the "dummy" functionality which forwards every message it receives from any party to the adversary $\mathcal{A}$, and when instructed by the adversary to send (as output) some message y to some party, it does so.

Passive: Upon receiving a message $(\texttt{fcorrupt}, \texttt{passive})$ from $\mathcal{A}$, the wrapper sends $\mathcal{A}$ all the messages it has seen so far as well as the internal randomness of the wrapped functionality.[5]

Fail: Upon receiving a message $(\texttt{fcorrupt}, \texttt{fail})$ from $\mathcal{A}$, the wrapper enters a *crashed* stated, i.e., answers all messages sent to it from that point on with a special $(\texttt{crashed})$ message; we say then that the adversary has *forced* $\mathcal{F}$ *to to crash.*

A wrapped functionality which accepts active or passive corruption requests will be called *actively corruptible* or *passively corruptible*, respectively. A setup-functionality which does not accept any corruption requests (i.e., behaves as unwrapped) is called *incorruptible.* In order to model mixed-type corruption of functionalities, we also consider wrapped functionalities that accept corruption messages of more than one type. Unless explicitly stated otherwise, we assume that *the adversary might corrupt the setup-functionalities in an adaptive manner*, i.e, might corrupt (additional) setups during the execution of a protocol.

Consistent with the terminology used in the literature for party corruption we refer to a setup-functionality $\mathcal{F}$ which has been actively corrupted, i.e., the adversary has sent $\mathcal{F}$ $(\texttt{fcorrupt}, \texttt{active})$, as *malicious.* Similarly, a passively corrupted functionality is called *semi-honest.* $\mathcal{F}$ is called *honest* or *uncorrupted* if the adversary has not sent $\mathcal{F}$ any $(\texttt{fcorrupt}, \textit{type})$ message.

Functionality Sets and Corruption Patterns.

Let $\mathcal{M} = \{\mathcal{F}_1, \ldots, \mathcal{F}_m\}$ be a set of (possibly different) setup functionalities. In slight abuse of terminology we often refer to $\mathcal{M}$ as a setup-functionality, where calls to $\mathcal{M}$ are addressed to the appropriate functionality. An $\mathcal{M}$*-hybrid* protocol π is a protocol that can make hybrid calls to all the functionalities in $\mathcal{M}$. We say that the adversary is t*-restricted* if he might corrupt up to a certain threshold t of the functionalities. We also consider the more general setting, where the adversary might corrupt any set of functionalities from a specific *corruption structure* $\mathcal{K}$; we refer to the latter as a $\mathcal{K}$*-restricted* adversary.[6] More precisely, $\mathcal{K}$ is a *monotone* set of subsets of the functionality-set $\mathcal{M}$, i.e., $\mathcal{K} = \{A_1, \ldots, A_n\}$, where each $A_i \subseteq \mathcal{M}$. A $\mathcal{K}$-restricted adversary might corrupt all the functionalities in some class $A^\star \in \mathcal{K}$.

Corruption Ignoring Strengthening.

Throughout this extended abstract we assume functionalities that allow for active and passive corruption. However, at times we need to refer to the corresponding honest functionality. Therefore we introduce the following notation: For a given setup-functionality $\mathcal{F}$, we define the *corruption-ignoring strengthening* of $\mathcal{F}$, denoted as $\mathcal{F}^{\not\mathbf{C}}$, to be the "unwrapped" functionality that ignores all $(\texttt{fcorrupt}, \cdot)$ requests. At times, we need to refer to the strengthening of a functionality that ignores only a specific type of corruption-requests; we refer to the strengthening of $\mathcal{F}$ that ignores only $(\texttt{fcorrupt}, \texttt{passive})$ (resp. $(\texttt{fcorrupt}, \texttt{active})$) commands as the *passively ignoring* (resp. *actively ignoring*) strengthening of $\mathcal{F}$, and denote it as $\mathcal{F}^{\not\mathbf{P}}$ (resp. $\mathcal{F}^{\not\mathbf{A}}$). Note that (depending on $\mathcal{F}$), the corruption ignoring strengthening $\mathcal{F}^{\not\mathbf{C}}$ might not necessarily be a useful setup, e.g., one could consider a functionality $\mathcal{F}$ which, upon receiving from the adversary the message $(\texttt{becomeDummy})$, acts as $\mathcal{F}^{\text{dum}}$.

Assumptions, Completeness, and Sufficiency.

For a pair of functionalities $\mathcal{F}$ and $\mathcal{F}'$ and a computational assumptions α, we say that a functionality $\mathcal{F}$ is *(UC)* α*-sufficient for* $\mathcal{F}'$ if, assuming α, there exists an $\mathcal{F}$-hybrid protocol which UC-realizes $\mathcal{F}'$ (using authenticated communication). Furthermore, we say that a functionality $\mathcal{F}$ is *(UC)* α*-complete* if, assuming α, for every well-formed[7] functionality $\mathcal{F}'$ there exists an $\mathcal{F}$-hybrid protocol which UC-realizes $\mathcal{F}'$. Note that for the definition of UC completeness we make no restriction on the adversary's party-corruption capabilities, i.e., *the adversary might corrupt arbitrary many parties.* Some specific examples of assumptions α_ℓ used in this work are the existence of one-way permutation (OWP), denoted as $\alpha_{\texttt{OWP}}$, and the existence of an oblivious-transfer (OT) protocol secure for semi-honest adversaries, denoted as $\alpha_{\texttt{shOT}}$.

The definition of completeness naturally extends to functionality sets and/or assumption sets. In particular, for

[5] Note that the adversary might send the $(\texttt{fcorrupt}, \texttt{passive})$ message several times in a protocol execution.

[6] This corruption pattern corresponds to the so called *general adversary* [17] from the MPC literature.

