Environmentally Friendly Composable Multi-Party Computation in the Plain Model from Standard (Timed) Assumptions

Brandon Broadnax[1], Jeremias Mechler[2], Jörn Müller-Quade[2]

[1] broadnax@ira.uka.de
[2] Karlsruhe Institute of Technology, {mechler,mueller-quade}@kit.edu

Abstract. Starting with the work of Rivest et al. in 1996, timed assumptions have found many applications in cryptography, building e.g. the foundation of the blockchain technology. They also have been used in the context of classical MPC, e.g. to enable fairness. We follow this line of research to obtain composable generic MPC in the plain model.
This approach comes with a major advantage regarding *environmental friendliness*, a property coined by Canetti et al. (FOCS 2013). Informally, this means that our constructions do not "hurt" game-based security properties of protocols that hold against polynomial-time adversaries when executed alone.
As an additional property, they can be plugged into any UC-secure protocol without loss of security.
Towards proving the security of our constructions, we introduce a variant of the UC security notion that captures timed cryptographic assumptions. Combining standard timed commitments and standard polynomial-time hardness assumptions, we construct a composable commitment scheme in the plain model. As this construction is constant-round and black-box, we obtain the *first fully* environmentally friendly composable constant-round black-box generic MPC protocol in the plain model from standard (timed) assumptions.

1 Introduction

In order to achieve the very strong notion of UC security, trusted setups are required [CF01]. However, in practice, trusted setups are often hard to come by. Therefore, a long line of research (e.g. [Pas03; BS05; LPV09; Gar+12; GKP18; Dac+13; PS04; CLP10; CLP13; Bro+17]) has investigated how composable multi-party computation (MPC) can be achieved in the plain model, i.e. only assuming authenticated communication channels.

Starting with the work of Pass [Pas03], many approaches for this task have been proposed. Common to their techniques is that the simulation is *environmentally unfriendly*, i.e. "hurts" the security of protocols that run along-side and that rely on polynomial-time hardness assumptions.

Formally, this is captured by the notion of *environmental friendliness* as defined by Canetti, Lin, and Pass [CLP13], which considers all game-based security properties of a protocol against polynomial-time adversaries.

The typical reason for limited environmental friendliness is a super-polynomial simulation, which can break polynomial-time assumptions used in other protocols, therefore impacting their security properties. This holds even if the super-polynomial resources are restricted by e.g. an angel.

However, super-polynomial simulation techniques are not the only danger to the security of other protocols: Non-uniform advice given to the simulator (e.g. as in [LPV09]) may impact the security of previously started protocols—even if they are concurrently composable and secure against non-uniform adversaries. This additional property is not considered by the definition of environmental friendliness.

Ever since composable MPC in the plain model has been investigated, the following question has been left unanswered:

Can we achieve composable MPC in the plain model that is friendly to protocols that are executed along-side and may have started previously?

Previous results suggest that a simulation technique that runs in polynomial-time and does not rely on non-uniform advice is needed. Such a simulation cannot be achieved, in principle, even by previous advanced approaches like angel-based security or shielded oracles. Therefore, a novel approach to overcome the impossibility results of the UC framework is needed.

With the advent of the blockchain era, timed cryptographic assumptions have seen widespread use in the real world. A very popular example is the *proof of work* protocol of the Bitcoin blockchain. Even though its hardness is not based on some well-understood cryptographic assumption, it has proven to work nevertheless for many years.

Timed variants for classic cryptographic primitives such as commitments can be constructed from timed assumptions that are inspired by well-understood standard assumptions. Rivest, Shamir, and Wagner [RSW96] have initiated this study and proposed a time-lock puzzle based on the hardness of factoring and the time required to square modulo a composite. Based on such assumptions, timed cryptographic primitives such as time-lock puzzles and timed-release encryption [RSW96] or timed signatures and timed commitments [BN00] can be constructed in the plain model. More recently, stronger primitives such as non-malleable time-lock puzzles and commitments have been constructed [Eph+20; KLX20] using a setup.

As timed assumptions can be broken in polynomial-time by definition, they seem destined to solve the problem of limited friendliness exhibited by previous approaches for composable MPC in the plain model. In the following, we thus investigate the following questions:

Can we use timed assumptions to achieve composable MPC in the plain model? What are the advantages and disadvantages of such an approach?

We answer this question affirmatively and propose a new approach for generic MPC in the plain model based on asymmetries that are only temporary and much smaller compared to previous approaches. Namely, these asymmetries consist of only a polynomial number of computation steps sufficient to leverage timed cryptographic assumptions. The very feasibility of this approach may seem surprising as timed cryptographic primitives eventually lose their security. For example, timed commitments will eventually leak their secret by definition. Previous constructions crucially rely on this not to happen, i.e. the complexity asymmetry and the ensuing security to hold throughout the whole execution. We side-step this problem by using timed assumptions to merely set up short-lived trapdoors that can only be used while the assumptions still hold. After their security has expired, the (now possibly leaked) trapdoor is useless for the adversary. Yet, a simulator can use it to establish a long-lived trapdoor based on some classical polynomial-time assumption.

