Adaptive Security of Multi-Party Protocols, Revisited

Martin Hirt, Chen-Da Liu-Zhang*, and Ueli Maurer

{hirt,maurer}@inf.ethz.ch, ETH Zurich
cliuzhan@andrew.cmu.edu, Carnegie Mellon University

Abstract. The goal of secure multi-party computation (MPC) is to allow a set of parties to perform an arbitrary computation task, where the security guarantees depend on the set of parties that are corrupted. The more parties are corrupted, the less is guaranteed, and typically the guarantees are completely lost when the number of corrupted parties exceeds a certain corruption bound.
Early and also many recent protocols are only statically secure in the sense that they provide no security guarantees if the adversary is allowed to choose adaptively which parties to corrupt. Security against an adversary with such a strong capability is often called *adaptive security* and a significant body of literature is devoted to achieving adaptive security, which is known as a difficult problem. In particular, a main technical obstacle in this context is the so-called "commitment problem", where the simulator is unable to consistently explain the internal state of a party with respect to its pre-corruption outputs. As a result, protocols typically resort to the use of cryptographic primitives like non-committing encryption, incurring a substantial efficiency loss.
This paper provides a new, clean-slate treatment of adaptive security in MPC, exploiting the specification concept of constructive cryptography (CC). A new natural security notion, called CC-adaptive security, is proposed, which is technically weaker than standard adaptive security but nevertheless captures security against a fully adaptive adversary. Known protocol examples separating between adaptive and static security are also insecure in our notion. Moreover, our notion avoids the commitment problem and thereby the need to use non-committing or equivocal tools. We exemplify this by showing that the protocols by Cramer, Damgard and Nielsen (EUROCRYPT'01) for the honest majority setting, and (the variant without non-committing encryption) by Canetti, Lindell, Ostrovsky and Sahai (STOC'02) for the dishonest majority setting, achieve CC-adaptive security. The latter example is of special interest since all UC-adaptive protocols in the dishonest majority setting require some form of non-committing or equivocal encryption.

1 Introduction

1.1 Multi-Party Computation

Secure multi-party computation (MPC) is one of the most fundamental problems in cryptography. It considers the setting where a set of parties wish to carry out a computation in a *secure* manner, where security informally means that parties obtain the correct output of the computation, while at the same time keeping their local inputs as private as possible.

A crucial step towards meaningfully designing and analyzing cryptographic protocols is to come up with appropriate definitions of security. Formulating good definitions is highly non-trivial: the definition should closely capture the aspects that we care about, while at the same time being simple and usable, even minimal, avoiding as much as possible unnecessary artifacts.

There is a vast literature on security definitions in the field of MPC. Initial works [26, 41, 3, 8, 21] considered the *stand-alone* setting, which examines only the protocol at hand and does not capture what it means to use the protocol in a larger context, for the task of secure function evaluation [48, 49, 25]. It was not until several years later, that definitions in so-called composable frameworks for general reactive tasks were introduced [45, 9, 20, 38, 42, 33, 29]. Such definitions aim to capture all aspects of a protocol that can be relevant, with respect to any possible application, hence the term *universal composability* [9].

An important aspect of security definitions for secure computation is the way in which the corrupted parties are chosen. Here, two models are commonly considered. The static security model assumes that the set of corrupted parties is fixed before the computation starts and does not change. In the more general adaptive security model, the adversary may corrupt parties during the protocol execution, based on information that has been gathered so far. Indeed, adaptive security captures important concerns

* This work was partially carried out while the author was at ETH Zurich.

regarding cryptographic protocols that static security does not capture. These include scenarios where attackers, viruses, or other adversarial entities can take advantage of the communication to decide which parties to corrupt.

The currently considered standard MPC definition for adaptive security is the one introduced by Canetti [9] in the UC framework. The UC-adaptive security definition follows the well-known simulation paradigm, and is formalized by comparing the execution of the protocol in the real world, to an ideal world that has the desired security properties by design. Intuitively, it guarantees that for any attack in the real world performed by an adversary that can adaptively corrupt parties, the attack can be equivalently performed in the ideal world, achieving a similar effect. This is formalized by the existence of a single simulator that has to simulate the entire protocol execution, with respect to any environment where the protocol is being executed.

1.2 The Commitment Problem in Adaptive Security

Despite the fact that the current standard notion has been the cornerstone of adaptive security in MPC and has lead to the development of many beautiful cryptographic protocols and primitives, one could argue that the definition is too strong.

To show this, consider the following example: Let $\boldsymbol{\pi}$ be any protocol for secure function evaluation that is adaptively secure. Now, consider a modified protocol $\tilde{\boldsymbol{\pi}}$, where each party i first commits to its input using for example any (non-equivocable) perfectly hiding and computationally binding commitment scheme and publishes the commitment. Then, all parties execute the protocol $\boldsymbol{\pi}$. The commitments are never again used, and in particular they are never opened. Intuitively, protocol $\tilde{\boldsymbol{\pi}}$ *should* be adaptively secure, since the commitments do not reveal any secret information (the commitments are even statistically independent of the inputs!). However, protocol $\tilde{\boldsymbol{\pi}}$ is no longer adaptively secure: we run into the so-called *commitment problem*, where the simulator is unable to consistently explain the internal state of the parties that are adaptively corrupted. This is because the simulator first has to publish a commitment on behalf of each honest party without knowing its input, and later, upon corruption, output an internal state on behalf of each party that is consistent with its input and the previously published commitment.

Common ways to address this issue include the use of non-committing encryption (see e.g. [12, 13]), or the availability of secure erasable memory (see e.g. [4]), therefore incurring to a substantial efficiency loss or an extra assumption.

However, at a more general level, this raises the question of whether one could have an alternative security definition that is not subject to this issue, but still captures natural security guarantees under adaptive corruption:

Is there a natural MPC security definition that captures security guarantees under adaptive corruption and is not subject to the commitment problem?

There have been a number of works that aimed to solve this issue. A line of work [44, 46, 7] considers simulators that have super-polynomial running time. Such approaches come at the price of being technical or sacrificing composition guarantees. Another approach [1] disallows certain activation sequences by the environment that cannot be simulated, avoiding some of the complications of the other approaches, but sacrificing some guarantees by excluding certain attacks. A recent work [30] addressed this issue by proposing a notion that formalizes guarantees that hold within a certain interval, between two events, and requiring the simulation to work within each interval, without forcing the simulation to be consistent between the intervals. Although this approach seems promising, the guarantees that are given turn out to be too weak for MPC applications. In particular, the corruptions can only depend on "external" events, and not on the outputs from the given resources.

1.3 Contributions

CC-Adaptive Security. Intuitively, an MPC protocol should provide, at any point during the protocol execution, security guarantees to the set of honest parties at that point. That is, for every set of parties, there is a guarantee as long as these parties are honest. This is exactly what CC-adaptive security captures: we phrase the guarantees naturally as the intersection (i.e. conjunction) of the guarantees for

every set of so-far honest parties. Informally, we require the same protocol to realize a possibly different functionality $\mathcal{F}_X$ for each subset X of parties. In each statement, there must exist a simulator that correctly simulates the protocol execution, as long as the parties in X are honest, without having access to the secret inputs and outputs of parties in X. As soon as a party in X gets corrupted, the guarantee for this set is dropped. (However, guarantees for other so-far honest sets still remain.)

The corruptions are completely adaptive in the strong and usual sense, where the selection of which parties become corrupted can be done based on information gathered during the protocol execution. The more parties are corrupted, the less guarantees remain.

Technically, the commitment problem does not arise because the guarantees are dropped (i.e. the simulation stops) at the point where a party in X gets corrupted. Therefore, the simulator does not need to explain the secret state of a party in X. This is in contrast to previous adaptive security definitions, which require the existence of a single simulator that explains all possible cases.

The described guarantees are naturally phrased within the constructive cryptography (CC) [37, 38, 39] composable framework, where each guarantee corresponds to a set specification of systems, and the conjunction of guarantees is simply the intersection of specifications. The protocol can then achieve all these guarantees within a single construction statement.

Comparison with Standard Static and Adaptive Security. At a technical level, we show that our new definition lies in-between the current standard UC-security definitions for static and adaptive security, respectively. Interestingly, popular examples that separate the standard static and adaptive security notions and do not exploit the commitment problem, also separate static from CC-adaptive security, therefore giving evidence that CC-adaptive security gives strong adaptive security guarantees. More concretely, we show the following.

Static vs CC-Adaptive Security. We first show that CC-adaptive security implies static security in all settings. Moreover, we also show that CC-adaptive security is strictly stronger than static security: for the case of passive corruption and a large number of parties, the protocol shown in [12] separates the notions of static and CC-adaptive security, and in the case of active corruption and at least three parties, the protocol shown in [10] makes the separation.

Adaptive vs CC-Adaptive Security. We show that UC-adaptive security is strictly stronger than CC-adaptive security in all settings, by showing a protocol example based on the commitment problem.

Applications. We demonstrate the usefulness of our notion with two examples, showing that known protocols achieve strong adaptive security guarantees without the use of non-committing or equivocal encryption.

CDN Protocol. First, we show that the protocol by Cramer, Damgard and Nielsen [17] (CDN) based on threshold (additively) homomorphic encryption (THE) achieves CC-adaptive security in the honest majority setting. In the passive corruption setting, the protocol is described assuming solely the key setup for the THE scheme, while in the active corruption setting, the protocol is described assuming in addition a multi-party zero-knowledge functionality. This shows that the CDN protocol approach achieves strong adaptive security guarantees as-is, even when using an encryption scheme that commits to the plaintext.

CLOS Protocol. Second, we show that the variant of the protocol by Canetti, Lindell, Ostrovsky and Sahai [13] (CLOS) that does not use non-committing encryption, previously only proven statically secure, actually achieves CC-adaptive security in the dishonest majority setting. This is achieved by showing that the oblivious transfer from [25] achieves CC-adaptivity, and the CLOS compiler transforming passive to active protocols preserves CC-adaptivity. Note that, to the best of our knowledge, all previous UC-adaptive protocols in the dishonest majority setting required some form of non-committing or equivocal encryption.

1.4 Further Related Work

The problem of MPC with adaptive security was first studied by Canetti, Feige, Goldreich and Naor [12], and there is a large literature on MPC protocols with adaptive security. In the case of honest majority, it was shown that classical MPC protocols are adaptively secure [5, 15, 47]. Using the results in [32, 31], it was shown that these protocols achieve UC adaptive security with abort in the plain model, or guaranteed output delivery in the synchronous model. A more efficient protocol was shown in [19], following the CDN-approach based on threshold homomorphic encryption and assuming a CRS. In the case of dishonest majority, the protocols achieve security with abort, and all known protocols assume

some form of non-committing encryption or equivocation. The first work achieving adaptive security for dishonest majority was the protocol by Canetti, Lindell, Ostrovsky and Sahai [13], assuming a CRS setup. Since then, several subsequent works have improved its round and communication complexity (e.g. [18, 23, 16, 6, 14]). The work by Garg and Sahai [24] considered adaptive security in the stand-alone model without trusted setup.

The work by Garay, Wichs and Zhou [22] consider the notion of *semi-adaptive* security for two parties, which considers guarantees for the case where one party is corrupted, and the other party is honest and can be adaptively corrupted. In contrast, our security notion imposes guarantees also when both parties start being honest.

2 Preliminaries: Constructive Cryptography

The basic concepts of the Constructive Cryptography framework by Maurer and Renner [38, 37, 39] needed for this paper are quite natural and are summarized below.

2.1 Specifications and Constructions

A basic idea, which one finds in many disciplines, is that one considers a set Φ of objects and *specifications* of such objects. A specification $\mathcal{U} \subseteq \Phi$ is a subset of Φ and can equivalently be understood as a predicate on Φ defining the set of objects satisfying the specification, i.e., being in $\mathcal{U}$. Examples of this general paradigm are the specification of mechanical parts in terms of certain tolerances (e.g. the thickness of a bolt is between 1.33 and 1.34 millimeters), the specification of the property of a program (e.g. the set of programs that terminate, or the set of programs that compute a certain function within a given accuracy and time limit), or in a cryptographic context the specification of a close-to-uniform n-bit key as the set of probability distributions over $\{0,1\}^n$ with statistical distance at most ϵ from the uniform distribution.

A specification corresponds to a guarantee, and smaller specifications hence correspond to stronger guarantees. An important principle is to *abstract* a specification $\mathcal{U}$ by a larger specification $\mathcal{V}$ (i.e., $\mathcal{U} \subseteq \mathcal{V}$) which is simpler to understand and work with. One could call $\mathcal{V}$ an ideal specification to hint at a certain resemblance with terminology often used in the cryptographic literature. If a construction (see below) requires an object satisfying specification $\mathcal{V}$, then it also works if the given object actually satisfies the stronger specification $\mathcal{U}$.

A *construction* is a function $\gamma : \Phi \to \Phi$ transforming objects into (usually in some sense more useful) objects. A well-known example of a construction useful in cryptography, achieved by a so-called extractor, is the transformation of a pair of independent random variables (say a short uniform random bit-string, called seed, and a long bit-string for which only a bound on the min-entropy is known) into a close-to-uniform string.

A construction statement of specification $\mathcal{S}$ from specification $\mathcal{R}$ using construction γ, denoted $\mathcal{R} \xrightarrow{\gamma} \mathcal{S}$, is of the form

$$\mathcal{R} \xrightarrow{\gamma} \mathcal{S} \quad :\Longleftrightarrow \quad \gamma(\mathcal{R}) \subseteq \mathcal{S}.$$

It states that if construction γ is applied to any object satisfying specification $\mathcal{R}$, then the resulting object is guaranteed to satisfy (at least) specification $\mathcal{S}$.

The composability of this construction notion follows immediately from the transitivity of the subset relation:

$$\mathcal{R} \xrightarrow{\gamma} \mathcal{S} \ \wedge \ \mathcal{S} \xrightarrow{\gamma'} \mathcal{T} \implies \mathcal{R} \xrightarrow{\gamma' \circ \gamma} \mathcal{T}.$$

2.2 Resources and Converters

The above natural and very general viewpoint of specifications is taken in Constructive Cryptography, where the objects in Φ are systems, called *resources*, with interfaces to the parties considered in the given setting.