[7] Roughly speaking, a functionality is well-formed if its code does not depend on the ID's of the corrupted parties; for a detailed description we refer to [7].

assumptions $\alpha_1, \ldots, \alpha_q$ we denote by $\{\alpha_1, \ldots, \alpha_q\}$ the assumption which holds if for every $\ell \in [q] : \alpha_\ell$ holds. We say that *a functionality-set* $\mathcal{M} = \{\mathcal{F}_1, \ldots, \mathcal{F}_m\}$ *is* $\{\alpha_1, \ldots, \alpha_q\}$*-complete* if, assuming $\{\alpha_1, \ldots, \alpha_q\}$ holds, for every well-formed functionality $\mathcal{F}'$ there exists an $\mathcal{M}$-hybrid protocol which UC-realizes $\mathcal{F}'$.

3. IMPOSSIBILITY FOR MALICIOUS MAJORITY

We start our analysis with our main impossibility result for the case of actively corruptible functionalities. Informally, we prove the following:

> Any collection of setups becomes trivial (i.e., implementable via secure channels— denoted by $\mathcal{F}_{\text{SEC}}$) if the adversary is allowed to actively corrupt more than half of them.

We state the result in Theorem 1 in its most general form, i.e., for an adversary characterized by a functionality-corruption structure $\mathcal{K}$. We use the following notation borrowed from [17]: we say that a structure $\mathcal{K}$ is a Q^2 structure if condition $Q^2(\mathcal{M}, \mathcal{K})$ holds, where

$$Q^2(\mathcal{M}, \mathcal{K}) \iff \forall A_i, A_j \in \mathcal{K} \ : \ A_i \cup A_j \neq \mathcal{M}$$

THEOREM 1 (STATIC PARTY-CORRUPTION). *For a 2-party setup-functionality set* $\mathcal{M} = \{\mathcal{F}_1, \ldots, \mathcal{F}_m\}$*, and a functionality-corruption structure* $\mathcal{K}$*, if* $Q^2(\mathcal{M}, \mathcal{K})$ *does not hold then there exists an* $\mathcal{F}_{\text{SEC}}$*-hybrid 2-party protocol which securely realizes (all the functionalities in)* $\mathcal{M}$*, in the presence of a* $\mathcal{K}$*-restricted adversary who might statically corrupt any (or even both) parties.*

PROOF. Let $A_1, A_2 \in \mathcal{K}$ be functionality sets such that $A_1 \cup A_2 = \mathcal{M}$ (such sets are guaranteed to exist by the assumption that $\mathcal{K}$ is not Q^2). Without loss of generality assume that $A_1 \cap A_2 = \emptyset$ (if this is not the case take $A_2 \setminus A_1$ instead of A_2).[8] The protocol Π for securely realizing $\mathcal{M}$ in the $\mathcal{F}_{\text{SEC}}$-hybrid world is as follows: The idea of the protocol is to have each p_i (for $i = 1, 2$) answer all queries that are for the functionalities in A_i. More precisely, each p_i internally simulates the code of every $\mathcal{F}_\ell \in A_i$ and stores the internal states of all (simulated) $\mathcal{F}_\ell$'s; whenever in the protocol some party sends a message to the other party, this is done by use of the secure channel $\mathcal{F}_{\text{SEC}}$. The detailed code of p_i is given in the following.

Protocol $\Pi(\{p_1, p_2\}, \mathcal{M}, A_1, A_2)$

Code for p_i:

- Upon receiving an input x for functionality $\mathcal{F}_\ell$, i.e., the command $(\texttt{input}, \ell, x)$, from the environment $\mathcal{Z}$:
  - if $\mathcal{F}_\ell \in A_i$ then p_i simulates the behavior of $\mathcal{F}_\ell$; if the simulated $\mathcal{F}_\ell$ would generate a message y for p_i then output $(\texttt{output}, \ell, y)$ to the environment, otherwise, if it would generate a message y for p_{3-i} then send $(\texttt{output}, \ell, y)$ to p_{3-i}.
  - else, i.e., if $\mathcal{F}_\ell \in A_{3-i}$, then p_i sends the message $(\texttt{input}, \ell, x)$ to p_{3-i}.
- Upon receiving a message $(\texttt{input}, \ell, x)$ from p_{3-i}, if $\mathcal{F}_\ell \notin A_i$ then ignore the message. Else, simulate the behavior of $\mathcal{F}_\ell$ on input x from p_{3-i}; if the simulated $\mathcal{F}_\ell$ would generate a message y for p_i then output $(\texttt{output}, \ell, y)$ to the environment $\mathcal{Z}$, otherwise, if it would generate a message y for p_{3-i} then send $(\texttt{output}, \ell, y)$ to p_{3-i}.
- Upon receiving a message $(\texttt{output}, \ell, y)$ from p_{3-i}, if $\mathcal{F}_\ell \in A_i$ then ignore the message, else output $(\texttt{output}, \ell, y)$ to the environment $\mathcal{Z}$.

We show that protocol Π UC securely realizes the functionality $\mathcal{M}$. The simulator for an adversary $\mathcal{A}$ works as follows: We consider the following four cases: (I) Both p_1 and p_2 are corrupted, (II) only p_1 is corrupted, (III) only p_2 is corrupted, and (IV) both p_1 and p_2 are honest.

In Case I: The simulator acts as a forwarder between the adversary and $\mathcal{Z}$ (and never invokes the functionality $\mathcal{M}$).

In Case II: The simulator corrupts all $\mathcal{F}_\ell \in A_1$. Any message which is received from the environment is forwarded to the adversary; any message which the adversary generates for the environment is outputted to $\mathcal{Z}$.