We introduce the notion of TLUC security, which is based on UC security and cast in the unmodified UC framework. With TLUC, honest parties may set up timers with some timeout $\ell \in \mathbb{N}$ that expire when all entities have spent more than ℓ steps in total. This allows to capture the security of (stand-alone) timed primitives such as time-lock puzzles or timed commitment schemes. While computations performed by protocol parties, environment and adversary are counted against timers, computations performed by the simulator are not. This allows simulators to break timed assumptions "at no cost" in terms of run-time accounting, while always remaining polynomially bounded. Such a simulator can then, for example, extract a timed commitment while it is still hiding for the environment.

With respect to the question of environmental friendliness, it suffices to see that the notion of TLUC security is a meaningful special case of UC security, which is fully environmentally friendly. This already implies that our notion also features full environmental friendliness as defined by [CLP13].

In order to be friendly to previously started protocols, a uniform simulation, i.e. one that does not rely on non-uniform advice (also in the reductions), is needed. Looking ahead, this is indeed the case for our composable commitment scheme.

To the best of our knowledge, our techniques are the first to achieve both of these properties simultaneously.

Leveraging timed assumptions for composability comes with a number of additional advantages. Namely, our notion is UC-compatible in the sense that if π UC-realizes ϕ for arbitrary protocols π and ϕ, then π also TLUC-realized ϕ. TLUC security allows the reuse of UC protocols in the sense that one can take a UC-secure protocol ρ making one subroutine call to $\mathcal{F}$ that UC-realizes some ideal functionality $\mathcal{G}$ and replace $\mathcal{F}$ with its TLUC realization π. The composite protocol ρ^{π} is then guaranteed to TLUC-realize $\mathcal{G}$. These properties are not generally offered in full by other notions that allow composable general MPC in the plain model and are not implied by (limited) environmental friendliness. What is more, TLUC security is meaningful for ideal functionalities that rely

on (even uniform) polynomial-time assumptions. This is in contrast to e.g. SPS security, where such functionalities are affected by the super-polynomial simulator or non-uniform simulation [LPV09].

Unfortunately, TLUC security is not closed under composition. Thus, one has to manually prove that multiple instances of π TLUC-realize multiple instances of $\mathcal{F}$ (i.e. $\hat{\pi}$ TLUC-realizes $\hat{\mathcal{F}}$).

Like previous approaches for generic MPC in the plain model and even UC security, TLUC security is not friendly to *timed* game-based properties of other protocols, e.g. the timed hiding property of a timed commitment scheme. This property is neither captured by the definition of environmental friendliness nor fulfilled by any previous notion that allows composable MPC—not even UC security.

Towards realizing composable generic MPC, we first construct a commitment scheme that TLUC-realizes the ideal functionality for multiple commitments $\mathcal{F}_{\text{MCOM}}$. In more detail, we combine a (possibly malleable) timed commitment with a non-malleable commitment to construct a commitment that is equivocal and concurrently simulation-sound, i.e. retains its binding property even if the adversary sees equivocated commitments. We show that this suffices to replace the CRS of the UC-secure commitment scheme of Canetti and Fischlin [CF01] with coin-tosses, assuming that trapdoor one-way permutations with dense public description [DP92] exist. The resulting composable commitment scheme is constant-round, black-box, in the plain model and makes use of standard polynomial-time and standard timed assumptions only. We note that our approach is conceptually different from recent results [Eph+20; KLX20; Bau+20b; Bau+20a] which define non-malleable or composable timed primitives and realize them using a trusted setup.

Due to the reusability of UC protocols, we can plug our construction into any UC protocol in the $\mathcal{F}_{\text{MCOM}}$-hybrid model while maintaining TLUC security. Using e.g. a variant of the MPC protocol of Hazay and Venkitasubramaniam [HV15], we are the first to obtain a composable constant-round, black-box and environmentally friendly generic MPC protocol from standard polynomial-time and timed assumptions that does not impact the security of other protocols relying on (non-timed) polynomial-time hardness assumptions.

1.1 Related Work

Informally, the very strong impossibility results for UC security [CF01; CKL03; PR08; KL11] imply that UC security can only be achieved using some kind of trusted setup. Lindell [Lin03] has shown that the impossibilities are not due to the particular definition of UC security, but apply to general concurrent composition.

Ever since, there have been numerous attemps to circumvent these impossibility results at least partially by considering security notions that are weaker than standard UC security.

SPS Security, introduced by [Pas03], considers simulators that may have a super-polynomial run-time, giving them an advantage over the polynomially-bounded environment at the expense of environmental friendliness and UC reusability.

While earlier approaches such as [Pas03; BS05] require (non-standard) super-polynomial hardness assumptions, newer approaches such as [LPV09; Gar+12; GKP18] require only standard polynomial-time hardness assumptions.

Due to the complexity asymmetry between environment and simulator, these constructions do not offer general composition. Thus, concurrent self-composability of SPS-secure protocols is often proven in a non-modular way.he transitivity of SPS security holds only with respect to protocols whose security is not "hurt" by the stronger simulator, e.g. protocols that are information-theoretically secure such as [IPS08]. Thus, (general) reusability of UC protocols is lost.