Resources. A resource R is a reactive system with interfaces. Formally, they are modeled as random systems [36, 40], where the interface address and the actual input value are encoded as part of the input. Then, the system answers with an output value at the same interface. One can take several independent

resources $R_1, \ldots, R_k$, and form a new resource $[R_1, \ldots, R_k]$, with the interface set being the union. This resource is denoted as the parallel composition.

Converters. A converter models the local actions executed by a party at its interface, which can be thought of as a system or protocol engine. Formally, converters are modeled as random systems with two interfaces, an outside interface and an inside interface. At its inside, the converter gives input to the party's interface of the resource and at the outside it emulates an interface (of the transformed resource). Upon an input at an outside interface, the converter is allowed to make a bounded number of queries to the inside interfaces, before returning a value at the queried interface. Applying a converter induces a mapping $\Phi \to \Phi$. We denote the set of converters as Σ.

For a converter α and a resource R, we denote by $\alpha^i R$ the resource obtained from applying the converter to the resource at interface i. One can then see that converter attachment satisfies *composition order invariance*, meaning that applying converters at distinct interfaces commutes. That is, for any converters α and β, any resource R and any disjoint interfaces j, k, we have that $\alpha^j \beta^k R = \beta^k \alpha^j R$.

Distinguisher. A distinguisher D is a reactive system that interacts with a resource by making queries at its interfaces, and outputs a bit. The advantage of D in distinguishing two resources R and T is defined as

$$\Delta^D(R, S) \coloneqq \Pr[D(S) = 1] - \Pr[D(R) = 1].$$

2.3 Relaxations

Often a construction statement does not achieve a desired specification $\mathcal{S}$, but only a *relaxed* version of $\mathcal{S}$. We capture this via so-called relaxations [39], which map specifications to weaker, or relaxed, specifications. A relaxation formalizes the idea that we are often happy with resources being almost as good as a target resource specification. For example, one could consider the relaxation that maps a resource S to the set of resources that are indistinguishable from S.

Definition 1. *Let Φ denote the set of all resources. A relaxation $\phi : \Phi \to 2^\Phi$ is a function such that $R \in \phi(R)$, for all $R \in \Phi$. In addition, for a specification $\mathcal{R}$, we define $\mathcal{R}^\phi \coloneqq \bigcup_{R \in \mathcal{R}} \phi(R)$.*

Relaxations satisfy two important properties. The first, is that $\mathcal{S} \subseteq \mathcal{S}^\phi$. And the second, is that if $\mathcal{R} \subseteq \mathcal{S}$ then $\mathcal{R}^\phi \subseteq \mathcal{S}^\phi$. This simplifies the modular analysis, as it means that one can typically consider assumed resources that are completely ideal, or not relaxed. More concretely, from the statements $\mathcal{R} \subseteq \mathcal{S}^\phi$ and $\mathcal{S} \subseteq \mathcal{T}^{\phi'}$, one can conclude that $\mathcal{R} \subseteq \mathcal{T}^{\phi \circ \phi'}$.

In the following, we introduce a few generic types of relaxations [39, 30] that we will use throughout the paper.

ϵ-Relaxation. We introduce a fundamental relaxation that captures computational security based on explicit reductions. For that, we define a function ϵ that maps distinguishers to their respective advantage in $[0, 1]$. The usual interpretation is that $\epsilon(D)$ is the advantage in the underlying computational problem of the distinguisher which is modified by the reduction.

Definition 2. *Let ϵ be a function that maps distinguishers to a real value in $[0, 1]$. We define the ϵ-relaxation of a resource R as:*

$$R^\epsilon \coloneqq \{S \in \Phi \mid \forall D : \Delta^D(R, S) \leq \epsilon(D)\}.$$

Until-Relaxation. Sometimes we want to consider guarantees that hold up to the point where a certain event happens. This is formally modeled by considering an additional so-called monotone binary output (MBO) [40], which is a binary value that can switch from 0 to 1, but not back. Such an MBO can for example model that all inputs to the system are distinct (no collisions).

Definition 3. *Let R be a resource, and let $\mathcal{E}$ be an MBO for the resource. We denote by $\mathsf{until}_\mathcal{E}(R)$ the resource that behaves like R, but halts when $\mathcal{E} = 1$. That is, for any inputs from the point when $\mathcal{E} = 1$ (and including the input that triggered the condition), the output is $\perp$.*

The until-relaxation of a system R [30] consists of the set of all systems, that behave equivalently up to the point where the MBO is set to 1.

Definition 4. *Let R be a resource, and let $\mathcal{E}$ be an MBO for the resource. The $\mathcal{E}$-until-relaxation of R, denoted $R^{\mathcal{E}]}$, is the set of all systems that have the same behavior as R until $\mathcal{E} = 1$. That is,*

$$R^{\mathcal{E}]} \coloneqq \{S \in \Phi \mid \mathsf{until}_{\mathcal{E}}(R) = \mathsf{until}_{\mathcal{E}}(S)\}.$$

Combined Relaxation. In this paper we are interested in the relaxation that corresponds to the intuitive interpretation of "the set of all systems that behave equally until $\mathcal{E} = 1$ given that the assumption of ϵ is valid". However, it was proven in [30] that the ϵ-relaxation and the until-relaxation do not generally commute, i.e., $(\mathcal{R}^{\mathcal{E}]})^{\epsilon} \not\subseteq (\mathcal{R}^{\epsilon})^{\mathcal{E}]}$ and $(\mathcal{R}^{\mathcal{E}]})^{\epsilon} \not\supseteq (\mathcal{R}^{\epsilon})^{\mathcal{E}]}$, and therefore it is not clear whether any of the two corresponds to the intuitive interpretation. Moreover, choosing one of these would partially limit the composability of such statements. That is, if one construction assumes $\mathcal{S}^{\mathcal{E}]}$ to construct $\mathcal{T}$, and another one constructs $\mathcal{S}^{\epsilon}$, then adjusting the first construction to use $\mathcal{S}^{\epsilon}$ is not trivial. Following the solution in [30], we consider the next combined relaxation.

Definition 5. *Let R be a resource, $\mathcal{E}$ be an MBO, and ϵ be a function mapping distinguishers to a real value in $[0, 1]$. The $(\mathcal{E}, \epsilon)$-until-relaxation of R, denoted $R^{\mathcal{E}:\epsilon}$, is defined as follows:*

$$R^{\mathcal{E}:\epsilon} \coloneqq \left(\left(R^{\mathcal{E}]}\right)^{\epsilon}\right)^{\mathcal{E}]}.$$

The combined relaxation benefits from the following desired properties, as shown in [30].

Lemma 1. *Let $\mathcal{R}$ be a specification, $\mathcal{E}_1$, $\mathcal{E}_2$ be MBOs for the resource, and ϵ_1, ϵ_2 be functions mapping distinguishers to a real value in $[0, 1]$. Then,*

$$(\mathcal{R}^{\mathcal{E}:\epsilon})^{\mathcal{E}':\epsilon'} \subseteq \mathcal{R}^{\mathcal{E}\vee\mathcal{E}':\epsilon_{\mathcal{E}\vee\mathcal{E}'}+\epsilon'_{\mathcal{E}\vee\mathcal{E}'}},$$

where $\epsilon_{\mathcal{E}\vee\mathcal{E}'}(D) = \epsilon(D \circ \mathsf{until}_{\mathcal{E}\vee\mathcal{E}'})$ is the advantage of the distinguisher interacting with the projected (by the function $\mathsf{until}_{\mathcal{E}\vee\mathcal{E}'}(\cdot)$) resource, and analogously for $\epsilon'_{\mathcal{E}\vee\mathcal{E}'}$.

Lemma 2. *Let $\mathcal{R}$ be a specification, $\mathcal{E}$ be an MBO for the resource, and ϵ be a function mapping distinguishers to a real value in $[0, 1]$. Further let α be a converter, and let i be an interface of $\mathcal{R}$. The $(\mathcal{E}, \epsilon)$-until-relaxation is compatible with converter application and with parallel composition. That is,*

1. *$\alpha^i\left(\mathcal{R}^{\mathcal{E}:\epsilon}\right) \subseteq \left(\alpha^i\mathcal{R}\right)^{\mathcal{E}:\epsilon_\alpha}$, for $\epsilon_\alpha(D) \coloneqq \epsilon(D\alpha^i)$, where $D\alpha^i$ denotes the distinguisher that first attaches α at interface i of the given resource, and then executes D.*
2. *$[\mathcal{R}^{\mathcal{E}:\epsilon}, \mathcal{S}] \subseteq [\mathcal{R}, \mathcal{S}]^{\mathcal{E}:\epsilon_{\mathcal{S}}}$, for $\epsilon_{\mathcal{S}}(D) \coloneqq \sup_{S\in\mathcal{S}} \epsilon(D[\cdot, S])$, where $D[\cdot, S]$ denotes the distinguisher that emulates S in parallel to the given resource, and then executes D.*

3 Multi-Party Constructive Cryptography with Adaptive Corruption

In this section, we present our model for n-party constructions with adaptive corruption. In this setting, we consider scenarios where an adversary may "hack" into the parties' systems during the protocol execution.

Multi-Party Resources. A multi-party resource for n parties is a resource with $n + 2$ interfaces: a set $\mathcal{P} = \{1, \ldots, n\}$ of n party interfaces, an adversary interface A and a free interface W [2]. The party interfaces allow each party to have access to the resource. The adversary interface models adversarial access to the resource. The free interface allows direct access by the environment to the resource[1], and is used to model aspects that are not used by the parties, but neither controlled by the adversary.

Examples of such resources are the available network resource (which allows parties to communicate with each other), as well as the constructed computation resources (which allows the parties to perform arbitrary computations of their secret inputs).

[1] This is reminiscent of the environment access to the global setup in UC [11].

Protocols. A *protocol* consists of a tuple of converters $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_n)$, one for each party. We denote by $\boldsymbol{\pi}R$ the resource where each converter π_j is attached to party interface j.

Basic Construction Notion. We say that a protocol $\boldsymbol{\pi}$ constructs specification $\mathcal{S}$ from specification $\mathcal{R}$, if and only if $\boldsymbol{\pi}\mathcal{R} \coloneqq \{\boldsymbol{\pi}R \mid R \in \mathcal{R}\}$ satisfies specification $\mathcal{S}$, i.e., $\boldsymbol{\pi}\mathcal{R} \subseteq \mathcal{S}$.

Definition 6. *Let $\mathcal{R}$ and $\mathcal{S}$ be specifications, and let $\boldsymbol{\pi}$ be a protocol. We say that $\boldsymbol{\pi}$ constructs $\mathcal{S}$ from $\mathcal{R}$, if and only if $\boldsymbol{\pi}\mathcal{R} \subseteq \mathcal{S}$.*

The specifications $\boldsymbol{\pi}\mathcal{R}$ and $\mathcal{S}$ are usually called the real world and the ideal world, respectively.

Typical constructions in the literature describe the ideal specification $\mathcal{S}$ with the so-called simulation-paradigm. That is, by showing the existence of a simulator σ attached to the adversary interface of a fixed ideal resource S, which can for example be a resource that computes a function over private inputs. Note that this is just a particular way of defining the security guarantees in our framework. One can express different types of ideal specifications, as we will show in our examples (see [34, 30] for further examples in other settings).

Protocol Converters as Resources. In order to model adaptive corruptions in Constructive Cryptography, we consider only trivial converters [39]. More concretely, we consider the class Σ of trivial converters which only define a wiring between resource interfaces, and the protocol engines are then interpreted as resources. In more detail, when writing a resource $\alpha^i R$ consisting of a converter α attached to interface i of resource R, we understand the converter α as a resource, for example denoted $\tilde{\alpha}$, in parallel with R. And we consider a trivial converter β for interface i that simply connects $\tilde{\alpha}$ and R, i.e., we have $\alpha^i R = \pi_i^i[R, \tilde{\alpha}]$. We depict in Figure 1 this interpretation.

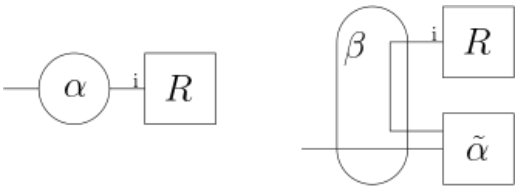

Fig. 1. On the left, the converter α is connected to a resource R. On the right, the interpretation where the converter α is interpreted as a resource $\tilde{\alpha}$ in parallel, with a trivial converter that connects interfaces.

Modeling Corruptions. Protocol converters as resources have, like any resource, an adversary interface A and a free interface W. Corruption is modeled explicitly as an input to the resource via the free interface W. Upon input corrupt at interface W, the resource adds additional capabilities at the adversary interface A.[2]

One can then model different types of corruption. In order to model passive corruption, we require that upon input corrupt at interface W, the resource makes accessible the entire local state at interface A. One can then access the local state of the resource via interface A with an input `leak`. If active corruption is considered, the adversary can in addition take over the inside and outside interfaces of the protocol engine via the adversarial interface A. That is, any inputs given at the inside or outside interface are first made available to A, who then decides what the values are.

4 CC-Adaptive Security

In an MPC protocol, the set of corrupted parties grows during the protocol execution, and at the same time, the set of parties that benefit from security guarantees, i.e. the set of so-far honest parties, shrinks.

We propose a very natural way to understand such guarantees obtained from an MPC protocol with adaptive corruption. The idea is to understand the guarantees as simultaneously achieving guarantees for

[2] One could alternatively model that the input corrupt is given at the adversary interface A, with an additional mechanism to ensure that the real and ideal world corruptions are the same; for example making available the set of currently corrupted parties via the free interface W.

every set of so-far honest parties. More concretely, we explicitly state one separate guarantee for every subset $X \subseteq \mathcal{P}$ of the parties, and as long as parties in X are honest, the guarantee remains. That is, the guarantee is dropped as soon as any party in X gets corrupted.

The corruptions are completely adaptive as usual, and the identity of the chosen parties to become corrupted can be made based on information gathered during the protocol execution. The more parties are corrupted, the less sets are so-far honest, and therefore less guarantees remain.