- Whenever $\mathcal{S}$ receives from $\mathcal{A}$ a message z to be sent to p_2, $\mathcal{S}$ simulates the view of $\mathcal{Z}$ in the secure channel, i.e., sends $|z|$ to $\mathcal{Z}$, and if z is of the form $(\texttt{input}, \ell, x)$ for some $\mathcal{F}_\ell \in A_2$, $\mathcal{S}$ inputs z to $\mathcal{F}_\ell$.
- Whenever p_2 sends input $(\texttt{input}, \ell, x)$ to some $\mathcal{F}_\ell \in A_1$, $\mathcal{S}$ (who, recall, has actively corrupted $\mathcal{F}_\ell$) receives this input, he simulates the view of $\mathcal{Z}$ corresponding to $(\texttt{input}, \ell, x)$ being sent through the secure channel, i.e., sends $|(\texttt{input}, \ell, x)|$ to $\mathcal{Z}$ and hands $(\texttt{input}, \ell, x)$ to the adversary $\mathcal{A}$.
- For every pending output $(\texttt{output}, \ell, y)$ (recall that the setup-functionalities $\mathcal{F}_\ell$ issue their output in a public delayed manner) for $\mathcal{F}_\ell \in A_2$, as long as $\mathcal{S}$ has finished simulating the protocol messages and the output should be delivered in the (simulated) protocol, $\mathcal{S}$ instructs $\mathcal{F}_\ell$ to deliver the message to the corresponding party.

In Case III: This case can be handled as in Case II, symmetrically.

In Case IV: If the adversary $\mathcal{A}$ (in the $\mathcal{F}_{\text{SEC}}$-hybrid world) corrupts no party, then the simulator simply needs to simulate the view of $\mathcal{A}$ from the secure channel $\mathcal{F}_{\text{SEC}}$. Note that the simulator can trivially do this as (1) all messages exchanged between the parties are of fixed length which allows the simulator to know the output of $\mathcal{F}_{\text{SEC}}$ whenever a message would be exchanged in the protocol, and (2) the functionality notifies the simulator for every input it receives and always asks for permission before delivering the outputs, which allows $\mathcal{S}$ to order the messages sent to the environment according to the order they would appear in the protocol.

It is straight-forward to verify that the above is a good simulator for Protocol Π. $\square$

We stress that Theorem 1 only claims *static* corruption of parties. In fact, counter to the standard intuition, we cannot obtain the same statement for adaptive party corruption as a corollary; we refer to Section 3.1 for a justification and for a complete handling of adaptive party-corruption. As a corollary of Theorem 1 we obtain the following impossibility result for t-restricted adversaries.

[8] Recall that by definition $\mathcal{K}$ is monotone, thus $A_2 \setminus A_1 \in \mathcal{K}$.

Corollary 2. *For a 2-party setup-functionality set $\mathcal{M} = \{\mathcal{F}_1, \ldots, \mathcal{F}_m\}$, there exists an $\mathcal{F}_{\text{SEC}}$-hybrid 2-party protocol which securely realizes (all the functionalities in) $\mathcal{M}$, in the presence of a $m/2$-restricted adversary who can (simultaneously) statically corrupt any (or even both) parties.*

Proof. Let $\mathcal{K}_t$ denote the structure including all subsets of $\mathcal{M}$ of cardinality at most t. When $t \geq m/2$, the the condition $Q^2(\mathcal{M}, \mathcal{K}_t)$ is violated. Thus the statement follows trivially from Theorem 1. □

Observe that the only impossibility result in the multi-setup UC model is due to Groth and Ostrovsky [15], who showed that if a language L has a non-interactive zero-knowledge (NIZK) proof system in the multi-string model, then either $L \in \mathsf{P}/\mathsf{poly}$ or at most half of the common reference strings might be adversarially chosen. Their proof uses a completely different approach than the one used here. Furthermore, our impossibility result is more general than the result from [15] in several aspects: In particular, for the case of a static adversary it extends to arbitrary setups. Moreover, as we show in Section 3.1, for the special case of the CRS (in fact for any adaptively well-formed setup [7]) our impossibility extends adaptive corruption of parties, thereby *strictly* generalizing [15]. Finally, our result proves that when a majority of setups is corrupted, then the multi-setup functionality in not just insufficient for NIZKs, but it is trivial, in the sense that it can be realized from secure channels.

Remark (A weaker corruption type [15] also covered by Theorem 1). In [15] the adversary is assumed to have less power on the CRS than our active corruption, i.e., $\mathcal{A}$ is merely allowed to change the value of the common string, but he cannot make the functionality distribute inconsistent (i.e., different) strings (see the discussion in Section 1.2). It is straight-forward to verify that our proof works, almost verbatim, even if we restrict the power of the simulator along these lines. The only difference is that in the protocol used in the proof, as soon as a value is output by some party the party does not output any different value in the future.

3.1 Adaptive Party Corruption

Although one is tempted to assume that Theorem 1 implies impossibility also for *adaptive* corruption of parties, this is not the case, as allowing the adversary to adaptively corrupt the parties makes simulation in the $\mathcal{M}$-hybrid world harder. In particular, the proof of Theorem 1 breaks down for an adversary who first corrupts one of the parties, say p_1, and later corrupts the other one p_2. Intuitively, the reason is that while emulating this adversary, the simulator needs, at the point when p_2 gets corrupted, to come up with internal states for the setups emulated by p_2 that are consistent with the outputs they generated so far. However, by that time the simulator has exhausted all his corruption resources in order to emulate the corruption of p_1 and is therefore stuck.

In this section we show that this limitation can be overcome if we assume that the setups are adaptively well-formed [7]. Roughly speaking, an adaptively well-formed setup hands the adversary its internal randomness as soon as every party gets corrupted (for a formal description we refer to [7]). We remark, that adaptive well-formedness is a natural assumption for setups and all natural (in particular all known) setups satisfy it or they can be trivially modified to satisfy it by adding an extra instruction to send their adversary their full internal randomness as soon as every party becomes corrupted). In particular, for the class of adaptively well-formed setups, the result of Theorem 1 holds even for an adversary who can corrupt parties in an adaptive manner. Note that, because the CRS is adaptively well-formed, Theorem 1 holds for the CRS even assuming adaptive corruption of parties. The following theorem extends Theorem 1 to an adversary who corrupts parties adaptively.