[LPV09] have generalized the notion of UC security to $(\mathcal{C}_{\mathsf{env}}, \mathcal{C}_{\mathsf{sim}})$-security, where $\mathcal{C}_{\mathsf{env}}$ and $\mathcal{C}_{\mathsf{sim}}$ denote the complexity classes of environment resp. simulator. They present a construction for non-malleable zero-knowledge from UC puzzles that can be plugged into an appropriate generic MPC protocol. For their construction in the plain model, [LPV09] assume simulators that run in non-uniform polynomial-time while the environment runs in uniform polynomial-time. However, the non-uniform simulation may impact the security of protocols that have started in the past. Also, if $\mathcal{C}_{\mathsf{sim}}$ is non-uniform polynomial-time, then the security notion is not meaningful for ideal functionalities that rely on uniform polynomial-time hardness assumptions.

[Dac+13] have extended the work of [LPV09] by considering adaptive security. Starting with a UC puzzle, they construct a commitment scheme satisfying their new and strong notion of non-malleability from simulatable public-key encryption. This non-black-box and non-constant-round construction can then be plugged into an appropriate protocol, yielding adaptively secure composable generic MPC.

Recently, [GKP18] have presented a SPS-secure black-box OT protocol from constant-round semi-honest OT and collision-resistant hash functions, i.e. standard polynomial-time hardness assumptions only. Their construction is secure against static corruptions and has a lower round complexity than other constant-round constructions such as [Bro+17].

Angel-based Security and Environmental Friendliness. The weak composition properties of SPS security have subsequently been improved upon by notions where the simulator itself remains polynomially bounded, but is aided by some super-polynomial entity that is also available to the environment. Such frameworks include *Angel-based security* [PS04], or *UC with super-polynomial helpers* [CLP10]. [CLP10] construct a non-constant-round CCA commitment scheme from one-way functions and use it to realize the ideal functionality for commitments. Their construction can be plugged into any constant-round UC protocol ρ in the $\mathcal{F}_{\mathrm{COM}}$-hybrid model without losing security. This property, called *round robustness*, has been generalized by [CLP13] to the property of *environmental friendliness*. The helper of [CLP13] is environmentally friendly for protocols

whose security is proven via black-box reductions to game-based cryptographic hardness assumptions with bounded polynomial round complexity.

Shielded Oracles. [Bro+17] have introduced the notion of UC security with *shielded oracles* that strictly lies between SPS security and Angel-based security. Their construction for a composable commitment scheme makes use of standard polynomial-time hardness assumptions only, is constant-round and black-box. While their notion is not environmentally friendly, they showed that the constructions can be plugged into a special class of UC-secure protocols without loss of security.

Other Models and Notions. There have been proposed a number of different models which enable (composable) MPC in the plain model. The *timing model* introduced by [KLP05] considers a communication network with time bounds and parties that have access to a local clock with little drift. There, non-constant-round non-black-box MPC secure under general composition is possible. This is done by *delaying* other protocols that are executed concurrently and incomparable to our approach.

The notion of *input indistinguishability*, first defined by [MPR06] and generalized and strengthened by [Gar+12], is another security notion capturing concurrent self-composition that can be achieved in the plain model. However, the constructions of [MPR06; Gar+12] are non-black-box. Also, input indistinguishability is weaker than UC security.

Non-Malleable Time-Lock Puzzles and Commitments [Eph+20] have introduced the notion of *non-malleable time-lock puzzles* and timed commitments and present constructions in the random oracle model. Similar results have been obtained by [KLX20] in the algebraic group model. While both results can possibly be used as building blocks in our constructions, they are not in the plain model.

TARDIS and CRAFT. TARDIS [Bau+20b] extends the GUC framework [Can+07] to include a notion of *abstract time* and *ticked functionalities* whose behavior can depend on the elapsed time. In this setting, universally composable abstractions of time-lock puzzles can be defined and realized in the random oracle model. We note that the goal of [Bau+20b] is different than ours. We use stand-alone-secure and possibly *malleable* timed primitives such as (malleable) timed commitments in order to achieve composability in the plain model. In contrast to TARDIS, we do not aim to define composable security notions for timed primitives. CRAFT [Bau+20a] realizes composable MPC in the TARDIS framework with additional guarantees such as output-independent abort, also relying on a random oracle.

1.2 Our Results

New Security Notion for Composable Security. The notion of UC security considers entities that are polynomially bounded and inherently unaware of other computations going on. Thus, timed assumptions cannot be properly used in UC

protocols. With TLUC security, we consider a variant of UC security that allows a party P to set up *timers* associated with a number of steps ℓ. At any point, P may query if the execution experiment in total (including the environment, adversary and other protocol parties) has performed ℓ or more steps. This allows the use of timed cryptographic primitives such as timed commitments.

Similar to SPS security, our security notion is not closed under composition (Appendix C.6) and features the single-instance composition theorem only (Theorem 4).