The described guarantees can naturally be captured within the constructive cryptography framework, where each guarantee corresponds to a resource specification, and the conjunction of guarantees is simply the intersection of specifications.

As we will show in Section 4.2, our notion of CC-adaptive security lies strictly in-between the standard UC-security notions of static and adaptive security. Popular examples that separate static and adaptive security and are not based on the commitment problem, also separate static and CC-adaptive security, and examples based on the commitment problem separate our notion from UC-adaptive security; therefore showing that CC-adaptive security achieves a strong resilience against adaptive corruption, while at the same time overcoming the commitment problem.

4.1 Definition of the Security Notion

As sketched above, our security notion gives a guarantee for every set of so-far honest parties. That is, we give a explicit guarantee for each subset $X \subseteq \mathcal{P}$ of parties, which lasts as long as the subset X is honest, irrespective of whether the other parties are honest or not. The guarantee provides privacy to the set of parties in X, and is described as usual, by requiring the existence of a simulator (for this set X) that correctly simulates the protocol execution. The simulator has to simulate without knowing the secret inputs and outputs of parties in X, but since the guarantee holds irrespective of the honesty of other parties, we allow the simulator to have access to the inputs of parties that are in $\overline{X} = \mathcal{P} \setminus X$. Moreover, as soon as a party in X is corrupted, the guarantee for this set is lost (and therefore the simulation stops at this point). However, guarantees for other so-far honest sets still remain.

Finally, we state the guarantees with respect to a (monotone[3]) *adversary structure* $\mathcal{Z} \subseteq 2^{\mathcal{P}}$, meaning that if too many corruptions happen, i.e., the set of corrupted parties exceeds the adversary structure, all guarantees are lost.

Addressing Adaptive Concerns. We will see with the examples below that this security notion captures the typical adaptive concerns: leaking sensitive information in the network, and also information leaked from an internal state of a party.

The former is prevented since the adversary is fully adaptive and can corrupt parties based on information seen in the network. Note, however, that when considering a small number of parties, and in particular two parties, our definition is intuitively very close to that of static security. This is because when X contains one party, the guarantee holds only for adversarial strategies that involve corrupting the (single) party in $\overline{X}$.[4] (When the number of parties increases, many of the sets $\overline{X}$ contain a large number of parties, and the guarantee holds even when the adversary adaptively corrupts parties among these large sets. See Lemma 5.)

To see why the definition also limits the information leaked from an internal state, note that upon corruption of party i, all guarantees where $i \notin X$, still remain. In all those simulations the internal state of party i must be explained (from the inputs of parties in $\overline{X}$). Concretely, for X being the so-far honest set, the state of party i does not reveal anything beyond what can be inferred from the current corrupted set.

Avoiding the Commitment Problem. Intuitively, the commitment problem does not arise, because the guarantees are lost (i.e. the simulation stops) at the point where a party in X gets corrupted. Therefore, the simulator does not need to explain the secret state of any party in X. Moreover, for parties that are in $\overline{X}$, the simulator can consistently explain the secret state because it has access to the inputs of these parties.

[3] If $Z \in \mathcal{Z}$ and $Z' \subseteq Z$, then $Z' \in \mathcal{Z}$.

[4] However, note that for passive corruption and a small number of parties, static security intuitively provides sufficiently strong guarantees, as the static adversary can always guess upfront which set of parties will be corrupted.

Let $\mathcal{E}_X$ be the MBO indicating whether any party in X is corrupted. Moreover, let $\sigma_{\overline{X}}$ be a simulator that has access to the inputs of all parties from the set $\overline{X}$.[5] Formally, any inputs given at interfaces from parties in $\overline{X}$, are forwarded to the adversary interface. Moreover, note that we only allow the simulator to modify the inputs of actively corrupted parties.[6]

Further let $\mathcal{E}_{\mathcal{Z}}$ be the MBO that is set to 1 when the set of corrupted parties does not lie in $\mathcal{Z}$. For the common case of threshold corruption where the adversary structure contains all sets of up to t parties, we denote $\mathcal{E}_t$ the MBO that is set to 1 when more than t parties are corrupted.

We require that for each set of parties X, there must be a simulator $\sigma_{\overline{X}}$ that simulates the protocol execution until any party in X is corrupted, or the adversary structure is no longer respected, i.e., until $\mathcal{E}_X \vee \mathcal{E}_{\mathcal{Z}} = 1$.

Definition 7. *Protocol $\boldsymbol{\pi}$ CC-adaptively constructs specification $\mathcal{S}$ from $\mathcal{R}$ with error ϵ and adversary structure $\mathcal{Z}$, if for each set $X \subseteq \mathcal{P}$, there exists (an efficient) simulator $\sigma_{\overline{X}}$, such that $\boldsymbol{\pi}\mathcal{R} \subseteq (\sigma_{\overline{X}}\mathcal{S})^{\mathcal{E}_X \vee \mathcal{E}_{\mathcal{Z}}:\epsilon}$. In short, $\boldsymbol{\pi}\mathcal{R}$ satisfies the following intersection of specifications:*

$$\boldsymbol{\pi}\mathcal{R} \subseteq \bigcap_{X \subseteq \mathcal{P}} (\sigma_{\overline{X}}\mathcal{S})^{\mathcal{E}_X \vee \mathcal{E}_{\mathcal{Z}}:\epsilon}$$

Moreover, we say that $\boldsymbol{\pi}$ CC-adaptively constructs $\mathcal{S}$ from $\mathcal{R}$ with error ϵ up to t corruptions if $\boldsymbol{\pi}\mathcal{R} \subseteq \bigcap_{X \subseteq \mathcal{P}} (\sigma_{\overline{X}}\mathcal{S})^{\mathcal{E}_X \vee \mathcal{E}_t:\epsilon}$.

The following lemma shows that this type of construction statement benefits from desirable composition guarantees.

Lemma 3. *Let $\mathcal{R}$, $\mathcal{S}$, $\mathcal{T}$ be specifications, and let $\boldsymbol{\pi}$, $\boldsymbol{\pi}'$ be protocols. Further let $\mathcal{Z} \subseteq 2^{\mathcal{P}}$ be a monotone set. Then, we have the following composition guarantees:*

$$\boldsymbol{\pi}\mathcal{R} \subseteq \bigcap_{X \subseteq \mathcal{P}} (\sigma_{\overline{X}}\mathcal{S})^{\mathcal{E}_X \vee \mathcal{E}_{\mathcal{Z}}:\epsilon} \;\wedge\; \boldsymbol{\pi}'\mathcal{S} \subseteq \bigcap_{X \subseteq \mathcal{P}} (\sigma'_{\overline{X}}\mathcal{T})^{\mathcal{E}_X \vee \mathcal{E}_{\mathcal{Z}}:\epsilon'} \implies \boldsymbol{\pi}'\boldsymbol{\pi}\mathcal{R} \subseteq \bigcap_{X \subseteq \mathcal{P}} (\sigma'_{\overline{X}}\sigma_{\overline{X}}\mathcal{T})^{\mathcal{E}_X \vee \mathcal{E}_{\mathcal{Z}}:\tilde{\epsilon}},$$

for $\tilde{\epsilon} \coloneqq \sup_{X \subseteq \mathcal{P}}\{(\epsilon_{\pi'})_{\mathcal{E}_X \vee \mathcal{E}_{\mathcal{Z}}} + (\epsilon'_{\sigma_{\overline{X}}})_{\mathcal{E}_X \vee \mathcal{E}_{\mathcal{Z}}}\}$, where $(\epsilon_{\pi'})_{\mathcal{E}_X \vee \mathcal{E}_{\mathcal{Z}}}$ is the advantage of the distinguisher that first attaches π' to the given resource, and then interacts with the projected resource, and same for $(\epsilon'_{\sigma_{\overline{X}}})_{\mathcal{E}_X \vee \mathcal{E}_{\mathcal{Z}}}$.

Furthermore, we have

$$\boldsymbol{\pi}\mathcal{R} \subseteq \bigcap_{X \subseteq \mathcal{P}} (\sigma_{\overline{X}}\mathcal{S})^{\mathcal{E}_X \vee \mathcal{E}_{\mathcal{Z}}:\epsilon} \implies \boldsymbol{\pi}[\mathcal{R}, \mathcal{T}] \subseteq \bigcap_{X \subseteq \mathcal{P}} (\sigma_{\overline{X}}[\mathcal{S}, \mathcal{T}])^{\mathcal{E}_X \vee \mathcal{E}_{\mathcal{Z}}:\epsilon_{\mathcal{T}}},$$

for $\epsilon_{\mathcal{T}}(D) \coloneqq \sup_{T \in \mathcal{T}} \epsilon(D[\cdot, T])$, where $D[\cdot, T]$ is the distinguisher that emulates T in parallel to the given resource, and then executes D.

Proof. The proof can be found in Section A.

4.2 Comparison to Traditional Notions of Security

In this section we show how to phrase the standard notions of static and adaptive security within our framework, and further show that our new definition lies in-between the two standard notions of static and adaptive security.

Static Security. In the standard notion of static security, the set of protocol engines that are corrupted is fixed before the computation starts and does not change. The possible corruption sets are modelled by a given adversary structure $\mathcal{Z} \subseteq 2^{\mathcal{P}}$. Given a set $Z \in \mathcal{Z}$, we denote by $\boldsymbol{\pi}_{\overline{Z}}\mathcal{R}$ the real-world resource, where the set of protocol engines π_i, $i \in Z$, are corrupted. The security definition requires the existence of a simulator σ_Z that simulates the protocol execution and has control over the inputs and outputs from corrupted parties. As usual, in the passive case, the simulator can read these values, while in the active case, it can also change them.

[5] Our basic approach of stating a guarantee per set X allows to consider many different definitions, depending on which information is accessible to the simulator. We choose to solely leak the inputs of parties in $\overline{X}$, since this seems to be the minimal necessary information to overcome the commitment problem.

[6] Allowing the simulator to modify the inputs of honest parties in $\overline{X}$ results in an unnecessarily weak notion.

Definition 8. *Protocol $\boldsymbol{\pi}$ statically constructs specification $\mathcal{S}$ from $\mathcal{R}$ with error ϵ and adversary structure $\mathcal{Z}$, if for each possible set of corrupted parties $Z \in \mathcal{Z}$, there exists a simulator σ_Z such that $\boldsymbol{\pi}_{\overline{Z}}\mathcal{R} \subseteq (\sigma_Z\mathcal{S})^\epsilon$, where $\boldsymbol{\pi}_{\overline{Z}}$ indicates that protocol converters π_i, $i \in Z$, are corrupted, and σ_Z indicates that the simulator has control over the inputs and outputs of parties in Z.*

Lemma 4. *CC-adaptive security implies static security.*

Proof. Let $\boldsymbol{\pi}$ be a protocol that constructs $\mathcal{S}$ from specification $\mathcal{R}$ with error ϵ and adversary structure $\mathcal{Z}$, with CC-adaptive security. We prove that $\boldsymbol{\pi}$ also satisfies static security with the same parameters. Fix a set $Z \in \mathcal{Z}$. Consider the particular corruption strategy, where parties in Z are corrupted at the start of the protocol execution, and no more corruptions happen.

In this case, $\mathcal{E}_{\overline{Z}} \vee \mathcal{E}_{\mathcal{Z}} = 0$, because no party in $\overline{Z}$ is corrupted, and the set of corrupted parties lies within $\mathcal{Z}$. Therefore, for the case where $X = \overline{Z}$, there must exist a simulator σ_Z (with access to the inputs and outputs of parties in Z, which are corrupted) that satisfies $\boldsymbol{\pi}_{\overline{Z}}\mathcal{R} \subseteq (\sigma_Z\mathcal{S})^{\mathcal{E}_{\overline{Z}} \vee \mathcal{E}_{\mathcal{Z}} : \epsilon} = (\sigma_Z\mathcal{S})^\epsilon$.

In the following, we show that known examples of protocols that separate the standard notions of static and adaptive security [12, 10], also separate static and CC-adaptive security, both in the case of passive as well as active corruption.

Lemma 5. *For passive corruption and a large number of parties, CC-adaptive security is strictly stronger than static security.*

Proof. We consider the classical example from Canetti et al. [12]. Consider a secure function evaluation protocol with guaranteed output delivery where parties evaluate the function that outputs $\perp$. The adversary structure contains sets of up to $t = O(n)$ parties.

The protocol $\boldsymbol{\pi}$ proceeds as follows: A designated party D secret shares its input to a randomly selected set of parties U (out of all parties except D) of small size κ parties using a κ-out-of-κ sharing scheme, where κ is the security parameter. Then, D makes the set U public (e.g. by sending the set to all parties). Subsequently, all parties output $\perp$.

It is known that $\boldsymbol{\pi}$ achieves static security. This is because an adversary not corrupting D only learns D's secret if U happens to be the predefined set of corrupted parties, which occurs with probability exponentially small in κ. More concretely, for each $Z \in \mathcal{Z}$ not containing D, the probability that $U = Z$ is $\binom{n-1}{\kappa}^{-1} = \mathtt{neg}(\kappa)$. (Note that in the case where $U \neq Z$, the simulator trivially succeeds simply by emulating the shares as random values.)

Now we show that $\boldsymbol{\pi}$ does not achieve CC-adaptive security. Consider the singleton set $X = \{D\}$, containing only the designated party. Note that U does not contain D, since D chooses a set of κ parties randomly from the set of parties without D. The adversary can then corrupt the set of parties in U to find out D's secret without corrupting D. Note that the simulator has access to all inputs from parties in $\overline{X}$, but has no access to D's input. Formally, the simulator $\sigma_{\overline{X}}$ has to output shares for parties in U that add up to D's input, without knowing the input, which is impossible.

Lemma 6. *For active corruption, CC-adaptive security is strictly stronger than static security, as long as there are at least three parties.*

Proof. We consider the example from Canetti et al. [10] with three parties D, R_1 and R_2. D has as input two bits $b_1, b_2 \in \{0, 1\}$, and R_1, R_2 have no input. The ideal resource evaluates the function f that, on input (b_1, b_2), it outputs b_1 to R_1, b_2 to R_2 and $\perp$ to D. The adversary structure contains $\{D, R_1\}$.