Theorem 3. *For a set $\mathcal{M} = \{\mathcal{F}_1, \ldots, \mathcal{F}_m\}$ of adaptively well-formed 2-party setup-functionalities,*[9] *and a corruption structure $\mathcal{K}$, if $Q^2(\mathcal{M}, \mathcal{K})$ does not hold then there exists an $\mathcal{F}_{\text{SEC}}$-hybrid 2-party protocol which securely realizes (all the functionalities in) $\mathcal{M}$, in the presence of a $\mathcal{K}$-restricted adversary who might adaptively corrupt any (or even both) parties.*

Proof (*Idea).* We can use the same protocol as in Theorem 1. Intuitively, the reason why this is simulatable is the following: as long as at most one party is corrupted, the simulator does not have a problem as he can choose which functionalities to corrupt depending on the adversary's choice of party corruption (recall that we assume that the corruption of functionalities is by default adaptive and the corruption wrapper keeps a record of past inputs and outputs). Hence a problem might occur only when the adversary corrupts both parties during the protocol execution. In that case, the adaptive well-formedness of the setups ensures that the simulator learns the randomness used by all (even the uncorrupted) setups and, as by corrupting both parties he learns all inputs/outputs, he can go on with the simulation. We refer to the full version for a formal proof. □

Similarly to Corollary 2, a threshold impossibility result (for $t \geq m/2$ corrupted setups) can be obtained also for the adaptive case as a corollary of Theorem 3.

4. ACTIVE CORRUPTION OF SETUPS

Having established our main impossibility result, we proceed to proving its tightness. Informally, the statement we prove is the following:

> For a set $\mathcal{M}$ of (possibly different) actively corruptible UC-complete setups, there exists a protocol which securely realizes any well-formed functionality while tolerating an adversary corrupting arbitrary many parties *if and only if* the adversary might (simultaneously) corrupt less than half of the setups.

The above statement is formalized in Theorem 4. Note that, throughout this section, the assumed functionalities only accept active corruption requests, i.e., they ignore any message of the form ($\texttt{fcorrupt}$, *type*) for *type* $\neq$ $\texttt{active}$.

For proving the following theorem we use a well-known result on robust combiners for Oblivious Transfer (OT). In [23] is was shown that from m copies of the OT functionality at most half of which might be faulty (in our terminology faulty corresponds to actively corrupted) we can implement the *incorruptible* OT functionality $\mathcal{F}_{\text{OT}}^{\not{\mathcal{C}}}$. We use this fact

[9] Note that if $\mathcal{F}$ is adaptively well-formed, then so is its corruption-ignoring strengthening $\mathcal{F}_{\ell}^{\not{\mathcal{C}}}$.

to prove our result; the idea of our proof is the following: We use each of the setups to implement an independent copy of $\mathcal{F}_{\mathrm{OT}}$ and then we use the OT-combiner to construct a single copy of $\mathcal{F}_{\mathrm{OT}}$ that behaves honestly (i.e., as being incorruptible). Note that because the combiner from [23] works even when the adversary corrupts the parties adaptively, our positive results also works for adaptive party-corruption. (The same holds also for tightness as long as the $\mathcal{F}_\ell$'s are adaptively well-formed.)

We remark that although the combiner from [23] does not explicitly claim adaptive corruption of the assumed OTs, or universal composability, we can use the combiner proposed by Wullschleger [26] which is explicitly claimed to be universally composable and to tolerate adaptive corruption of the assumed functionality.[10] Importantly, the combiner from [W09] uses only the input/output of the OLFE protocol (and *not* the OLFE-protocol instructions). We remark that, although the relevant theorem statement [26, Theorem 2] does not explicitly mention UC, it provides a simulation-based proof of the combiner, where the simulator is black-box, straight-line, and does not depend on the actual OLFE implementation but only on the inputs/outputs. Hence, we can use the same simulator-structure with the only difference that when both parties are honest (but some of the OT's are corrupted) one needs to additionally observe that the simulator can simulate the inputs of the (honest) parties to the t corrupted OLFEs; this follows trivially from the fact that (by construction of the combiner) these inputs are shares lying on random polynomials of degree more than t. In the full version we provide the details of how the proof of [26, Theorem 2] can be adapted to work in our setting.

Theorem 4. *Let $\mathcal{M} = \{\mathcal{F}_1, \ldots, \mathcal{F}_m\}$ be a set of setup-functionalities and $\alpha_1, \ldots, \alpha_m$ be assumptions such that for each $\mathcal{F}_\ell \in \mathcal{M}$ the corruption-ignoring strengthening ${\mathcal{F}_\ell}^{\not\mathcal{C}}$ is (UC) α_ℓ-sufficient for OT. Assuming that the adversary is t-restricted, $\mathcal{M}$ is (UC) $\{\alpha_1, \ldots, \alpha_m\}$-complete if and only if $t < m/2$.*

Proof (*sketch).* The necessity of the $t < m/2$ bound follows immediately from Corollary 2 (the case of adaptive party-corruption and adaptively well-formed setups follows from Theorem 3). We next argue its sufficiency: Because each $\mathcal{F}_\ell \in \mathcal{M}$ is α_ℓ-complete, we can use each of them to UC implement an independent copy of the OT functionality $\mathcal{F}_{\mathrm{OT}}$. If $\mathcal{F}_\ell$ is corrupted, then there is no guarantee on the corresponding emulated copy of $\mathcal{F}_{\mathrm{OT}}$. However, if $\mathcal{F}_\ell$ is uncorrupted, then the resulting construction is a good UC-emulation of the OT functionality. Moreover, as more than half of the $\mathcal{F}_\ell$'s are uncorrupted, there are more than $m/2$ emulations of (honest) OT functionalities. But the results of [23, 26] state that from m copies of the OT functionality a majority of which is good (honest) we can implement an incorruptible copy of the OT functionality (in our terminology this corresponds to implementing the corruption-ignoring strengthening ${\mathcal{F}_{\mathrm{OT}}}^{\not\mathcal{C}}$ of $\mathcal{F}_{\mathrm{OT}}$). The UC composition theorem ensures that plugging the $\mathcal{F}_\ell$-to-OT reductions into the OT-combiner (along with an appropriate addressing mechanism) yields a secure protocol for ${\mathcal{F}_{\mathrm{OT}}}^{\not\mathcal{C}}$, i.e., an $\mathcal{M}$-hybrid protocol π_{OT} for securely emulating an incorruptible copy of the OT functionality assuming $t < m/2$. The UC completeness of $\mathcal{M}$ follows then from the (unconditional) UC completeness of OT [21, 10, 19] by applying the UC composition theorem: For any given functionality $\mathcal{F}'$, let $\Pi^{\mathcal{F}_{\mathrm{OT}}}$ denote the OT-hybrid protocol for $\mathcal{F}'$ which is guaranteed to exist from [21, 10, 19]. The UC composition theorem ensures that replacing in $\Pi^{\mathcal{F}_{\mathrm{OT}}}$ every instance of $\mathcal{F}_{\mathrm{OT}}$ by a call to π_{OT} yields a secure protocol for $\mathcal{F}'$.[11] □