Environmental Friendliness. Very informally, *environmental friendliness*, introduced by Canetti, Lin, and Pass [CLP13], deals with the problem of negative "side-effects" a protocol π may have on game-based properties of another protocol π' that runs *along-side* (where neither protocol is a subroutine of the other) and relies on polynomial-time hardness assumptions. Formally, this is captured in a stand-alone model for game-based security properties, cf. Appendix C.3. Previous notions that feature generic MPC in the plain model suffer from limited environmental friendliness because super-polynomial simulation, e.g. due to use of a super-polynomial helper, may break polynomial-time hardness assumptions of other protocols that run along-side, resulting in limited environmental friendliness. While not considered by the definition of environmental friendliness, giving the simulator non-uniform advice may hurt the security of (even non-uniformly) secure protocols or protocols that have been previously executed. Being a special case of UC security, TLUC security is fully environmentally friendly (Proposition 6).

We note that the established notion does not consider *timed* game-based properties such as the timed hiding property of a timed commitment scheme. As such, our notion as well as *all* previous notions such as e.g. SPS security, Angel-based security and even UC security are not fully friendly in this respect.

UC Compatibility and Reusability. As *all* UC protocols retain their security under our notion (UC compatibility, Proposition 4) and TLUC simulators run in strict polynomial-time, we can realize a UC-complete functionality $\mathcal{F}$ in TLUC and plug it into *any* existing UC-secure protocol making one subroutine call to $\mathcal{F}$ without loss of security (UC reusability, Corollary 3). This is not implied by environmental friendliness *per se*. As the simulation is always polynomial-time, (even uniformly only) computationally secure ideal functionalities are meaningful in our framework.

Composable Commitment Scheme in the Plain Model. Combining a timed commitment scheme and a pCCA-secure commitment scheme, we construct a non-malleable and partially simulatable coin-toss that is sufficient to "bootstrap" the CRS of a UC-secure commitment scheme such as the $\mathsf{UCC}_{\mathsf{OneTime}}$ scheme of Canetti and Fischlin [CF01] in the plain model. The resulting commitment scheme is concurrently composable and TLUC-realizes the ideal functionality for multiple commitments $\mathcal{F}_{\mathrm{MCOM}}$ (Theorem 6). As all our reductions can be

performed uniformly, π_{MCOM} does not hurt the security of any protocol making use of polynomial-time assumptions, including uniform ones.

Our protocols can also be easily adapted to other security notions, e.g. SPS security (see Section 7.1).

Composable Constant-Round Generic MPC in the Plain Model. Plugging our construction for $\mathcal{F}_{\text{MCOM}}$ into a variant of the generic MPC protocol due to [HV15], we obtain a constant-round black-box and environmentally friendly generic MPC protocol from standard polynomial and standard timed assumptions in the plain model (Theorem 7). We remark that our results are in the static corruption setting.

2 Overview of our Techniques

In order to achieve composable security, the simulator needs an advantage over to the environment. In the case of UC security, this advantage is the ability to simulate the setup. For example, the simulator may embed a trapdoor into a common reference string in a way that is indistinguishable for the environment.

In contrast, composable security in the plain model is usually achieved by using a superpolynomial simulation, e.g. by either allowing simulators having super-polynomial run-time complexity or by giving the polynomial-time simulator access to super-polynomial resources. The inherent drawback of such approaches is that superpolynomial simulation affects other polynomial-time protocols running concurrently. The consequences are two-fold: i) A protocol π realizing some functionality $\mathcal{F}$ using super-polynomial simulation cannot be plugged into arbitrary UC protocols ρ in the $\mathcal{F}$-hybrid model while retaining security. ii) The simulation may not be *environmentally friendly*, i.e. it may affect the security of polynomial-time protocols running along-side independently.

In order to circumvent these problems, we propose a dual approach that is, to the best of our knowledge, completely novel: Instead of giving an advantage to the simulator, which remains polynomially bounded, we *temporarily* restrict the environment in a way that allows the use of timed primitives such as timed commitments in a meaningful way. As the simulator is not subject to the same restrictions as the environment, it is still able to e.g. extract a timed commitment that remains hiding for the environment for some time.

To this end, we allow parties to set up *timers* parameterized by a number of steps t and require the environment to obey these timers. In our constructions, we use timers to "protect" the timed hiding property of timed commitment schemes. Informally, a timed commitment scheme has the following properties: i) There is a bound $t \in \mathbb{N}$ such that commitments remain hiding for adversaries performing at most t steps (*timed hiding*) and ii) there is an extraction algorithm that extracts a valid commitments in polynomial time $T > t$ (*polynomial-time extractability, binding*). For the formal definition, see Section 3.3.

By setting up appropriate timers, we ensure that timed commitments where the receiver is corrupted initially remain hiding for the environment. Conversely,

the simulator, which is not required to obey timers, can extract timed commitments created by the environment. This is sufficient to set up a long-lived trapdoor based on standard polynomial-time hardness assumptions.

These fine-grained asymmetries are not captured by previous notions for composable security. We thus introduce the notion of TLUC security, which is a variant of UC security that allows the use of timed assumptions. As TLUC simulators run in strict polynomial-time, TLUC-secure protocols are environmentally friendly to all game-based properties against polynomial-time adversaries of protocols running alongside.