The protocol $\boldsymbol{\pi}$ proceeds as follows: at step 1 D sends b_1 to R_1. After that, at step 2 D sends b_2 to R_2. Finally, at step 3 each R_i outputs the bit that they received from D and terminates, and D outputs $\perp$ and terminates.

It was proven that $\boldsymbol{\pi}$ achieves static security: for the set $Z = \{D\}$, the simulator gets the values s'_1 and s'_2 from the adversary interface, and it sends (b'_1, b'_2) to the ideal resource, who forwards each b'_i to R_i. It is easy to see that this simulator perfectly simulates the protocol execution. The case where $Z = \{D, R_1\}$ is similar.

For the set $Z = \{R_1\}$, the simulator obtains the output b_1 from the ideal resource, so it can simply forward this bit to the adversary interface. Again, it is easy to see that the simulation is successful.

Now let us argue why the above protocol does not satisfy CC-adaptive security. To show that, consider the singleton set $X = \{R_2\}$, containing only party R_2. We will show that the adversary can break

correctness of the protocol, by 1) learning the value s_1 that is sent to R_1, and 2) depending on the value received after step 1 from the so-far honest D, possibly corrupt D and modify the value that is sent to R_2 at step 2. More concretely, the adversary strategy is as follows: Initially corrupt R_1, and learn the value s_1 from D. If $s_1 = 1$, then corrupt D and choose the value $s_2' = 0$ as the value that is sent to R_2 at step 2. With this strategy, in the real-world protocol, whenever $s_1 = 1$, R_2 never outputs 1.

Consider the case where the input to D is $(s_1, s_2) = (1, 1)$. As argued above, in the real-world R_2 outputs 0. However, since D is honest at step 1, the simulator $\sigma_{\overline{X}}$ (even with knowledge of the input of D) has no power to change the input of D. Therefore, D inputs $(1, 1)$ to the ideal resource, and therefore the output of party R_2 is 1.

Adaptive Security. In the standard notion of UC-adaptive security, the ideal-world is described as a single specification that consists of a simulator –with passive or active capabilities (i.e. can read, or also change the inputs and outputs of corrupted parties)– attached to the adversary interface of a fixed ideal resource. Moreover, guarantees are given as long as the corrupted parties respect the adversary structure.

Definition 9. *Protocol $\boldsymbol{\pi}$ UC-adaptively constructs specification $\mathcal{S}$ from $\mathcal{R}$ with error ϵ and adversary structure $\mathcal{Z}$, if there is a simulator σ, such that $\boldsymbol{\pi}\mathcal{R} \subseteq (\sigma\mathcal{S})^{\mathcal{E}_{\mathcal{Z}}:\epsilon}$.*

Lemma 7. *UC-adaptive security implies CC-adaptive security.*

Proof. Let $\boldsymbol{\pi}$ be a protocol that constructs $\mathcal{S}$ from specification $\mathcal{R}$ with error ϵ and adversary structure $\mathcal{Z}$, with standard adaptive security. We prove that $\boldsymbol{\pi}$ also satisfies CC-adaptive security. For each set $X \subseteq \mathcal{P}$, we have that there exists a simulator $\sigma_{\overline{X}}$ such that $\boldsymbol{\pi}\mathcal{R} \subseteq (\sigma_{\overline{X}}\mathcal{S})^{\mathcal{E}_{\mathcal{Z}}:\epsilon}$. This is because one can consider the simulator $\sigma_{\overline{X}}$ that ignores the inputs from parties in $\overline{X}$ and simulates according to the UC-adaptive simulator σ. Moreover, we have that

$$(\sigma_{\overline{X}}\mathcal{S})^{\mathcal{E}_{\mathcal{Z}}:\epsilon} \subseteq (\sigma_{\overline{X}}\mathcal{S})^{\mathcal{E}_X \vee \mathcal{E}_{\mathcal{Z}}:\epsilon},$$

because $\mathcal{E}_{\mathcal{Z}} = 1$ implies $\mathcal{E}_X \vee \mathcal{E}_{\mathcal{Z}} = 1$. Therefore, we have the following:

$$\boldsymbol{\pi}\mathcal{R} \subseteq (\sigma\mathcal{S})^{\mathcal{E}_{\mathcal{Z}}:\epsilon} \subseteq \bigcap_{X \subseteq \mathcal{P}} (\sigma_{\overline{X}}\mathcal{S})^{\mathcal{E}_X \vee \mathcal{E}_{\mathcal{Z}}:\epsilon}.$$

Lemma 8. *For passive corruption and any number of parties, CC-adaptive security does not imply UC-adaptive security.*

Proof. Consider a secure function evaluation protocol where parties evaluate the function that outputs $\perp$. The adversary structure contains sets of up to $t = 2$ parties. The protocol $\boldsymbol{\pi}$ proceeds as follows: A designated party D computes a commitment of its private input, using a (non-equivocable) perfectly hiding and computationally binding commitment scheme and makes this commitment public. Then, all parties output $\perp$.

The protocol does not achieve standard adaptive security. Consider the corruption strategy where D is corrupted "after" he sent his commitment. The simulator then first has to come up with a commitment without knowing D's input, and then, upon corruption, learns D's input and has to output randomness consistent with D's input. Since the commitment is non-equivocable, this is not possible. That is, the simulation strategy runs into the commitment problem.

It is easy to see that $\boldsymbol{\pi}$ satisfies CC-adaptive security. This is because for any set X not containing D, the simulator can read D's input, so the simulation is straightforward. On the other hand, for any set X containing D, the simulation is only required until the point in time where D becomes corrupted (without including the answer to the corruption query, i.e., there is no need to output D's private state).

5 Some Ideal Resource Specifications

In this section we introduce some typical ideal specifications that will be used in later sections, such as the network model, broadcast and MPC. We consider the setting with open authenticated and asynchronous channels, where the adversary can choose to drop messages. As a consequence, the ideal building blocks for broadcast and MPC that we consider achieve so-called security with abort.

5.1 Communication Primitives

Network Model. We consider a multi-party communication network with point-to-point asynchronous authenticated channels among any pair of parties, in line with the $\mathcal{F}_{\mathsf{auth}}$ functionality in [9]. Asynchronous means that the channels do not guarantee delivery of the messages, and the messages are not necessarily delivered in the order which they are sent. Authenticity ensures that a recipient will only receive a message from a sender if the sender actually sent the message. In the case of adaptive corruptions, this authenticity requirement holds as long as both sender and recipient are honest. In particular, a dishonest sender is allowed to modify the messages that have been sent, as long as they have not been delivered to the recipient yet. We denote $\mathcal{N}$ the complete network of pairwise authenticated channels (see Section C).

Broadcast with Abort. In our protocols, we assume that parties have access to an authenticated broadcast channel with abort[27], which guarantees that no two honest parties receive different messages, and does not guarantee delivery of messages. We denote the broadcast specification $\mathcal{BC}$ a broadcast channel that allows any party to broadcast. Such a broadcast resource can be constructed with standard adaptive security and arbitrary number of corruptions in the communication network $\mathcal{N}$ using the protocol by Goldwasser and Lindell [27].[7] See Section C for a detailed description.

5.2 MPC with Abort

We briefly describe an ideal resource MPC capturing secure computation with abort (and no fairness). The resource has $n+2$ interfaces, n party interfaces, an adversary interface A and a free interface W. Via the free interface, the resource keeps track of the set of corrupted parties. The resource allows each party i to input a value x_i, and then once all honest parties provided its input, it evaluates a function $y = f(x_1, \ldots, x_n)$ (corrupted parties can change their input as long as the output was not evaluated). The adversary can then select which parties obtain output and which not.

Resource MPC_f

Initialization

1: $x_1, \ldots, x_n \leftarrow \perp$
2: $y, y_1, \ldots, y_n \leftarrow \perp$
3: $C = \varnothing$

Party Interfaces

1: On input $(\texttt{input}, x)$ at interface $i \in [n]$, if $x_i = \perp$, set $x_i = x$. Output $\perp$ at interface i. Moreover, if $x_j \neq \perp$ for each $j \notin C$, then set $y = f(x_1, \ldots, x_n)$.
2: On input `output` at interface $i \in [n]$, output y_i at interface i.

Adversary Interface

1: On input $(\texttt{deliver}, j)$, $j \in [n]$, at interface A, set $y_j = y$. Output $\perp$ at interface A.
2: On input $(\texttt{input}, x, i)$ at interface A, if $i \in C$ and $y = \perp$, then set $x_i = x$. Output $\perp$ at interface A.
3: On input $(\texttt{leak}, j)$, $j \in C$, at interface A, output x_j at interface A.
4: On input `leakOutput`, at interface A, output y at interface A.

Free Interface

1: On input $(\texttt{corrupt}, i)$ at interface W, set $C = C \cup \{i\}$. Output $\perp$ at interface W.

5.3 Multi-Party Zero-Knowledge

Let R be a binary relation, consisting of pairs (x, w), where x is the statement, and w is a witness to the statement. A zero-knowledge proof allows a prover to prove to a verifier knowledge of w such that $R(x, w) = 1$. In our protocols we will consider the *multi-party* version, which allows a designated party i

[7] Note that in broadcast with abort, even when the sender is honest, it is allowed that parties output $\perp$.

to prove a statement towards all parties according to relation R. Such a resource $\mathsf{ZK}_{i,R}$ can be seen as a special instance of MPC with abort MPC_f resource, where the function f simply takes as input (x, w) from the designated party i, and no input from any other party, and outputs x in the case $R(x, w) = 1$, and otherwise $\perp$. We denote ZK_R the parallel composition of $\mathsf{ZK}_{i,R}$, for $i \in [n]$.

Such a resource can be constructed assuming $\mathcal{BC}$ and a CRS even with standard adaptive security for arbitrary many corruptions (see e.g. [13]). Alternatively, the resource can be constructed less efficiently solely from $\mathcal{BC}$ for the case where $t < n/2$ (see e.g. [47] with [32]).

5.4 Oblivious Transfer

An oblivious transfer resource involves a designated sender s, with input $(x_1, \ldots, x_\ell)$, and a designated receiver with input $i \in \{1, \ldots, \ell\}$. The output for the receiver is x_i, and the sender has no output. For our purposes, we can see the resource as a special instance of MPC with abort MPC_f resource, where the function f simply takes as input $(x_1, \ldots, x_\ell)$ from the designated sender s, an input i from the designated receiver r, and no other inputs, and it outputs x_i to the receiver, and no output to any other party.

6 Application to the CDN Protocol

In this section we show that the protocol by Cramer, Damgard and Nielsen [17] based on threshold (additively) homomorphic encryption essentially achieves MPC with abort and with CC-adaptive security, in the communication network $\mathcal{N}$ of authenticated asynchronous channels. With similar techniques, one could achieve MPC with guaranteed output delivery and with CC-adaptive security in a synchronous communication model (assuming a broadcast specification).

The CDN protocol is perhaps the iconic example that suffers from the commitment problem, and the goal of this example is to conceptually distil out at which steps the protocol is subject to relevant adaptive attacks, and conclude that the CDN-approach of broadcasting encrypted inputs in the first step and computing on ciphertexts, actually achieves strong adaptive security guarantees, even when the encryption commits to the plaintext. We showcase the applicability of our definition with two versions of CDN, for passive corruption below, and active corruption in Section D.

Finally, note that the protocol is typically described assuming a synchronous network, where the protocol advances in a round to round basis, and messages send at round r are assumed to arrive by round $r + 1$. However, our assumed network $\mathcal{N}$ is asynchronous. To address this, we follow the standard approach of executing a synchronous protocol over an asynchronous network (see [32]). The idea is simply that each party waits for all round r messages before proceeding to round $r + 1$. The consequence is that the CDN protocol, which achieves full security under a synchronous network, achieves security with abort under an asynchronous network.

6.1 Passive Corruption Case

The protocol relies on an adaptively secure threshold homomorphic encryption scheme (see for example the scheme by Lysyanskaya and Peikert [35], which is based on the Paillier cryptosystem [43]). In such a scheme, given the public key, any party can encrypt a message. However, decrypting the ciphertext requires the collaboration of at least $t + 1$ parties, where t is a parameter of the scheme. The scheme is additively homomorphic in the sense that one can perform additions on ciphertexts without knowing the underlying plaintexts (see Section B).

For a plaintext a, let us denote $\overline{a}$ an encryption of a. Given encryptions $\overline{a}$, $\overline{b}$, one can compute (using homomorphism) an encryption of $a + b$, which we denote $\overline{a} \boxplus \overline{b}$. Similarly, from a constant plaintext α and an encryption $\overline{a}$ one can compute an encryption of αa, which we denote $\alpha \boxdot \overline{a}$. For concreteness, let us assume that the message space of the encryption scheme is the ring $R = \mathbf{Z}_N$, for some RSA modulus N.

Let us first describe a version of the protocol for the passive case (see the section below for a complete description in the active case). The protocol $\Pi_{\mathtt{pcdn}}$ starts by having each party publish encryptions of its input values. Then, parties compute addition and multiplication gates to obtain a common ciphertext, which they jointly decrypt using threshold decryption. Any linear operation (addition or multiplication by a constant) can be performed non-interactively, due to the homomorphism property of the threshold

encryption scheme. Given encryptions $\overline{a}$, $\overline{b}$ of input values to a multiplication gate, parties can compute an encryption of $c = ab$ as follows:

1. Each party i chooses a random $d_i \in \mathbf{Z}_N$ and distribute encryptions $\overline{d_i}$ and $\overline{d_i b}$ to all parties.
2. Parties compute $\overline{a} \boxplus (\boxplus_i \overline{d_i})$ and decrypt it using a threshold decryption.
3. Parties set $\overline{c} = (a + \sum_i d_i) \boxdot \overline{b} \boxminus (\boxplus_i \overline{d_i b})$.