As an immediate corollary of Theorem 4 and the UC $\alpha_{\mathtt{shOT}}$-completeness of the CRS functionality [7], we get sufficiency of the $t < m/2$ bound for the m-multi-$\mathcal{F}_{\mathrm{CRS}}$ functionality, which consists of m independent CRSs.

Corollary 5. *Let $\mathcal{M} = \{\mathcal{F}_1, \ldots, \mathcal{F}_m\}$, where for each $\ell \in [m]$: $\mathcal{F}_\ell$ is the CRS functionality. Assuming that the adversary is t-restricted, $\mathcal{M}$ is (UC) $\alpha_{\mathtt{shOT}}$-complete if and only if $t < m/2$.*

A similar possibility result as in the above corollary—i.e., sufficiency of having a majority of "good" CRS's—was proved in [15] for a slightly restricted adversary, who cannot make the CRS send inconsistent values to different parties (i.e., the adversary can only choose the value of the reference string). In fact, this restriction is crucial in the security proof of [15] , as one can verify that otherwise their suggested protocol is insecure. Intuitively, the reason is that the CRS's are parsed as keys for the constructed commitment scheme, hence dropping this assumption means that the parties use inconsistent keys which leads to violation of correctness (an honest party might not be able to open its commitment towards another honest party). Note that the naïve approach for resolving inconsistencies in the distributed reference strings, i.e., have the parties exchange the values they received from the CRS, would not work for the *multi-party* setting, as it would require that all parties can achieve *consensus* on a common string, which is not possible when a majority of the parties is corrupted.

5. PASSIVE (AND MIXED) CORRUPTION OF SETUPS

In this section we investigate the setting where the adversary can passively corrupt setups. We show that the $m/2$ bound which was established in the previous section for active corruption of setups is in fact tight even in this setting. More precisely, we show that there are UC complete setups for which passive corruption is as severe as active corruption. Finally, combining the results from the current and the previous section we obtain a tight feasibility bound for *mixed* corruption of setups, i.e., for an adversary who might both actively and passively corrupt setups.

In the following we show that a collection of m (independent) copies of the OT functionality half of which might be

[10] We point out that the combiner from [26] constructs a Rabin-OT from m copies of the so called *oblivious linear function evaluation (OLFE) functionality* less than half of which might be corrupted. However, one can unconditionally implement OLFE from OT (e.g., by using [19]).

[11] We implicitly assume that each instance of $\mathcal{F}_{\mathrm{OT}}$ within $\Pi^{\mathcal{F}_{\mathrm{OT}}}$ is replaced by an $\mathcal{M}$-hybrid protocol π_{OT} using an independent instance of (each functionality in) $\mathcal{M}$ (see Section 2). If the adversary corrupts some $\mathcal{F}_\ell \in \mathcal{M}$, then he is allowed to corrupt it across all protocols using it as a hybrid "for free" (i.e., this does not count as additional corruptions towards his threshold t). Alternatively, we could have all invocations of π_{OT} use the same hybrid $\mathcal{M}$ and apply the joint state composition theorem (JUC) [9].

only passively corrupted is no stronger than just secure communication. Because $\mathcal{F}_{\mathrm{OT}}$ is deterministic, therefore adaptively well-formed in a trivial sense, the lemma holds even for adaptive party-corruption. The proof of the lemma follows the structure of the proof of Theorems 1 and 3.

Lemma 6. *Let $\mathcal{M} = \{\mathcal{F}_1, \ldots, \mathcal{F}_m\}$, where for each $\ell \in [m]$: $\mathcal{F}_\ell$ is the OT functionality $\mathcal{F}_{\mathrm{OT}}$. For a functionality-corruption structure $\mathcal{K}$, if $Q^2(\mathcal{M}, \mathcal{K})$ does not hold then there exists an $\mathcal{F}_{\mathrm{SEC}}$-hybrid 2-party protocol which securely realizes $\mathcal{M}$, in the presence of an adversary who might corrupt any (or even both) parties.*

As a corollary of the above result (along the lines of Corollary 2) we obtain a simple proof of impossibility of OT-combiners when at least half of the candidates are passively corrupted which holds even for adaptive corruption of the assumed OT functionalities.

Mixed (active and passive) setup-corruption..

We next obtain a feasibility statement for the case of an adversary who might corrupt setups both actively and passively. Consistently with the terminology introduced in Section 2, we say that an adversary is (t_a, t_p)*-restricted* if he corrupts up to t_a functionalities actively and (at the same time) up to t_p functionalities passively. The proof of the theorem follows from Theorem 4 and the fact that passive corruption is weaker than active. The simplicity of the proof demonstrates the power of having corruption types that can be ordered with respect to their strength which is one of the advantages of our model with respect to [12].