In previous approaches, the simulator kept its advantage throughout and beyond the protocol execution. In our setting, this is not the case anymore. Indeed, the environment will eventually be able to break all timed assumptions and thus e.g. learn the values of *all* timed commitments. We thus had to come up with a novel simulation technique that allows the simulator to set up a trapdoor while preventing the environment, which will eventually be just as powerful as the simulator, to do the same. At the same time, the environment, will eventually "catch up" the run-time advantage of the simulator, must not be able to notice when the simulator sets up a long-lived trapdoor.

Combining stand-alone timed commitments with stand-alone non-malleable commitments, we construct a commitment scheme that is concurrently realizes many instances of $\mathcal{F}_{\text{COM}}$ under TLUC security.

Theorem 1 (Composable Commitments in the Plain Model, informal). *Assume that trapdoor PRGs with dense public description, perfectly binding homomorphic commitment schemes and timed commitment schemes exist. Then, there exists a black-box constant-round commitment scheme in the plain model that concurrently TLUC-realizes $\mathcal{F}_{\text{COM}}$.*

As TLUC simulation is strictly strict polynomial-time, it does not "hurt" the security of other polynomial-time protocols running concurrently. This allows us to re-use arbitrary UC protocols without loss of security.

Proposition 1 (UC Reusability). *Let ρ be a protocol that makes one subroutine call to a protocol ϕ such that ρ^{ϕ} UC-realizes σ for some protocol σ. Let π be a protocol such that π TLUC-realizes ϕ. Then, ρ^{π} TLUC-realizes σ.*

Using an appropriate generic MPC protocol in the $\mathcal{F}_{\text{COM}}$-hybrid model, we can achieve composable generic TLUC-secure MPC in the plain model.

Theorem 2 (Constant-Round MPC in the Plain Model, informal). *Assume that timed commitment schemes and perfectly binding homomorphic commitment schemes exist. Also, assume that enhanced trapdoor one-way permutations with dense public descriptions exist. Then, for every well-formed functionality $\mathcal{F}$, there exists a constant-round black-box protocol $\pi_{\mathcal{F}}^{BB}$ in the plain model such that $\pi_{\mathcal{F}}^{BB}$ concurrently TLUC-realizes $\mathcal{F}$.*

The resulting protocol is environmentally friendly.

In the following, we give a quick overview of the security notion, the composable commitment scheme and one important building block.

2.1 TLUC Security in a Nutshell

Timed primitives such as timed commitment schemes can be meaningfully used in practice. Consider performing a coin-toss using a timed commitment scheme secure for, say, $t = 10^{15}$ steps. Assuming that the adversary can perform at most 10^{10} steps per second (equating 10 GHz, assuming that steps equate cycles)[3], a coin-toss using this timed commitment should be considered secure if the adversary's second-round message comes within e.g. one second of receiving the timed commitment, with plenty time left as security margin.

TLUC Security. Unfortunately, this intuition is not easily captured in the UC framework, which neither offers a notion of time nor makes assumptions with respect to the (concrete) computational power of entities. Instead of considering a model with time or modifying the framework, we propose a variant of UC security, called TLUC security, that enables honest parties to check if more than ℓ steps have been performed since a certain point in the execution. This allows to capture the security guarantees of timed primitives and to use them in protocols.

With TLUC, parties can set up *timers* parameterized by an ID and a number of computation steps ℓ by sending $(\texttt{timer}, id, \ell)$ to the adversary. At any point, a party that has set up a timer may check if it has expired, i.e. if the whole execution experiment has performed ℓ or more steps since the timer has been set up. This is done by sending $(\texttt{notify}, id)$ to the adversary. The adversary queries the environment if the timer has expired answers with $(\texttt{notify}, id, b)$, where $b = 1$ denotes an expired timer and $b = 0$ an unexpired one.

Mechanisms. The correct handling of timers is ensured by considering only *legal environments* and *legal adversaries*. Intuitively, legal environments correctly account for timers set up by honest parties by never under-estimating the number of computation steps performed by the execution experiment relative to a presumptive execution of a protocol π (counting obliviously of the parties' inputs and outputs) and adversary $\mathcal{A}$, denoted by $\mathcal{Z}[\pi, \mathcal{A}]$. This guarantees that timed assumptions protect against environment and adversary, but can be broken by the simulator in polynomial time (as the environment $\mathcal{Z}[\pi, \mathcal{A}]$ always counts relative to π and $\mathcal{A}$, even when interacting with ϕ and $\mathcal{S}$). For technical reasons, we require handling of timers and inquiries to go through the adversary. An adversary is legal if it immediately and correctly forwards timer setup messages or status inquiries by honest parties, as well as the environment's responses. Based on this, we define TLUC emulation as a special case of UC emulation, and consider legal adversaries and environments only. At first glance, this might seem restrictive, but when considering standard UC protocols without timers, then all UC environments and adversaries are legal under our definition. Thus, the restrictions only apply for classes of protocols that are not considered by UC security.

[3] This is even more plausible when using timed cryptographic assumptions that are belived to be hard even for parallel adversaries.