The main problem that arises when dealing with standard adaptive security, even in the passive case, is that of the commitment problem: the simulator has to first output encryptions on behalf of the so-far honest parties during the input stage, and then if one of these honest parties is later corrupted, the simulator learns the real input of this party and must reveal its internal state to the adversary. However, the simulator is now stuck, since the real input is not consistent with the encryption output earlier. To overcome this issue, protocols usually make use of non-committing encryption schemes. An exception to this, is the protocol by Damgard and Nielsen [19], which is a variant of the CDN protocol that even achieves standard adaptive security, and overcomes the commitment problem by assuming a CRS which is programmed in a very clever way.

We show that this issue does not arise when aiming for CC-adaptive security. Technically, for each subset of parties $X \subseteq \mathcal{P}$, the simulator only needs to lie about the inputs of parties in X, since it knows the inputs of the other parties. Moreover, the simulation is only until the point where a party in X gets corrupted, and so we do not need to justify the internal state of this party. We propose CC-adaptive security as a natural security goal to aim for, providing strong security guarantees against adaptive corruption, and at the same time overcoming the commitment problem. Conceptually, this example shows that the CDN-approach achieves such strong adaptive security guarantees, without the need to use non-committing encryption tools or erasures. Note that in the passive case, the protocol assumes solely a setup for threshold homomorphic encryption, whereas the protocol in [19] requires in addition a CRS.

Key Generation. As usual, we model the setup for the threshold encryption scheme with an ideal resource KeyGen that generates its keys. The resource KeyGen simply generates the public key $\mathtt{ek}$ and private key $\mathtt{dk} = (\mathtt{dk}_1, \ldots, \mathtt{dk}_n)$ for the threshold encryption scheme, and outputs to each party i the public key $\mathtt{ek}$ and its private key share $\mathtt{dk}_i$, and to the adversary the public key $\mathtt{ek}$.

Theorem 1. *Protocol $\Pi_{\mathtt{pcdn}}$ CC-adaptively constructs MPC_f from $[\mathcal{N}, \mathsf{KeyGen}]$, with error ϵ and up to $t < n/2$ passive corruptions, where ϵ reduces distinguishers to the corresponding advantage in the security of the threshold encryption scheme and is described in the proof.*

Proof. Let $\mathcal{R} = [\mathcal{N}, \mathsf{KeyGen}]$ and $\mathcal{S} = \{\mathsf{MPC}_f\}$. Fix a set $X \subseteq \mathcal{P}$. We need to show that there is a simulator $\sigma_{\overline{X}}$ such that $\Pi_{\mathtt{pcdn}}\mathcal{R} \subseteq (\sigma_{\overline{X}}\mathcal{S})^{\mathcal{E}_X \vee \mathcal{E}_t : \epsilon}$. At any point in time, if the event $\mathcal{E}_X \vee \mathcal{E}_t = 1$, the simulator halts. The simulator works as follows.

Key Generation. The simulator $\sigma_{\overline{X}}$ simulates this step by invoking the simulator for the threshold encryption scheme. Let $\mathtt{ek}$ denote the public key, and $\mathtt{dk}_i$ denote the decryption key share for party i. It then outputs $\mathtt{ek}$ at the adversary interface.

Network Messages. The simulator simulates each step of the protocol, given that all messages before that step have been delivered by the adversary i.e., the simulator receives all the corresponding `deliver` messages at the adversary interface. If not, it simply keeps waiting. The messages in the steps below are output to the adversary at the corresponding steps, upon receiving the corresponding `leak` messages at the adversary interface.

Input Stage. For each party i that is honest at this step and gave input to the ideal resource, $\sigma_{\overline{X}}$ outputs an encryption c_i on behalf of this party at the adversary interface. If $i \in \mathcal{X}$, then the simulator does not know its input, and computes the ciphertext $c_i = \mathtt{Enc}_{\mathtt{ek}}(0)$ as an encryption of 0. Otherwise, $i \notin \mathcal{X}$ and the simulator knows its input x_i, so it computes $\overline{x_i} = \mathtt{Enc}_{\mathtt{ek}}(x_i)$ as the ciphertext.

For each party i that is corrupted at this point, the simulator knows its input x_i, and forwards this input to the ideal resource.

Addition Gates. This step can be simulated in a straightforward manner, performing local homomorphic operations on behalf of each honest party.

Multiplication Gates. Let $\overline{a}$ and $\overline{b}$ denote the ciphertexts input to the multiplication gate. The simulator $\sigma_{\overline{X}}$ can execute the honest protocol. That is, it generates a random value d_i on behalf of each honest party i, and locally computes $\overline{d_i} = \mathtt{Enc}_{\mathtt{ek}}(d_i)$, and $\overline{d_i b} = d_i \boxdot \overline{b}$. It then outputs the pair of ciphertexts to the adversary interface. For each corrupted party at this step, the simulator obtains the pair of ciphertexts $\overline{d_i}$ and $\overline{d_i b}$.

Upon receiving the pairs of ciphertexts from all parties, compute the ciphertext $\overline{a} \boxplus (\boxplus_i d_i)$, and simulate an honest threshold decryption protocol of this ciphertext. That is, the simulator outputs a decryption share of the ciphertext to the adversary interface. Upon computing $t+1$ decryption shares, reconstruct the plaintext. Let $a + \sum_i d_i$ be the reconstructed plaintext. Then, compute the ciphertext $\overline{c} = (a + \sum_i d_i) \boxdot \overline{b} \boxminus (\boxplus_i \overline{d_i b})$.

Output. The simulator inputs to the ideal resource $(\mathtt{deliver}, j)$, for each party j that obtains an output in the protocol. It also obtains the output with the instruction `leakOutput`. Then, upon obtaining an output y from the ideal resource, use the simulator of the (adaptively secure) threshold decryption protocol to compute decryption shares on behalf of the honest parties (see [35], where one can simply choose as the inconsistent party one of the parties in X).

Corruptions. On input a command `leak`, at interface $A.i$, if i is corrupted, the simulator outputs the internal state of party i. Note that this is easily done since for parties not in X, the simulator has access to its input. And if any party in X gets corrupted, the corresponding MBO is triggered, $\mathcal{E}_X = 1$, and the simulator halts.

We now prove that $\Pi_{\mathtt{pcdn}}\mathcal{R} \subseteq (\sigma_{\overline{X}}\mathcal{S})^{\mathcal{E}_X \vee \mathcal{E}_t : \epsilon}$, for the simulator $\sigma_{\overline{X}}$ described above. For that, we first describe a sequence of hybrids.

Hybrid H_1. In this system, we assume that the simulator has access to all inputs from the parties. It then executes the real-world protocol, except that the key generation and the decryption are executed using the respective simulators for the threshold encryption scheme. By security of the threshold encryption scheme, we have that $\mathsf{until}_{\mathcal{E}_X \vee \mathcal{E}_t}\left(\Pi_{\mathtt{pcdn}}\mathcal{R}\right)$ is ϵ_1-close to $\mathsf{until}_{\mathcal{E}_X \vee \mathcal{E}_t}(H_1)$. That is:

$$\mathsf{until}_{\mathcal{E}_X \vee \mathcal{E}_t}\left(\Pi_{\mathtt{pcdn}}\mathcal{R}\right) \subseteq \left(\mathsf{until}_{\mathcal{E}_X \vee \mathcal{E}_t}(H_1)\right)^{\epsilon_1},$$

where ϵ_1 is the advantage of the distinguisher (modified by the reduction) to the security of the threshold encryption scheme. Moreover, by definition we have that $\mathsf{until}_{\mathcal{E}_X \vee \mathcal{E}_t}(H_1) \in H_1^{\mathcal{E}_X \vee \mathcal{E}_t]}$. Therefore:

$$\mathsf{until}_{\mathcal{E}_X \vee \mathcal{E}_t}\left(\Pi_{\mathtt{pcdn}}\mathcal{R}\right) \subseteq \left(H_1^{\mathcal{E}_X \vee \mathcal{E}_t]}\right)^{\epsilon_1} \iff \Pi_{\mathtt{pcdn}}\mathcal{R} \subseteq H_1^{\mathcal{E}_X \vee \mathcal{E}_t : \epsilon_1}.$$

Hybrid H_2. The simulator in addition sets the input encryption of the honest parties in X at the Input Stage to an encryption of 0. By semantic security of the threshold encryption scheme, and following the same reasoning as above, we have that $H_1 \in H_2^{\mathcal{E}_X \vee \mathcal{E}_t : \epsilon_2}$, where ϵ_2 is the advantage of the distinguisher (modified by the reduction) to the semantic security of the encryption scheme. Moreover, the hybrid specification $\{H_2\}$ corresponds exactly to the ideal specification $(\sigma_{\overline{X}}\mathcal{S})$.

Combining the above steps, we have that $\Pi_{\mathtt{pcdn}}\mathcal{R} \subseteq (\sigma_{\overline{X}}\mathcal{S})^{\mathcal{E}_X \vee \mathcal{E}_t : \epsilon_X}$, where $\epsilon_X = \epsilon_1 + \epsilon_2$. Therefore, choosing the function ϵ where $\epsilon(D) = \sup_{X \subseteq \mathcal{P}}\{\epsilon_X(D)\}$, the statement follows. □

7 Application to the CLOS Protocol

In this section, we show another application of our new definition, with the iconic CLOS protocol [13], which is based on the classical GMW protocol [25].

We show that the CLOS protocol can be used to achieve a CC-adaptively secure protocol for arbitrary number of active corruptions, assuming a CRS resource, and the existence of enhanced trapdoor permutations. Note that, since CC-adaptivity implies static security, and some form of setup is required even for static security, then it is impossible to achieve CC-adaptivity in the plain model (where only the network is assumed) for the dishonest majority setting. However, note that in contrast to the UC-adaptive version of the CLOS protocol, the construction does not require the use of augmented non-committing encryption. In fact, to the best of our knowledge, all UC-adaptively secure protocols in the dishonest majority setting require some form of non-committing encryption.

Theorem 2. *Assume that enhanced trapdoor permutations exist. Then, there exists a non-trivial*[8] *protocol that CC-adaptively constructs* MPC_f *from* $[\mathcal{N}, \mathsf{CRS}]$*, for appropriate error* ϵ *(as defined by the steps below) and for any number of active corruptions.*

We only sketch the proof of the above theorem. We follow the steps of the CLOS protocol. First, a construction of a passively secure protocol assuming the asynchronous communication network $\mathcal{N}$ is shown. This construction is achieved by first constructing an ideal oblivious transfer (OT), and then designing a secure computation protocol assuming an ideal OT. The following lemma shows that the protocol $\Pi_{\mathtt{ot}}$ of [25] achieves CC-adaptive security. We describe the protocol and the proof of the following lemma in Section E.

Lemma 9. *Assume that enhanced trapdoor permutations exist. Then,* $\Pi_{\mathtt{ot}}$ *CC-adaptively constructs* OT *from* $\mathcal{N}$*, for error* ϵ *(described in the proof), and for any number of passive corruptions.*

Given that UC-adaptive security implies CC-adaptive security by Lemma 7, and there is a UC-adaptively secure MPC protocol assuming an ideal OT resource [13], we have the following lemma:

Lemma 10. *There exists a non-trivial protocol that CC-adaptively constructs* MPC_f *from* $[\mathcal{N}, \mathsf{OT}]$*, with no error, and for any number of passive corruptions.*

As a corollary of the above two lemmas and the composition guarantees from Lemma 3, we have:

Corollary 1. *Assume that enhanced trapdoor permutations exist. There exists a non-trivial protocol that CC-adaptively constructs* MPC_f *from* $\mathcal{N}$*, for error* ϵ *(defined by the composition Lemma 3 and error from Lemma 9), and for any number of passive corruptions.*

Second, we use the CLOS compiler that transforms any passively secure protocol operating in the network $\mathcal{N}$, to an actively secure protocol assuming in addition an ideal *commit-and-prove* CP resource (see Section F for a description). One can see that the compiler preserves the adaptivity type in the sense that if the passive protocol is CC-adaptively secure, the compiled protocol is CC-adaptively secure.

Corollary 2. *Let* Π *be a multi-party protocol and let* Π' *be the protocol obtained by applying the CLOS compiler. Then, the following holds: if* Π *CC-adaptively constructs* MPC_f *from* $\mathcal{N}$ *for error* ϵ *and any number of passive corruptions, then* Π' *CC-adaptively constructs* MPC_f *from* $[\mathcal{N}, \mathsf{CP}]$ *for error* ϵ' *defined in the proof and any number of active corruptions.*

Proof (Sketch). The proof in CLOS (Proposition 9.6) shows that a malicious adversary cannot cheat in the compiled protocol because the resource CP checks the validity of each input received. In particular, they show that for any adversary interacting in the compiled protocol Π', there is a passive adversary interacting in protocol Π that simulates the same view.

More precisely, the proof shows that the specification $\Pi\tau\mathcal{N}$ and $\Pi'[\mathcal{N}, \mathsf{CP}]$ are the same, where τ is the translation converter attached at the adversary interface, which translates from the adversary in Π' to the adversary in Π. (Typically the translation is called a simulator, and it happens between a real resource and an ideal resource. Here, the translation is between two real resources.)

Fix a set $X \subseteq \mathcal{P}$. Since Π CC-adaptively constructs MPC_f from $\mathcal{N}$ for error ϵ and any number of passive corruptions, we have that $\Pi\mathcal{N} \subseteq (\sigma_{\overline{X}}\mathsf{MPC}_f)^{\mathcal{E}_X:\epsilon}$.

Using Lemma 2, this implies the desired result:

$$\Pi'[\mathcal{N}, \mathsf{CP}] = \Pi\tau\mathcal{N} \subseteq \tau(\sigma_{\overline{X}}\mathsf{MPC}_f)^{\mathcal{E}_X:\epsilon} \subseteq (\tau\sigma_{\overline{X}}\mathsf{MPC}_f)^{\mathcal{E}_X:\epsilon'} \coloneqq (\sigma'_{\overline{X}}\mathsf{MPC}_f)^{\mathcal{E}_X:\epsilon'},$$

where $\epsilon' = \epsilon_\tau$. □

It was also shown in [13] that CP can be constructed with UC-adaptive security assuming a zero-knowledge resource ZK and broadcast BC. Given that ZK can be constructed assuming a resource CRS and broadcast BC, and BC can be constructed from $\mathcal{N}$, the authors conclude that CP can be constructed from CRS and $\mathcal{N}$. Therefore, since UC-adaptive security implies CC-adaptive security, Lemma 7 shows:

[8] The ideal specification does not require any of the simulators to deliver the messages to the parties. This implies that a protocol that "hangs" (i.e., never sends any messages and never generates output) securely realizes any ideal resource, which is uninteresting. Following [13], we therefore let a non-trivial protocol be one for which all parties generate output if the adversary delivers all messages and all parties are honest.