Theorem 7. *Let $\mathcal{M} = \{\mathcal{F}_1, \ldots, \mathcal{F}_m\}$ be a set of setup-functionalities and $\alpha_1, \ldots, \alpha_m$ be assumptions so that the corruption-ignoring strengthening ${\mathcal{F}_\ell}^{\mathbb{C}}$ of each $\mathcal{F}_\ell \in \mathcal{M}$ is (UC) α_ℓ-complete. Assuming the adversary is (t_a, t_p)-restricted, $\mathcal{M}$ is (UC) $\{\alpha_1, \ldots, \alpha_m\}$-complete if $t_a + t_p < m/2$.*

Proof. Because passively corrupting a setup gives the adversary strictly less power than actively corrupting it, a protocol which tolerates $t_a + t_p$ actively corrupted setups is also secure for t_a actively and t_p passively corrupted setups. Because $t_a + t_p < m/2$, the existence of such a protocol is guaranteed by Theorem 4. □

5.1 Passive Immunity

The above feasibility result is tight for certain complete setups, e.g., for the OT. Indeed, tightness of the bound in Theorem 7 when $\mathcal{F}_\ell = \mathcal{F}_{\mathrm{OT}}$ for all $\ell \in [m]$ follows easily from Lemma 6: Assume, towards contradiction that a generic MPC protocol exists tolerating t_a actively and t_p passively corrupted (arbitrary) setups, where $t_a + t_p > m/2$. Then this protocol can be use to implement a non-trivial functionality (e.g., OT) from m copies of the OT out of which $t_a + t_p$ could be passively corrupted, contradicting Lemma 6.

Interestingly, as it turns out for many of the setups used in the literature giving the power of passive corruption to the adversary "does *not* buy her much". These include, for example, the CRS or the version and the Key Registration functionality, denoted as $\mathcal{F}_{\mathrm{KR}}(\mathcal{P}, f)$, where the keys are sampled according to a given distribution generated by a function f (still, corrupted parties get to choose their coins). To model this fact, we introduce the notion of *passively immune* setups which, informally, satisfy the property that their incorruptible version reduces to the version that might be at most passively (but not actively) corrupted.

Definition 1. For some assumption α, a (wrapped) functionality $\mathcal{F}$ is said to be α*-passively immune* if for the actively ignoring strengthening $\mathcal{F}^{\mathbb{A}}$ of $\mathcal{F}$ (which, recall, accepts only passive-corruption requests) there exists an $\mathcal{F}^{\mathbb{A}}$-hybrid protocol which securely realizes (assuming α) the corruption-ignoring strengthening $\mathcal{F}^{\mathbb{C}}$ of $\mathcal{F}$.

Note that the uniformly random reference string functionality $\mathcal{F}_{\mathrm{URS}}$ is trivially passively immune as the adversary does not learn anything by passively corrupting it. Furthermore, for the case of the (imperfect) CRS functionality, $\mathcal{F}_{\mathrm{CRS}}(\mathcal{P}, f)$ which samples the reference string using a sampling algorithm f, the results by Canetti, Pass, and shelat [8, 22, 25] imply that $\mathcal{F}_{\mathrm{CRS}}(\mathcal{P}, f)$ is passively immune under the following assumption: f is polynomially computable and $\mu(\lambda) - d(\lambda) > \lambda^\varepsilon$ for some $\varepsilon > 0$, where $d(\lambda)$ is the size of the description of f and $\mu(\lambda)$ is the min-entropy of the corresponding distribution (λ denotes the length of the reference string).

In what follows, we show that under standard complexity assumptions, i.e., semi-honest OT ($\alpha_{\mathtt{shOT}}$) and one-way permutations ($\alpha_{\mathtt{OWP}}$), two additional important setups used in the UC literature are passively immune. To prove passively immunity of these setups $\mathcal{F}$ we use *UC puzzles* [22]. On a high level, a UC puzzle is a two party protocol with the following property: no adversary can successfully complete the puzzle while simultaneously obtaining a trapdoor associated with the puzzle; however, there does exist a simulator $\mathcal{S}$ who can generate statistically close puzzle transcripts and the associated trapdoors. We refer to the appendix for a more formal description and a "tweak" of the definition from [22] which is used in our proofs.

As proved in [22], UC puzzles are $\alpha_{\mathtt{shOT}}$-complete. Thus, to prove our result, we show that there exists an $\mathcal{F}^{\mathbb{A}}$-hybrid UC puzzle, where $\mathcal{F}^{\mathbb{A}}$ denotes the actively ignoring strengthening of $\mathcal{F}$. For readers familiar with the use of UC puzzles, the high-level idea is to have the UC puzzle simulator "hide" the corresponding trapdoor in the pre-image of a one-way permutation. We point out that the lemma is stated and proved for the case of static party-corruption; however using ideas from [11] (and the stronger assumptions used there) one can extend the proof to the adaptive setting. A proof can be found in the full version.

Lemma 8. *Let f be a second-preimage collision resistant function.*[12] *The functionalities for the Common Reference String $\mathcal{F}_{\mathrm{CRS}}(\mathcal{P}, f)$ and the Key Registration $\mathcal{F}_{\mathrm{KR}}(\mathcal{P}, f)$ are $\{\alpha_{\mathtt{OWP}}, \alpha_{\mathtt{shOT}}\}$-passively immune assuming the adversary statically chooses the set of corrupted parties.*

The following theorem establishes a tight feasibility bound for passively immune setups with a mixed (active/passive) adversary.

Theorem 9. *For a setup set $\mathcal{M} = \{\mathcal{F}_1, \ldots, \mathcal{F}_m\}$ let $\alpha_1, \ldots, \alpha_m$ be assumptions such that for each $\mathcal{F}_\ell \in \mathcal{M}$ the following properties hold: $\mathcal{F}_\ell$ is α_ℓ-passively immune and the corruption-ignoring strengthening ${\mathcal{F}_\ell}^{\mathbb{C}}$ is α_ℓ-complete. Assuming that the adversary is (t_a, t_p)-restricted and corrupts parties non-adaptively, $\mathcal{M}$ is (UC) $\{\alpha_1, \ldots, \alpha_m\}$-complete if and only if $t_a < m/2$ (and thus for* arbitrary *$t_p \leq m$).*

[12] Note that existence of such f is implied by $\alpha_{\mathtt{OWP}}$, and in particular f can be any injective function.