We want to emphasize that we have only modified the *security notion* (TLUC security vs. UC security), but not the security framework itself (which is the UC framework in both cases), e.g. the control function or the machine model. Interestingly, not all properties of UC security carry over to TLUC security. An important example is the composition theorem, which holds for the single-instance case but not the general one. Even proving the properties that *do* carry over require non-trivial proofs due to emulation overhead that is of no consequence when considering polynomial-time security only.

2.2 Composable Commitment Scheme

Our main contribution is the composable commitment scheme π_{MCOM} in the plain model. Our construction is based on the $\mathsf{UCC}_{\mathsf{OneTime}}$ commitment scheme in the $\mathcal{F}_{\mathrm{CRS}}$-hybrid model due to Canetti and Fischlin [CF01].

In the original scheme $\mathsf{UCC}_{\mathsf{OneTime}}$, which is suitable for a single commitment only, the CRS consists of two parts: a pair of public keys (pk_0, pk_1) for a trapdoor PRG PRG (cf. [CF01]) as well as a uniformly random string $\sigma \in \{0,1\}^{4\kappa}$. With the knowledge of the associated secret keys (sk_0, sk_1) for the trapdoor PRG, it is possible to extract commitments. By changing the distribution of σ in an indistinguishable way, the commitment scheme becomes equivocal.

Assuming trapdoor one-way permutations with dense public description [DP92], one can, in principle, use a coin-toss to generate (pk_0, pk_1) and σ. This would, however, require a commitment scheme that is both extractable and equivocal, which is essentially what we try to construct.

However, when analyzing the commitment scheme $\mathsf{UCC}_{\mathsf{OneTime}}$ carefully, one sees that the knowledge of only *one* trapdoor, depending on which party is corrupted, is sufficient. The other trapdoor does not even have to exist. This allows to use a different approach: We perform one coin-toss for each part of the CRS. Only the part of the CRS for the relevant property (equivocation or extraction) needs to be simulatable, depending on which party is corrupted. In this setting, a commitment scheme that is either equivocal or extractable suffices as a building block for the coin-toss.

The commitment scheme SSCOM that we use for the coin-toss is straight-line equivocal. This suffices to set up the extraction trapdoor if the sender is corrupted by having the commitment receiver, played by the simulator, start the coin-toss for (pk_0, pk_1). The simulator can equivocate the result to public keys for which it knows the secret keys. Conversely, the coin-toss for σ is started by the commitment sender. If it is honest, the simulator can simulate the coin-toss such that σ contains an equivocation trapdoor. From that point on, $\mathsf{UCC}_{\mathsf{OneTime}}$ can be executed as-is, using the values obtained by this preamble phase instead of the CRS as in the original protocol. For each new commitment between two parties, this preamble phase is re-executed. A similar approach is used in [Dac+13].

The key insight here is that the commitment must be equivocated in time, i.e. before the coin-toss has finished. Even if the commitment were to become equivocal later on, it would be too late for a malicious committer to bias the result of the coin-toss. Looking ahead, we use a timed commitment to set up

the equivocation trapdoor. The simulator can straight-line extract this timed commitment early enough and thus simulate the coin-toss where it plays the committer. The other coin-toss, where the environment acts as committer, is also equivocal in principle. However, as long as the timed commitment remains hiding until after the coin-toss has finished, the environment is unable to bias the result.

Interestingly, this technique protects to a certain extent from timed assumptions that eventually turn out to be false. This is because the timed commitments only have to be secure while timers protecting them are active. Even if they turn out to be completely insecure immediately after the timer's expiration, a protocol's security is unaffected. That is a big difference to previous constructions using e.g. sub-exponential hardness assumptions, whose security needs to hold indefinitely.

To achieve concurrent self-composability, we need SSCOM to have additional properties beyond equivocality. In particular, SSCOM commitments created by the environment have to remain binding even if the environment receives equivocated openings for commitments created by the simulator. This established property is called *simulation-soundness*. However, previous definitions requiring simulation-soundness against *all* polynomial-time adversaries are not applicable to our setting. Such definitions would not allow the use timed commitments as trapdoor. We thus define the relaxed notion of *timed simulation-soundness* (Definition 3) for a setting where adversaries obey timers.

To the best of our knowledge, timed commitments have not been used in conjunction with a polynomial-time assumption to establish a long-term trapdoor before. We believe that this technique is of independent interest.

Construction 1 (Commitment Scheme π_{MCOM}, informal)**.** *Parameterized by a timed security parameter $\ell(\kappa)$ and a trapdoor PRG PRG with key space $\{0,1\}^{l(\kappa)}$ for some polynomial l, domain $\{0,1\}^{\kappa}$ and range $\{0,1\}^{4\kappa}$.*

Commit Phase.