Corollary 3. *There exists a non-trivial protocol that CC-adaptively constructs* CP *from* $[\mathcal{N}, \mathsf{CRS}]$*, for error* ϵ *(as in [13]), and for any number of active corruptions.*

From Corollaries 1, 2 and 3, and the composition Lemma 3 we achieve the theorem statement.

References

[1] Michael Backes, Markus Dürmuth, Dennis Hofheinz, and Ralf Küsters. Conditional reactive simulatability. In *European Symposium on Research in Computer Security*, pages 424–443. Springer, 2006.

[2] Christian Badertscher, Ueli Maurer, and Björn Tackmann. On composable security for digital signatures. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 494–523. Springer, Heidelberg, March 2018.

[3] Donald Beaver. Secure multiparty protocols and zero-knowledge proof systems tolerating a faulty minority. *Journal of Cryptology*, 4(2):75–122, January 1991.

[4] Donald Beaver and Stuart Haber. Cryptographic protocols provably secure against dynamic adversaries. In Rainer A. Rueppel, editor, *EUROCRYPT'92*, volume 658 of *LNCS*, pages 307–323. Springer, Heidelberg, May 1993.

[5] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th ACM STOC*, pages 1–10. ACM Press, May 1988.

[6] Fabrice Benhamouda, Huijia Lin, Antigoni Polychroniadou, and Muthuramakrishnan Venkitasubramaniam. Two-round adaptively secure multiparty computation from standard assumptions. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part I*, volume 11239 of *LNCS*, pages 175–205. Springer, Heidelberg, November 2018.

[7] Brandon Broadnax, Nico Döttling, Gunnar Hartung, Jörn Müller-Quade, and Matthias Nagel. Concurrently composable security with shielded super-polynomial simulators. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 351–381. Springer, Heidelberg, April / May 2017.

[8] Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, January 2000.

[9] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001.

[10] Ran Canetti, Ivan Damgård, Stefan Dziembowski, Yuval Ishai, and Tal Malkin. On adaptive vs. non-adaptive security of multiparty protocols. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 262–279. Springer, Heidelberg, May 2001.

[11] Ran Canetti, Yevgeniy Dodis, Rafael Pass, and Shabsi Walfish. Universally composable security with global setup. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 61–85. Springer, Heidelberg, February 2007.

[12] Ran Canetti, Uriel Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *28th ACM STOC*, pages 639–648. ACM Press, May 1996.

[13] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th ACM STOC*, pages 494–503. ACM Press, May 2002.

[14] Ran Canetti, Oxana Poburinnaya, and Muthuramakrishnan Venkitasubramaniam. Equivocating yao: constant-round adaptively secure multiparty computation in the plain model. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *49th ACM STOC*, pages 497–509. ACM Press, June 2017.

[15] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *20th ACM STOC*, pages 11–19. ACM Press, May 1988.

[16] Ran Cohen, abhi shelat, and Daniel Wichs. Adaptively secure MPC with sublinear communication complexity. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 30–60. Springer, Heidelberg, August 2019.

[17] Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. Multiparty computation from threshold homomorphic encryption. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 280–299. Springer, Heidelberg, May 2001.

[18] Dana Dachman-Soled, Jonathan Katz, and Vanishree Rao. Adaptively secure, universally composable, multiparty computation in constant rounds. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 586–613. Springer, Heidelberg, March 2015.

[19] Ivan Damgård and Jesper Buus Nielsen. Universally composable efficient multiparty computation from threshold homomorphic encryption. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 247–264. Springer, Heidelberg, August 2003.

[20] Anupam Datta, Ralf Küsters, John C. Mitchell, and Ajith Ramanathan. On the relationships between notions of simulation-based security. In Joe Kilian, editor, *TCC 2005*, volume 3378 of *LNCS*, pages 476–494. Springer, Heidelberg, February 2005.

[21] Yevgeniy Dodis and Silvio Micali. Parallel reducibility for information-theoretically secure computation. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 74–92. Springer, Heidelberg, August 2000.

[22] Juan A. Garay, Daniel Wichs, and Hong-Sheng Zhou. Somewhat non-committing encryption and efficient adaptively secure oblivious transfer. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 505–523. Springer, Heidelberg, August 2009.

[23] Sanjam Garg and Antigoni Polychroniadou. Two-round adaptively secure MPC from indistinguishability obfuscation. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 614–637. Springer, Heidelberg, March 2015.

[24] Sanjam Garg and Amit Sahai. Adaptively secure multi-party computation with dishonest majority. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 105–123. Springer, Heidelberg, August 2012.

[25] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987.

[26] Shafi Goldwasser and Leonid A. Levin. Fair computation of general functions in presence of immoral majority. In Alfred J. Menezes and Scott A. Vanstone, editors, *CRYPTO'90*, volume 537 of *LNCS*, pages 77–93. Springer, Heidelberg, August 1991.

[27] Shafi Goldwasser and Yehuda Lindell. Secure computation without agreement. In *Proceedings of the 16th International Conference on Distributed Computing*, DISC '02, page 17–32, Berlin, Heidelberg, 2002. Springer-Verlag.

[28] Martin Hirt and Vassilis Zikas. Adaptively secure broadcast. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 466–485. Springer, Heidelberg, May / June 2010.

[29] Dennis Hofheinz, Dominique Unruh, and Jörn Müller-Quade. Polynomial runtime and composability. *Journal of Cryptology*, 26(3):375–441, July 2013.

[30] Daniel Jost and Ueli Maurer. Overcoming impossibility results in composable security using interval-wise guarantees. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 33–62. Springer, Heidelberg, August 2020.

[31] Jonathan Katz, Ueli Maurer, Björn Tackmann, and Vassilis Zikas. Universally composable synchronous computation. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 477–498. Springer, Heidelberg, March 2013.

[32] Eyal Kushilevitz, Yehuda Lindell, and Tal Rabin. Information-theoretically secure protocols and security under composition. In Jon M. Kleinberg, editor, *38th ACM STOC*, pages 109–118. ACM Press, May 2006.

[33] Ralf Küsters and Max Tuengerthal. The iitm model: a simple and expressive model for universal composability. *IACR Cryptology EPrint Archive*, 2013:25, 2013.

[34] Chen-Da Liu-Zhang and Ueli Maurer. Synchronous constructive cryptography. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part II*, volume 12551 of *LNCS*, pages 439–472. Springer, Heidelberg, November 2020.

[35] Anna Lysyanskaya and Chris Peikert. Adaptive security in the threshold setting: From cryptosystems to signature schemes. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 331–350. Springer, Heidelberg, December 2001.

[36] Ueli Maurer. Indistinguishability of random systems. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 110–132. Springer, 2002.

[37] Ueli Maurer. Constructive cryptography - a new paradigm for security definitions and proofs. In *In TOSCA*, pages 33,56, 2011.

[38] Ueli Maurer and Renato Renner. Abstract cryptography. In *In Innovations in Computer Science*. Citeseer, 2011.

[39] Ueli Maurer and Renato Renner. From indifferentiability to constructive cryptography (and back). In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part I*, volume 9985 of *LNCS*, pages 3–24. Springer, Heidelberg, October / November 2016.

[40] Ueli M. Maurer, Krzysztof Pietrzak, and Renato Renner. Indistinguishability amplification. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 130–149. Springer, Heidelberg, August 2007.

[41] Silvio Micali and Phillip Rogaway. Secure computation (abstract). In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 392–404. Springer, Heidelberg, August 1992.

[42] Daniele Micciancio and Stefano Tessaro. An equational approach to secure multi-party computation. In Robert D. Kleinberg, editor, *ITCS 2013*, pages 355–372. ACM, January 2013.

[43] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 223–238. Springer, Heidelberg, May 1999.

[44] Rafael Pass. Simulation in quasi-polynomial time, and its application to protocol composition. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 160–176. Springer, Heidelberg, May 2003.
[45] Birgit Pfitzmann and Michael Waidner. *Composition and integrity preservation of secure reactive systems*. IBM Thomas J. Watson Research Division, 2000.
[46] Manoj Prabhakaran and Amit Sahai. New notions of security: Achieving universal composability without trusted setup. In László Babai, editor, *36th ACM STOC*, pages 242–251. ACM Press, June 2004.
[47] Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *21st ACM STOC*, pages 73–85. ACM Press, May 1989.
[48] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd FOCS*, pages 160–164. IEEE Computer Society Press, November 1982.
[49] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986.

Supplementary Material

A Proof of Lemma 3

The proof of the lemma follows from the properties of the ϵ-relaxation and the until-relaxation, and is in line with the composition theorem for interval-wise relaxations in [30].

We start from the first property, which shows sequential composition. That is, if $\boldsymbol{\pi}$ constructs $\mathcal{S}$ from $\mathcal{R}$ with error ϵ, and $\boldsymbol{\pi'}$ constructs $\mathcal{T}$ from $\mathcal{S}$ with error ϵ', then one can construct $\mathcal{T}$ from $\mathcal{R}$ with a new error $\tilde{\epsilon}$, corresponding essentially to the sum of ϵ and ϵ'.

$$\boldsymbol{\pi}\mathcal{R} \subseteq \bigcap_{X\subseteq\mathcal{P}} (\sigma_{\overline{X}}\mathcal{S})^{\mathcal{E}_X\vee\mathcal{E}_\mathcal{Z}:\epsilon} \;\wedge\; \boldsymbol{\pi'}\mathcal{S} \subseteq \bigcap_{X\subseteq\mathcal{P}} (\sigma'_{\overline{X}}\mathcal{T})^{\mathcal{E}_X\vee\mathcal{E}_\mathcal{Z}:\epsilon'} \implies \boldsymbol{\pi'}\boldsymbol{\pi}\mathcal{R} \subseteq \bigcap_{X\subseteq\mathcal{P}} (\sigma'_{\overline{X}}\sigma_{\overline{X}}\mathcal{T})^{\mathcal{E}_X\vee\mathcal{E}_\mathcal{Z}:\tilde{\epsilon}},$$

for $\tilde{\epsilon} \coloneqq \sup_{X\subseteq\mathcal{P}}\{(\epsilon_{\pi'})_{\mathcal{E}_X\vee\mathcal{E}_\mathcal{Z}} + (\epsilon'_{\sigma_{\overline{X}}})_{\mathcal{E}_X\vee\mathcal{E}_\mathcal{Z}}\}$, where $(\epsilon_{\pi'})_{\mathcal{E}_X\vee\mathcal{E}_\mathcal{Z}}$ is the advantage of the distinguisher that first attaches π' to the given resource, and then interacts with the projected resource, and analogously for $(\epsilon'_{\sigma_{\overline{X}}})_{\mathcal{E}_X\vee\mathcal{E}_\mathcal{Z}}$.

Let $X \subseteq \mathcal{P}$ be a set. From the first part, Lemma 2 and composition order invariance, we have:

$$\boldsymbol{\pi'}\boldsymbol{\pi}\mathcal{R} \subseteq \boldsymbol{\pi'}\left((\sigma_{\overline{X}}\mathcal{S})^{\mathcal{E}_X\vee\mathcal{E}_\mathcal{Z}:\epsilon}\right) \subseteq \left((\boldsymbol{\pi'}\sigma_{\overline{X}}\mathcal{S})^{\mathcal{E}_X\vee\mathcal{E}_\mathcal{Z}:\epsilon_{\boldsymbol{\pi'}}}\right) \subseteq \left((\sigma_{\overline{X}}\boldsymbol{\pi'}\mathcal{S})^{\mathcal{E}_X\vee\mathcal{E}_\mathcal{Z}:\epsilon_{\boldsymbol{\pi'}}}\right).$$

Moreover, from the second part we have:

$$(\sigma_{\overline{X}}\boldsymbol{\pi'}\mathcal{S}) \subseteq \sigma_{\overline{X}}\left((\sigma'_{\overline{X}}\mathcal{T})^{\mathcal{E}_X\vee\mathcal{E}_\mathcal{Z}:\epsilon'}\right) \subseteq (\sigma_{\overline{X}}\sigma'_{\overline{X}}\mathcal{T})^{\mathcal{E}_X\vee\mathcal{E}_\mathcal{Z}:\epsilon'_{\sigma_{\overline{X}}}}.$$

Combining both statements and using Lemma 1 yields:

$$\boldsymbol{\pi'}\boldsymbol{\pi}\mathcal{R} \subseteq \left((\sigma_{\overline{X}}\sigma'_{\overline{X}}\mathcal{T})^{\mathcal{E}_X\vee\mathcal{E}_\mathcal{Z}:\epsilon'_{\sigma_{\overline{X}}}}\right)^{\mathcal{E}_X\vee\mathcal{E}_\mathcal{Z}:\epsilon_{\boldsymbol{\pi'}}} \subseteq (\sigma_{\overline{X}}\sigma'_{\overline{X}}\mathcal{T})^{\mathcal{E}_X\vee\mathcal{E}_\mathcal{Z}:\tilde{\epsilon}}.$$

The second property ensures that the construction notion achieves parallel composition. That is, if $\boldsymbol{\pi}$ constructs $\mathcal{S}$ from $\mathcal{R}$, then it also constructs $[\mathcal{S},\mathcal{T}]$ from $[\mathcal{R},\mathcal{T}]$.

$$\boldsymbol{\pi}\mathcal{R} \subseteq \bigcap_{X\subseteq\mathcal{P}} (\sigma_{\overline{X}}\mathcal{S})^{\mathcal{E}_X\vee\mathcal{E}_\mathcal{Z}:\epsilon} \implies \boldsymbol{\pi}[\mathcal{R},\mathcal{T}] \subseteq \bigcap_{X\subseteq\mathcal{P}} (\sigma_{\overline{X}}[\mathcal{S},\mathcal{T}])^{\mathcal{E}_X\vee\mathcal{E}_\mathcal{Z}:\epsilon_\mathcal{T}},$$

for $\epsilon_\mathcal{T}(D) \coloneqq \sup_{T\in\mathcal{T}} \epsilon(D[\cdot,T])$, where $D[\cdot,T]$ denotes the distinguisher that emulates T in parallel to the given resource, and then executes D.