Proof *(sketch).* The necessity of the bound follows trivially from Corollary 2. We next argue the sufficiency: Because for each $\mathcal{F}_\ell \in \mathcal{M}$: $\mathcal{F}_\ell$ is α_ℓ-passively immune and ${\mathcal{F}_\ell}^{\not\subset}$ is α_ℓ-complete we can use each of them to generate an independent copy of the OT functionalities. The statement follows then from Theorem 4. □

6. EXTENSIONS

In the full version of this paper we demonstrate the flexibility of our model by using it to prove the following results. We provide a "best-of-both worlds" result for the setting where there is either a corrupt majority of setups, or a corrupt majority of parties.

Theorem 10. *(informal) For a set of m UC-complete setup-functionalities and any given thresholds $s < n/2$ and t such that $s + t < n$, there exists a protocol which securely realizes any well-formed n-party functionality tolerating* static *party-corruption if some of the following holds: (1) at most s parties are corrupted and arbitrary many of the setup-functionalities are corrupted, or (2) up to $t > s$ parties are corrupted but at most $t' < m/2$ functionalities are actively-corrupted.*

Additionally, we extend our positive result for actively-corruptible functionalities by additionally considering fail-corruption.

Theorem 11. *(informal) For a set of m UC complete setup-functionalities, if the adversary adaptively corrupts up to t_a of them actively and fail corrupts up to t_f of them (and corrupts arbitrarily many parties actively), then there exists a protocol which securely realizes any well-formed functionality if and only if $2t_a + t_f < m$.*

Finally, we we generalize our positive result to the setting of an arbitrary functionality-corruption structure $\mathcal{K}$ (instead of a threshold structure). Recall that $\mathcal{K}$ is an enumeration of all possible corruptible sets of functionalities.

Theorem 12. *(informal) For a set $\mathcal{M}$ of (possibly different) actively corruptible UC-complete setups and a functionality corruption structure $\mathcal{K}$, there exists a protocol which securely realizes any well-formed functionality while tolerating a $\mathcal{K}$-restricted adversary corrupting arbitrary many parties* if and only if *$Q^2(\mathcal{M}, \mathcal{K})$ holds.*

7. ACKNOWLEDGMENTS

We thank Yuval Ishai for his comments during the early stages of our work.

8. REFERENCES

[1] B. Barak, R. Canetti, J. B. Nielsen, and R. Pass. Universally composable protocols with relaxed set-up assumptions. In E. Upfal, editor, *FOCS 2004*, pages 186–195. IEEE Computer Society Press, Oct. 2004.

[2] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2005. An extended abstract appeared in FOCS 2001.

[3] R. Canetti. Obtaining universally compoable security: Towards the bare bones of trust (invited talk). In K. Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 88–112. Springer, Dec. 2007.

[4] R. Canetti, Y. Dodis, R. Pass, and S. Walfish. Universally composable security with global setup. In S. P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 61–85. Springer, Feb. 2007.

[5] R. Canetti and M. Fischlin. Universally composable commitments. In J. Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 19–40. Springer, Aug. 2001.

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

[7] R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. Universally composable two-party and multi-party secure computation. In J. H. Reif, editor, *STOC 2002*, pages 494–503. ACM Press, May 2002.

[8] R. Canetti, R. Pass, and A. Shelat. Cryptography from sunspots: How to use an imperfect reference string. In A. Sinclair, editor, *FOCS 2007*, pages 249–259. IEEE Computer Society Press, Oct. 2007.

[9] R. Canetti and T. Rabin. Universal composition with joint state. In D. Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 265–281. Springer, Aug. 2003.

[10] C. Crépeau, J. van de Graaf, and A. Tapp. Committed oblivious transfer and private multi-party computation. In D. Coppersmith, editor, *CRYPTO'95*, volume 963 of *LNCS*, pages 110–123. Springer, Aug. 1995.

[11] D. Dachman-Soled, T. Malkin, M. Raykova, and M. Venkitasubramaniam. Adaptive and concurrent secure computation from new adaptive, non-malleable commitments. In *ASIACRYPT 2013*, pages 316–336. Springer, Dec. 2013.

[12] S. Garg, V. Goyal, A. Jain, and A. Sahai. Bringing people of different beliefs together to do UC. In Y. Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 311–328. Springer, Mar. 2011.

[13] S. Goldwasser and Y. Lindell. Secure multi-party computation without agreement. *Journal of Cryptology*, 18(3):247–287, July 2005. Preliminary version appeared at DISC 2002.

[14] V. Goyal and J. Katz. Universally composable multi-party computation with an unreliable common reference string. In R. Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 142–154. Springer, Mar. 2008.

[15] J. Groth and R. Ostrovsky. Cryptography in the multi-string model. In A. Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 323–341. Springer, Aug. 2007.

[16] D. Harnik, J. Kilian, M. Naor, O. Reingold, and A. Rosen. On robust combiners for oblivious transfer and other primitives. In R. Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 96–113. Springer, May 2005.

[17] M. Hirt and U. M. Maurer. Complete characterization of adversaries tolerable in secure multi-party computation (extended abstract). In *16th ACM PODC*, pages 25–34. ACM Press, Aug. 1997.

[18] D. Hofheinz and J. Müller-Quade. Universally composable commitments using random oracles. In

M. Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 58–76. Springer, Feb. 2004.

[19] Y. Ishai, M. Prabhakaran, and A. Sahai. Founding cryptography on oblivious transfer - efficiently. In D. Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 572–591. Springer, Aug. 2008.

[20] J. Katz. Universally composable multi-party computation using tamper-proof hardware. In M. Naor, editor, *EUROCRYPT 2007*, volume 4515 of *LNCS*, pages 115–128. Springer, May 2007.

[21] J. Kilian. Founding crytpography on oblivious transfer. In *STOC '88*, pages 20–31, New York, NY, USA, 1988. ACM.

[22] H. Lin, R. Pass, and M. Venkitasubramaniam. A unified framework for concurrent security: universal composability from stand-alone non-malleability. In M. Mitzenmacher, editor, *STOC 2009*, pages 179–188. ACM Press, May / June 2009.