1. *Upon receiving $(\mathtt{commit}, sid, cid, P_i, P_j, b)$ as input for the committer P_i, committer P_i and receiver P_j execute two coin-tosses with timed security parameter $\ell(\kappa)$ to generate*
   (a) *$(pk_0, pk_1) \in \{0,1\}^{l(\kappa)} \times \{0,1\}^{l(\kappa)}$ (the "extraction CRS") with the receiver acting as initiator in the coin-toss.*
   (b) *$\sigma \in \{0,1\}^{4\kappa}$ (the "equivocation CRS") with the sender acting as initiator in the coin-toss.*

   If both coin-tosses terminate successfully, both parties store $(sid, cid, (pk_0, pk_1, \sigma))$. Otherwise, they halt the execution.
2. *The committer samples $r \xleftarrow{\$} \{0,1\}^{\kappa}$ and sets $c = \mathsf{PRG}(pk_0, r)$ if $b = 0$ and $c = \mathsf{PRG}(pk_1, r) \oplus \sigma$ if $b = 1$. Then, the committer sends $(\mathtt{commitment}, sid, cid, c)$ to the receiver. The committer stores $(sid, cid, (b, r, c))$, the receiver stores (sid, cid, c) and outputs $(\mathtt{committed}, sid, cid, P_i, P_j)$.*

Unveil Phase.

1. *Upon receiving* $(\texttt{unveil}, sid, cid, P_i, P_j)$ *as input, the committer sends* $(\texttt{unveil}, sid, cid, (b, r))$ *to the receiver.*
2. *Upon receiving* $(\texttt{unveil}, sid, cid, (b, r))$ *from the sender, the receiver checks if* $c = \mathsf{PRG}(pk_0, r)$ *for* $b = 0$ *or if* $c = \mathsf{PRG}(pk_1, r) \oplus \sigma$ *for* $b = 1$*, relative to the values stored for this* (sid, cid)*. If the check is successful, the receiver outputs* $(\texttt{unveil}, sid, cid, P_i, P_j, b)$ *and halts otherwise.*

2.3 Construction SSCOM

We now give a short overview of the construction SSCOM (Construction 2) for a timed simulation-sound string commitment scheme, which is based on the commitment scheme due to [Bro+17], which is inspired by [DS13]. Roughly, our scheme works as follows: First, the committer creates κ 2-out-of-2 shares of the secret and commits to these shares individually, using a standard non-timed commitment scheme. Then, the committer commits to a random index vector $I \in \{0,1\}^\kappa$ chosen by the receiver, using a timed commitment. In the unveil phase, the committer first sends its shares without unveiling the share commitments. Then, the receiver unveils the commitment to I. Finally, the committer unveils the share commitments denoted by I (i.e. if the i-th bit of I is b, then the b-th share of the i-th share tuple is unveiled), while the other commitments remain unopened. If the commitment to I can be extracted before sending the shares in the unveil phase, the commitment is equivocal. As inconsistent share commitments remain unopened and hiding, a malicious receiver cannot distinguish between an equivocated and a honest commitment. If the unveil phase is finished before the commitment to I loses its timed hiding property, the commitment is binding.

The main difference to the construction of [Bro+17] is the use of a timed commitment scheme for the commitment to the index vector I, which allows polynomial-time equivocation of SSCOM commitments.

We want to note that we need only a weak form of extractability for the timed commitment: Due to the structure of the protocol, it does not matter if the simulator accidentally extracts an *invalid* timed commitment to the index set. This is because the protocol will abort early enough if the timed commitment cannot be opened correctly. We account for this in our relaxed definition of timed commitment schemes (Definition 1). Allowing this over-extraction may lead to more efficient constructions in contrast to timed commitment schemes where the receiver must be able to efficiently determine the validity after the commit phase, as required by the established definition of [BN00].

At first glance, one might think that the timed commitment to I must be non-malleable in order to achieve timed simulation-soundness. In practice, this would constitute a problem as we are not aware of non-malleable timed commitments from standard timed assumptions in the plain model. Fortunately, were able to circumvent this problem by requiring the share commitment scheme to be pCCA-secure (Definition 9). In order for the proofs to go through, we also had to move the timed commitment to I to the end of the commit phase.

Looking ahead to the proof of our composable commitment scheme, we also need the SSCOM commitment scheme to be (not necessarily straight-line) extractable. This can be easily achieved by using an appropriate extractable commitment scheme for the share commitments.

In the following, we present our results in more detail.

3 Definitions

3.1 Notation

Let $n \in \mathbb{N}$. Then, $[n]$ denotes the set $\{1, \ldots, n\}$. Let H_i be some hybrid. Then out_i denotes the output of H_i. $\mathsf{negl}(\kappa)$ denotes an unspecified negligible function in the security parameter $\kappa \in \mathbb{N}$. $x \xleftarrow{\$} Y$ denotes that x is drawn uniformly at random from the set Y. $x \leftarrow Y$ denotes that x is either the output of the probabilistic algorithm Y or sampled according to the probability distribution Y. Let π_1, π_2 be protocols. Then, $\pi_1 \geq_{\mathrm{UC}} \pi_2$ denotes that π_1 UC-emulates π_2 and $\pi_1^{\pi_2}$ denotes that π_1 makes at least one subroutine call to π_2.