Let $X \subseteq \mathcal{P}$ be a set. From composition order invariance and Lemma 2, we have:

$$\boldsymbol{\pi}[\mathcal{R},\mathcal{T}] = [\boldsymbol{\pi}\mathcal{R},\mathcal{T}] \subseteq [(\sigma_{\overline{X}}\mathcal{S})^{\mathcal{E}_X\vee\mathcal{E}_\mathcal{Z}:\epsilon},\mathcal{T}] \subseteq [\sigma_{\overline{X}}\mathcal{S},\mathcal{T}]^{\mathcal{E}_X\vee\mathcal{E}_\mathcal{Z}:\epsilon_\mathcal{T}} = \left(\sigma_{\overline{X}}[\mathcal{S},\mathcal{T}]\right)^{\mathcal{E}_X\vee\mathcal{E}_\mathcal{Z}:\epsilon_\mathcal{T}}.$$

B Threshold Homomorphic Encryption

We recall the definition of a threshold encryption scheme. A threshold encryption scheme with standard adaptive security can be found in [35], based on the Paillier cryptosystem.

Definition 10. *A homomorphic threshold encryption scheme consists of five algorithms:*

- *(Key generation) The key generation algorithm is parameterized by (t, n) and outputs $(\mathtt{ek}, \mathtt{dk}) = \mathtt{Keygen}_{(t,n)}(1^\kappa)$, where $\mathtt{ek}$ is the public key, and $\mathtt{dk} = (\mathtt{dk}_1, \ldots, \mathtt{dk}_n)$ is the list of secret keys.*
- *(Encryption) There is an algorithm $\mathtt{Enc}$, which on input public key $\mathtt{ek}$ and plaintext m, it outputs an encryption $\overline{m} = \mathtt{Enc}_{\mathtt{ek}}(m; r)$ of m, with random input r.*
- *(Partial decryption) There is an algorithm that, given as input a decryption key $\mathtt{dk}_i$ and a ciphertext, it outputs $d_i = \mathtt{DecShare}_{\mathtt{dk}_i}(c)$, a decryption share.*
- *(Reconstruction) Given $t+1$ decryption shares $\{d_i\}$, one can reconstruct the plaintext $m = \mathtt{Rec}(\{d_i\})$.*
- *(Additively Homomorphic) There is an algorithm which, given public key $\mathtt{ek}$ and encryptions $\overline{a}$ and $\overline{b}$, it outputs a uniquely-determined encryption $\overline{a+b}$. We write $\overline{a+b} = \overline{a} \boxplus \overline{b}$. Likewise, there is an algorithm performing substraction: $\overline{a-b} = \overline{a} \boxminus \overline{b}$.*
- *(Multiplication by constant) There is an algorithm, which, given public key $\mathtt{ek}$, a plaintext a and a ciphertext $\overline{b}$, it outputs a uniquely-determined encryption $\overline{a \cdot b}$. We write $\overline{a \cdot b} = a \boxdot \overline{b}$.*

C Description of Communication Primitives

C.1 Network Model

We first describe a single-message authenticated channel $\mathsf{AUTH}_{i,j}$ from party i to party j. A multi-message authenticated channel is then accordingly obtained via parallel composition of single-message resources. The resource has $n + 2$ interfaces, n party interfaces, an adversary interface A and a free interface W.

The channel expects an input message m at interface i, which is stored upon receipt. The adversary can learn the message that is input, and can choose to deliver the message by making it available at interface j. Moreover, if party i is corrupted, it can inject a new message, as long as the message has not been delivered yet.

Resource $\mathsf{AUTH}_{i,j}$

Initialization

1: $m_i, m_j \leftarrow \perp$
2: $\mathtt{CorruptSender} = 0$

Party Interfaces

1: On input $(\mathtt{send}, m)$ at interface i, if $m_i = \perp$, set $m_i = m$. Output $\perp$ at interface i.
2: On input $\mathtt{receive}$ at interface j, output m_j at interface j.
3: On any input at interface $k \in [n] \setminus \{i, j\}$, output $\perp$ at the same interface.

Adversary Interface

1: On input $\mathtt{leak}$ at interface A, output m_i at interface A.
2: On input $\mathtt{deliver}$ at interface A, set $m_j = m_i$. Output $\perp$ at interface A.
3: On input $(\mathtt{inject}, m)$ at interface A, if $\mathtt{CorruptSender} = 1$ and $m_j \neq \perp$, set $m_i = m$.

Free Interface

1: On input $(\mathtt{corrupt}, i)$ at interface W, set $\mathtt{CorruptSender} = 1$. Output $\perp$ at interface W.

Let $\mathcal{N}$ be the complete network of pairwise authenticated channels, i.e., the parallel composition of $\mathsf{AUTH}_{i,j}$, for $i, j \in [n]$.

C.2 Broadcast with Abort

The resource has $n + 2$ interfaces, n party interfaces, an adversary interface A and a free interface W. As a first step, we model a broadcast resource BC_i where party i is the sender. We then define the broadcast specification $\mathcal{BC}$ that allows any party to broadcast as the specification containing the parallel composition of broadcast resources BC_i, for each $i \in [n]$.

The broadcast specification $\mathcal{BC}$ we consider corresponds to that of *broadcast with abort* [27], and does not guarantee delivery of messages. It only guarantees that no two uncorrupted parties will receive two different messages.

As pointed out by Hirt and Zikas [28], traditional broadcast protocols do not construct the stronger broadcast functionality which simply forwards the sender's message to all parties, since they are subject to adaptive attacks, where the adversary can corrupt an initially honest sender depending on the broadcasted message, and change the broadcasted message. Therefore, we consider the "relaxed" version, which allows an adaptively corrupted sender to change the message sent, as long as the message was not delivered to any party.

Resource BC_i

Initialization

1: $m^* = \perp$
2: $m_1, \ldots, m_n \leftarrow \perp$
3: $\texttt{CorruptSender} = 0$

Party Interfaces

1: On input $(\texttt{bc}, m)$ at interface i, if $m^* = \perp$, set $m^* = m$. Output $\perp$ at interface i.
2: On input `receive` at interface $j \in [n]$, output m_j at interface j.

Adversary Interface

1: On input `leak` at interface A, output m^* at interface A.
2: On input $(\texttt{deliver}, j)$, $j \in [n]$, at interface A, set $m_j = m^*$. Output $\perp$ at interface A.
3: On input $(\texttt{inject}, m)$ at interface A, if $\texttt{CorruptSender} = 1$ and $m_j = \perp$ for all $j \in [n]$, set $m^* = m$.

Free Interface

1: On input $(\texttt{corrupt}, i)$ at interface W, set $\texttt{CorruptSender} = 1$. Output $\perp$ at interface W.

D CDN Protocol: Active Corruption Case

To handle the case of active corruption, the CDN protocol makes use of a broadcast primitive to ensure consistency of distributed messages among the parties, and also zero-knowledge proofs at the appropriate steps of the protocol. As a side remark, we note that the original CDN protocol used a multi-party zero-knowledge proof based on Σ-protocols, where the challenge is chosen by a small randomly-selected committee. This step is subject to crucial adaptive attacks, since the adversary can trivially wait until the committee is selected and corrupt all members. With this step, the protocol would not satisfy CC-adaptive security. However, we will show that the protocol, assuming an ideal multi-party zero-knowledge resource (with abort) does achieve CC-adaptive security. Note that this resource guarantees consistency on whether the designated prover succeeded in the proof, and therefore also can be used as a broadcast resource. Therefore, in the protocol we do not need to assume the broadcast specification $\mathcal{BC}$ or the network $\mathcal{N}$ (these are used to construct $\mathcal{ZK}$). As noted in Section 5.3, such a resource may be instantiated even with standard adaptive security from $\mathcal{N}$ for $t < n/2$, or more efficiently from bilateral zero-knowledge proofs, assuming a CRS.

Multi-Party Zero-Knowledge Relations. We state the protocol assuming ZK_R resources for appropriate relations. More concretely, we are interested in zero-knowledge proofs for three types of relations, parameterized by a threshold encryption scheme with public key `ek`:

1. *Proof of Plaintext Knowledge:* The statement consists of $\mathtt{ek}$, and a ciphertext c. The witness consists of a plaintext m and randomness r such that $c = \mathtt{Enc}_{\mathtt{ek}}(m; r)$.
2. *Proof of Correct Multiplication:* The statement consists of $\mathtt{ek}$, and ciphertexts c_1, c_2 and c_3. The witness consists of a plaintext m_1 and randomness r_1, r_3 such that $c_1 = \mathtt{Enc}_{\mathtt{ek}}(m_1; r_1)$ and $c_3 = m_1 \cdot c_2 + \mathtt{Enc}_{\mathtt{ek}}(0; r_3)$.
3. *Proof of Correct Decryption:* The statement consists of $\mathtt{ek}$, a ciphertext c, and a decryption share d. The witness consists of a decryption key share $\mathtt{dk}_i$, such that $d = \mathtt{Dec}_{\mathtt{dk}_i}(c)$.

Let us denote $\mathcal{ZK}$ the specification containing the parallel composition of $\mathsf{ZK}_{\mathtt{popk}}$, $\mathsf{ZK}_{\mathtt{pocm}}$ and $\mathsf{ZK}\mathtt{pocd}$.

Protocol Description. We describe the protocol engine formally below. The (sub)-interfaces of the converter are self-explanatory and are connected to the resource that the interface is naming. For example, $\mathtt{in.ZKpopk}$, indicates the inside sub-interface of the converter that is connected to resource $\mathsf{ZK}_{\mathtt{popk}}$.

Moreover, as noted above, the assumed resources have security with abort. The protocol steps are executed sequentially, where messages from step r are computed only if all messages from step $r-1$ have been received, in line with the standard way of executing a synchronous protocol in an asynchronous network (see [32]).

Protocol $\Pi_{\mathtt{cdn}}(i)$

Key Setup

1: On input the public key $\mathtt{ek}$ and secret key share $\mathtt{dk}_i$ at interface $\mathtt{in.keygen}$, store them.

Input Distribution

1: On input x_i at interface $\mathtt{out}$, compute $\overline{x_i} = \mathtt{Enc}_{\mathtt{ek}}(x_i; r)$ and input $(\overline{x_i}, (x_i, r))$ at interface $\mathtt{in.ZKpopk}$.
2: On input $\overline{x_j}$ at interface $\mathtt{in.ZKpopk}$, assign this ciphertext as the input ciphertext of party j. Otherwise, assign a default ciphertext.

Addition Gates Input: $\overline{a}, \overline{b}$. Output: $\overline{c}$.

1: Locally compute $\overline{c} = \overline{a} \boxplus \overline{b}$.

Multiplication Gates Input: $\overline{a}, \overline{b}$. Output: $\overline{c}$.

1: Sample a random plaintext d_i and compute the ciphertexts $\overline{d_i} = \mathtt{Enc}_{\mathtt{ek}}(d_i; r_1)$ and $\overline{d_i b} = d_i \boxdot \overline{b} \boxplus \mathtt{Enc}_{\mathtt{ek}}(0; r_3)$. Then, output the triple $((\overline{d_i}, \overline{b}, \overline{d_i b}), (r_1, r_3))$ at interface $\mathtt{in.ZKpocm}$.
2: On input $(\overline{d_j}, \overline{b}, \overline{d_j b})$ at interface $\mathtt{in.ZKpocm}$, add j to the set S. That is, S is the set of parties that succeeded in the proof.
3: Compute $\overline{a} \boxplus \left(\boxplus_{i \in S} \overline{d_i}\right)$. Then, execute the Threshold Decryption sub-protocol on this ciphertext. Let $a + \sum_{i \in S} d_i$ be the decrypted plaintext.
4: Compute $\overline{c} = \left(a + \sum_{i \in S} d_i\right) \boxdot \overline{b} \boxminus \left(\boxplus_{i \in S} \overline{d_i b}\right)$.

Output

1: Upon obtaining c', the output ciphertext of the circuit, execute the Threshold Decryption sub-protocol on c'.

Threshold Decryption Input: ciphertext c. Output: y.

1: Compute a decryption share $s_i = \mathtt{DecShare}_{\mathtt{dk}_i}(c)$. Then, input $((\mathtt{ek}, c, s_i), \mathtt{dk}_i)$ at interface $\mathtt{in.ZKpocd}$.
2: Upon receiving $(\mathtt{ek}, c, s_j)$ from $t+1$ different parties at interface $\mathtt{in.ZKpocd}$, compute $y = \mathtt{Rec}(\{s_j\})$.

The following theorem states that the protocol $\Pi_{\mathtt{cdn}}$ achieves CC-adaptive security in the model assuming a threshold encryption setup and multi-party zero-knowledge resource. The main difference in the proof with respect to the passive protocol, is that the simulator extracts the inputs from corrupted parties from the inputs to the zero-knowledge resource, and also checks that the values received from the adversary interface satisfy the appropriate zero-knowledge relations.

Theorem 3. *Protocol $\Pi_{\mathtt{cdn}}$ CC-adaptively constructs MPC_f from $[\mathcal{ZK}, \mathsf{KeyGen}]$, with error ϵ and up to $t < n/2$ active corruptions, where ϵ reduces distinguishers to the corresponding advantage in the security of the threshold encryption scheme and is described in the proof.*

Proof. Let $\mathcal{R} = [\mathcal{ZK}, \mathtt{Keygen}]$ and $\mathcal{S} = \{\mathsf{MPC}_f\}$. Fix a set $X \subseteq \mathcal{P}$. We need to show that there is a simulator $\sigma_{\overline{X}}$ such that $\Pi_{\mathtt{cdn}}\mathcal{R} \subseteq (\sigma_{\overline{X}}\mathcal{S})^{\mathcal{E}_X \vee \mathcal{E}_t : \epsilon}$. At any point in time, if the event $\mathcal{E}_X \vee \mathcal{E}_t = 1$, the simulator halts. The simulator works as follows.