[23] R. Meier, B. Przydatek, and J. Wullschleger. Robuster combiners for oblivious transfer. In S. P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 404–418. Springer, Feb. 2007.

[24] M. Rabin. How to exchange secrets by oblivious transfer. Technical Report TR-81,Harvard Aiken Computation Laboratory, 1981.

[25] M. Venkitasubramaniam. A universal framework for concurrent security. PhD thesis, Cornell University, 2010.

[26] J. Wullschleger. Efficiently from semi-honest to malicious ot via OLFE. Cryptology ePrint Archive, Report 2009/428, 2009.

APPENDIX

A. UC PUZZLES

We sketch the (general) definition of UC puzzles from [22] to the standard setting, i.e., with a non-uniform PPT environment $\mathcal{Z}$, a uniform PPT adversary $\mathcal{A}$ and a uniform PPT simulator $\mathcal{S}$. We let $\langle S, R\rangle$ represent a protocol between two parties, the sender S and the receiver R. We consider a concurrent puzzle execution in which $\mathcal{A}$ participates as a sender concurrently in $m = poly(k)$ puzzles (where k is a security parameter) with honest receivers $R_1, \ldots, R_m$. We now describe the real and the ideal executions.

Real execution. In the real execution, the adversary $\mathcal{A}$ on input 1^k, interacts with a puzzle-environment $\mathcal{Z}$ and participates as a sender in m interactions using $\langle S, R\rangle$. After every puzzle-interaction, $\mathcal{A}$ sends trans to $\mathcal{Z}$, where trans is the puzzle-transcript. We denote the probability ensemble corresponding to the output of $\mathcal{Z}$ at the end of the execution by $\{\text{REAL}_{\mathcal{A},\mathcal{Z}}(k)\}_{k\in\mathbb{N}}$.

Ideal execution. In the ideal execution, there is a simulator $\mathcal{S}$ that on input 1^k interacts with puzzle-environment $\mathcal{Z}$. Note that $\mathcal{S}$ has a special output tape which is not accessible to $\mathcal{Z}$. We denote the probability ensemble corresponding to the output of $\mathcal{Z}$ at the end of the execution by $\{\text{IDEAL}_{\mathcal{S},\mathcal{Z}}(k)\}_{k\in\mathbb{N}}$.

Definition (UC PUZZLE [22]). A pair $(\langle S, R\rangle, \mathcal{R})$, where $\mathcal{R}$ is a polynomial time computable binary relation, is a UC-puzzle if the following conditions hold:

SOUNDNESS: For every malicious PPT receiver R^*, there exists a negligible function $negl(\cdot)$ such that the probability that R^*, after an execution with S on common input 1^k, outputs τ such that $\mathcal{R}(\mathsf{trans}, \tau) = 1$ (where trans is the transcript of the messages exchanged in the interaction) is at most $negl(k)$.

STATISTICAL SIMULATABILITY: For every adversary $\mathcal{A}$ participating in a concurrent puzzle execution there is a simulator $\mathcal{S}$ such that for all puzzle environments $\mathcal{Z}$, the ensembles $\{\text{REAL}_{\mathcal{A},\mathcal{Z}}(k)\}_{k\in\mathbb{N}}$ and $\{\text{IDEAL}_{\mathcal{S},\mathcal{Z}}(k)\}_{k\in\mathbb{N}}$ are statistically close over $k \in \mathbb{N}$; moreover, whenever $\mathcal{S}$ outputs a message of the form trans to $\mathcal{Z}$, it also outputs τ in a special output tape (which cannot be accessed by $\mathcal{Z}$) such that $\mathcal{R}(\mathsf{trans}, \tau) = 1$.

Lin et al. [22, 25] proved the following theorem stating that UC puzzles are $\alpha_{\mathtt{shOT}}$-complete.

THEOREM 13 ([22, 25]). *Assume the existence of a $t_1(\cdot)$-round UC-secure puzzle Σ using some set-up $\mathcal{F}$, the existence of a $t_2(\cdot)$-round standalone secure semi-honest oblivious transfer protocol. Then for every m-ary functionality $\mathcal{G}$, there exists an $O(t_1(\cdot) + t_2(\cdot))$-round protocol Π—using the same set-up $\mathcal{F}$—that UC-realizes $\mathcal{G}$.*

UC puzzles with Strong Statistical Simulatability.

In order to be able to tolerate adaptive corruption of the assumed setups, our protocols need puzzles that satisfy a somewhat strengthened version of the statistical simulatability property, namely we want to ensure that the simulator (playing as the receiver) is able to simulate *even if the adversarial sender corrupts the hybrid (i.e., the setup) used by the puzzle protocol during its execution* (however, if such a corruption request occurs we no longer require the simulator to output a trapdoor for the puzzle). We refer to this property as *strong statistical simulatability.*

Note that for an adversary who (non-adaptively) corrupts the hybrid functionality before the beginning of any puzzle, the strong statistical simulatability property is implied by the definition of the UC puzzle: Indeed, if $\mathcal{Z}$ requests to corrupt the hybrid $\mathcal{F}$ at the beginning then the simulator simply sends $\mathcal{Z}$ the code and randomness and forwards all calls of the (simulated) puzzle protocol to $\mathcal{Z}$ from that point on; otherwise, strong simulatability follows trivially from the statistical simulatability property of the UC puzzle. However, for an adversary who might (adaptively) corrupt the hybrid functionality at any point in the protocol execution, the strong statistical simulatability property is not necessarily implied from the definition of UC puzzles. Nevertheless, one can verify that for most, if not all, setups for which a UC puzzle is described in the literature [25, 12] this property is indeed satisfied, as the simulator considered in these puzzles explicitly emulates the execution of the hybrid. Furthermore, most of these puzzles exist under the semi-honest OT assumption $\alpha_{\mathtt{shOT}}$ which, as shown in [25, 12] is also sufficient for the completeness of standard UC puzzles.