Key Generation. The simulator $\sigma_{\overline{X}}$ simulates this step by invoking the simulator for the threshold encryption scheme. Let $\mathtt{ek}$ denote the public key, and $\mathtt{dk}_i$ denote the decryption key share for party i. It then outputs $\mathtt{ek}$ at the adversary interface.

Network Messages. The simulator simulates each step of the protocol, given that all messages before that step have been delivered by the adversary i.e., the simulator receives all the corresponding `deliver` messages at the adversary interface. If not, it simply keeps waiting. The messages below in the steps below are output to the adversary at the corresponding steps, upon receiving the corresponding `leak` messages at the adversary interface.

Input Stage. For each party i that is honest at this step and gave input to the ideal resource, $\sigma_{\overline{X}}$ outputs an encryption c_i on behalf of this party at the adversary interface. If $i \in \mathcal{X}$, then the simulator does not know its input, and computes the ciphertext $c_i = \mathtt{Enc}_{\mathtt{ek}}(0)$ as an encryption of 0. Otherwise, $i \notin \mathcal{X}$ and the simulator knows its input x_i, so it computes $\overline{x_i} = \mathtt{Enc}_{\mathtt{ek}}(x_i)$ as the ciphertext. It then outputs the ciphertext c_i, indicating that the zero-knowledge proof was successful.

For each party i that is corrupted at this point, the simulator obtains (c, w) and checks that the witness w, consisting of a plaintext x and randomness r, satisfy the proof of plaintext knowledge relation, i.e., that $c = \mathtt{Enc}_{\mathtt{ek}}(x; r)$. It this holds, forward x to the ideal resource.

Addition Gates. This step can be simulated in a straightforward manner, performing local homomorphic operations on behalf of each honest party.

Multiplication Gates. Let $\overline{a}$ and $\overline{b}$ denote the ciphertexts input to the multiplication gate. The simulator $\sigma_{\overline{X}}$ can execute the honest protocol. That is, it generates a random value d_i on behalf of each honest party i, and locally computes $\overline{d_i} = \mathtt{Enc}_{\mathtt{ek}}(d_i)$, and $\overline{d_i b} = d_i \boxdot \overline{b}$. It then outputs the pair of ciphertexts to the adversary interface, indicating that the proof of correct multiplication is valid. For each corrupted party at this step, the simulator obtains (as input of the zero-knowledge proof of correct multiplication) the statement containing the ciphertexts c_1^i, $c_2 := \overline{b}$ and c_3^i as the statement, and as witness the plaintexts d_i, and randomness r_1 and r_3. The simulator checks that $c_1^i = \mathtt{Enc}_{\mathtt{ek}}(d_i; r_1)$, and $c_3^i = d_i \boxdot c_2 + \mathtt{Enc}_{\mathtt{ek}}(0; r_3)$. If this holds, accept the pair of ciphertexts from party i.

Upon receiving the pairs of ciphertexts from all parties, let S be the set of accepted parties. Then, compute the ciphertext $\overline{a} \boxplus (\boxplus_{i \in S} c_1^i)$, and simulate an honest threshold decryption protocol on this ciphertext. Let $a + \sum_{i \in S} d_i$ be the reconstructed plaintext. Then, compute the ciphertext $\overline{c} = (a + \sum_{i \in S} d_i) \boxdot c_2 \boxminus (\boxplus_{i \in S} c_3^i)$.

Output. The simulator inputs to the ideal resource $(\mathtt{deliver}, j)$, for each party j that obtains an output in the protocol. It also obtains the output with the instruction `leakOutput`. Then, upon obtaining an output y from the ideal resource, use the simulator of the (adaptively secure) threshold decryption protocol to compute decryption shares on behalf of the honest parties (see [35], where one can simply choose as the inconsistent party one of the parties in X).

Corruptions. On input a command `leak`, at interface $A.i$, if i is corrupted, the simulator outputs the internal state of party i. Note that this is easily done since for parties not in X, the simulator has access to its input. And if any party in X gets corrupted, the corresponding MBO is triggered, $\mathcal{E}_X = 1$, and the simulator halts.

We prove that $\Pi_{\mathtt{cdn}} \mathcal{R} \subseteq (\sigma_{\overline{X}} \mathcal{S})^{\mathcal{E}_X \vee \mathcal{E}_t : \epsilon}$ via a sequence of hybrids.

Hybrid H_1. Here, we assume that the simulator has access to all inputs from the parties. It then executes the real-world protocol, except that in the zero-knowledge proofs, when the simulator has to output a proof on behalf of an honest party it simply outputs a valid response without checking the witness from the honest party. It is trivial to see that the real-world specification is the same as H_1, since honest parties always send a valid witness to $\mathcal{ZK}$.

Hybrid H_2. We modify the above hybrid to also change the key generation and the decryption, which are now executed using the respective simulators for the threshold encryption scheme. By security of the threshold encryption scheme, we have that $\mathsf{until}_{\mathcal{E}_X \vee \mathcal{E}_t}(H_1)$ is ϵ_1-close to $\mathsf{until}_{\mathcal{E}_X \vee \mathcal{E}_t}(H_2)$. With the same argument as in the passive case, we have that:

$$\mathsf{until}_{\mathcal{E}_X \vee \mathcal{E}_t}(H_1) \in \left(H_2^{\mathcal{E}_X \vee \mathcal{E}_t]}\right)^{\epsilon_1} \iff H_1 \in H_2^{\mathcal{E}_X \vee \mathcal{E}_t : \epsilon_1}.$$

Hybrid H_3. This hybrid is the same as above, except that the simulator sets the input encryption of the honest parties in X during the Input Stage as an encryption of 0. By semantic security of the threshold encryption scheme, we have that $H_2 \in H_3^{\mathcal{E}_X \vee \mathcal{E}_t : \epsilon_2}$, where ϵ_2 is the advantage of the distinguisher (modified by the reduction) to the semantic security of the encryption scheme. Moreover, the hybrid H_2 corresponds exactly to the ideal specification $(\sigma_{\overline{X}}\mathcal{S})$.

Combining the above steps, we have that $\Pi_{\mathtt{cdn}}\mathcal{R} \subseteq (\sigma_{\overline{X}}\mathcal{S})^{\mathcal{E}_X \vee \mathcal{E}_t : \epsilon_X}$, where $\epsilon_X = \epsilon_1 + \epsilon_2$. Therefore, choosing the function ϵ where $\epsilon(D) = \sup_{X \subseteq \mathcal{P}}\{\epsilon_X(D)\}$ concludes the proof.

□

E Protocol and Proof of Lemma 9

The oblivious transfer protocol $\Pi_{\mathtt{ot}}$ from [25] is presented below. We describe the protocol in the n-party setting, where two of the parties, the sender s and the receiver r exchange the usual messages, and other parties do not perform any instructions.

Protocol $\Pi_{\mathtt{ot}}$

Converter for Sender s

1: On input $(x_1 \ldots, x_\ell)$ at `out`, choose a trapdoor permutation f over $\{0,1\}^\kappa$, and its inverse f^{-1}. Then, output f to `in.net`.r.
2: On input $(y_1 \ldots, y_\ell)$ at `in.net`.r, output $(b_1, \ldots, b_\ell) \coloneqq (x_1 \oplus B(f^{-1}(y_1)), \ldots, x_\ell \oplus B(f^{-1}(y_\ell)))$ to `in.net`.r, where $B(\cdot)$ is a hard-core predicate for f.

Converter for Receiver r

Set $f' = \bot$.
1: On input i at interface `out`, if $f' \neq \bot$, choose $y_1, \ldots, y_{i-1}, r, y_{i+1}, \ldots, y_\ell \in_R \{0,1\}^\kappa$, and compute $y_i = f(r)$, and output $(y_1, \ldots, y_\ell)$ at interface `in.net`.s.
2: On input f at interface `in.net`.s, set $f' = f$.
3: On input $(b_1, \ldots, b_\ell)$, output $b_i \oplus B(r)$ at `out`.

We prove that the protocol achieves CC-adaptive security against any number of passive corruptions.

Lemma 9. *Assume that enhanced trapdoor permutations exist. Then, $\Pi_{\mathtt{ot}}$ CC-adaptively constructs OT from $\mathcal{N}$, for error ϵ (described in the proof), and for any number of passive corruptions.*

Proof. Fix a set $X \subseteq \mathcal{P}$. We divide four cases, depending on whether the sender s or the receiver r are in the set X, and show that $\Pi_{\mathtt{ot}}\mathcal{N} \subseteq (\sigma_{\overline{X}}\mathsf{OT})^{\mathcal{E}_X : \epsilon}$. In all cases, the simulator halts at the point where a party in X is corrupted.

Case 1: $s \notin X$ and $r \notin X$. This case is trivial, since the simulator knows the inputs to both parties, and can therefore locally simulate all steps.

Case 2: $r \in X$. $\sigma_{\overline{X}}$ generates f, f^{-1} as in the protocol, outputs f at the adversary interface A. It then generates $y_1, \ldots, y_{i-1}, y_i, y_{i+1}, y_\ell$ randomly, where $y_j \in_R \{0, 1^\kappa\}$. Output $y_1, \ldots, y_\ell$ at interface A. Finally, compute each bit $b_i = x_i \oplus B(f^{-1}(y_i)$, $i \in [1, \ell]$. Output $b_1, \ldots, b_\ell$ at interface A, input $x_1, \ldots, x_\ell$ to OT and deliver the output to the receiver.

Corruptions. At any point in time, on input `leak` from A output the sender's secret state.

Simulation is perfect in this case. Since f is a permutation, choosing z at random and computing $y_i = f(r)$, as occurs in the real protocol, gives a uniform random y_i. Moreover, the equality $b_i = x_i \oplus B(f^{-1}(y_i))$ is satisfied. Therefore, both real and ideal systems behave the same until the event $\mathcal{E}_X$ triggers. This means that $\Pi_{\mathtt{ot}}\mathcal{N} \subseteq (\sigma_{\overline{X}}\mathsf{OT})^{\mathcal{E}_X : 0}$.

Case 3: $s \in X$. $\sigma_{\overline{X}}$ generates f, f^{-1} as in the protocol, outputs f at the adversary interface A. It then generates $y_1, \ldots, y_{i-1}, r, y_{i+1}, y_\ell$ randomly, where $y_j \in_R \{0, 1^\kappa\}$, and $y_i = f(r)$. Output $y_1, \ldots, y_\ell$ at interface A. Finally, input i at OT, and obtain x_i. Then, compute $b_1, \ldots, b_{i-1}, b_{i+1}, b_\ell$ as uniform bits, and set $b_i = x_i \oplus B(f^{-1}(y_i)$. Output $b_1, \ldots, b_\ell$ at interface A.

Corruptions. On input `leak` from A before Step 2, if the receiver r is corrupted, output the input i. If the corruption is after Step 2, output $y_1, \ldots, y_\ell$.

The only difference between the real and ideal world, is in the third message $b_1, \ldots, b_\ell$. The bit b_i is identical in both worlds, and is set to $x_i \oplus B(f^{-1}(y_i))$. For the other bits b_j, $j \neq i$, the simulation chooses them uniformly, while the protocol chooses them as $x_j \oplus B(f^{-1}(y_j))$. However, since $B(\cdot)$ is a hard-core predicate and y_j is uniform random, these are distinguishable up to the hard-core property.

Therefore, we have that until $\mathcal{E}_X = 1$, the real system is the same as the ideal system, up to the security of the hard-core predicate. That is, $\Pi_{\mathsf{ot}}\mathcal{N} \subseteq (\sigma_{\overline{X}}\mathsf{OT})^{\mathcal{E}_X : \epsilon_{hc}}$, where ϵ_{hc} is the advantage for the distinguisher modified by the reduction in distinguishing the hard-core bit from uniform.

Case 4: $s \in X$ and $r \in X$. In this case, the simulator $\sigma_{\overline{X}}$ simply generates f, f^{-1} as in the protocol, outputs f at the adversary interface A. It then generates $y_1, \ldots, y_\ell$, where $y_j \in_R \{0,1\}^\kappa$ and outputs this to interface A, and also sets the second messages $b_1, \ldots, b_\ell$ where $b_j \in_R \{0,1\}$ and outputs this to interface A.

This case can be argued similarly as the previous case. □

F Commit-and-Prove Resource

The commit-and-prove resource [13] is a generalization of the commitment resource. It is parameterized by a relation R and a designated party i, the committer. It consists of two phases. In the first phase, party i can commit to a value w, and all parties receive a "committed" message. In the second phase, instead of opening the value, the resource receives some value x, and checks whether $R(x, w) = 1$. If so, it outputs x to all parties, and otherwise it ignores the input. In fact, the resource allows the committer to commit to multiple values, and the relation can depend on all these values.

We denote the resource CP_R the parallel composition of resources $\mathsf{CP}_{i,R}$ for each designated party $i \in [n]$, and omit writing the relation when it is clear from the context.

Resource $\mathsf{CP}_{i,R}$

Local Variable

$\overline{w}$ is initially an empty list.
$\mathcal{Q} = \varnothing$.

Commit Phase

1: On input $(\texttt{commit}, id, w)$ at interface i, append w to $\overline{w}$ and output $\perp$ at interface i.
2: On input id at interface $j \neq i$, if a value with this id was committed, make the `receipt` available to the adversary, who then can choose to deliver the message at interface j.
3: On input id at the adversary interface, if a value with this id was committed, output `receipt` at interface j.

Prove Phase

1: On input $(\texttt{prove}, id, x)$ at interface i, if $R(x, \overline{w}) = 1$, add (id, x) to $\mathcal{Q}$.
2: On input id at interface $j \neq i$ if a pair (id, x) was stored, make x available to the adversary, who then can choose to deliver the message at interface j.
3: On input id at the adversary interface, if a pair (id, x) was stored, output x at the same interface.