3.2 Machine Model, Notion of Time

When considering polynomial-time hardness assumptions, the particularities of machine models rarely matter. This is because different (classical) machine models can be usually emulated by each other with polynomial run-time overhead or speedup. With polynomial-time being closed under addition and multiplication, polynomial-time hardness assumptions do not become insecure if there is a machine model where some problem can be solved (polynomially) more efficient.

In this paper, we consider timed primitives such as timed commitment schemes. For timed primitives, security often is only guaranteed against adversaries adhering to some kind of (concrete) run-time bound in a fixed machine model. For such assumptions, changing the machine model can make the difference between security and insecurity. This is obvious for stark differences, e.g. when going from a sequential to a parallel machine model when considering timed assumptions that hold only against sequential adversaries. However, this problem also manifests with more subtle changes like allowing a larger alphabet for Turing machines, which may result in a linear speedup.

More problems arise during security reductions that require the emulation of Turing machines. Suppose that we want to show the security of some protocol π by using a ℓ-bounded timed assumption. We call ℓ the *timed security parameter*. In the security proof, the adversary $\mathcal{A}'$ against the timed assumption has to internally emulate the ℓ-bounded adversary $\mathcal{A}$ as well as (parts of) the protocol π. Just internally emulating the ℓ-bounded adversary may incur an overhead that does not allow the reduction to go through, because $\mathcal{A}'$ may always require more than ℓ steps due to its emulation overhead, even when just running the code of $\mathcal{A}$ and relaying messages. Additional overhead may occur e.g. for extracting

the correct answer based on the internally emulated adversary's output. These caveats have to be accounted for.

Later on, we use timed primitives in the UC framework (cf. Section 5). While UC security can be stated using various machine models [Can01], we adhere to the standard model of interactive Turing machines. However, as e.g. the particular alphabet or the number of work tapes is left unspecified[4], so is the exact notion of run-time in that particular model. In order to argue about the security of timed assumptions in our security notion, we thus have to map the underspecified notion of run-time of interactive Turing machines as defined in the UC framework to the (possibly also underspecified) notion of run-time for the timed assumption. Following the Cobham-Edmonds thesis (see e.g. [Gol08]) or the extended Church-Turing thesis, we assume that this is always possible with a multiplicative polynomial overhead or speedup in a classical setting, i.e. when not considering quantum computations. For common machine models such as Turing machines, Boolean circuits or (parallel) random access machines, explicit emulation constructions and bounds for the overhead resp. speedup are known.

When constructing a protocol with security against $\ell(\kappa)$-bounded adversaries, we thus require the timed building blocks to be secure against adversaries with timed security parameter $\ell'(\ell(\kappa), \kappa)$[5] where ℓ' is a sufficiently large polynomial that accounts for possible run-time mismatches due to emulation overhead, reduction overhead or (polynomial) efficiency changes between machine models. As we do not want to make assumptions about the machine models being used, we do not explicitly specify ℓ'. However, as soon as all machine models and reductions are fixed, ℓ' is well-defined. Also, for our constructions, we show that ℓ' is sufficiently generic and e.g. is independent of the TLUC environment under consideration.

Note that the timed security parameter generally grows with increasing protocol nesting depth, similar to the tightness loss in standard reductions.

In our protocols, we use `timer` messages parameterized by an ID id to allow protocol parties later check if more steps than allowed by the timed security parameter ℓ have been elapsed by sending a message $(\texttt{notify}, id)$. If the answer is $(\texttt{notify}, id, 1)$, then more than ℓ steps have passed and we say that the "timer has timed out" or "expired". Conversely, $(\texttt{notify}, id, 0)$ denotes that the timer has not expired. Later on, we will only consider adversaries (or environments) that handle such messages correctly.

As the default machine model and execution experiment of UC are inherently sequential, we refer to *computation steps* instead of *run-time*, as the latter may capture many steps performed in parallel, which we want to count individually.

[4] Newer versions of the UC framework such as UC2020 explicitly allow multiple work tapes, allowing the emulation of other Turing machines with only additive overhead.

[5] In order to capture the setting where $\ell(\kappa)$ is constant but e.g. the reduction overhead depends on κ, we parameterize ℓ' with both values.

3.3 Timed Commitment Schemes

Boneh and Naor [BN00] have introduced the notion of *timed commitment schemes*. Instead of the hiding property holding against all polynomial-time adversaries, a (T, ℓ, ε)-timed commitment scheme guarantees the hiding property to hold only for some bound of steps ℓ performed by an adversarial receiver, except with probability ε.

However, the (ℓ, ε)-hiding property does not guarantee that there exists a value $T \in \mathbb{N}$ such that a valid timed commitment can be opened "forcefully" in at most $T > \ell$ steps. To this end, the definition of [BN00] also requires the existence of a `forced-open` algorithm that runs in time T, takes the transcript of a successful commit phase and outputs the unique value $v \in M$ committed to, where M is the message space of the commitment scheme. In other words, in addition to the binding property, a malicious committer must not be able to open its commitment to a value that is inconsistent with the output of the `forced-open` algorithm. This extractability is crucial for our simulation later on, as it guarantees that simulators can extract timed commitments in polynomial time (if T is bounded by a polynomial in κ).

In the definition of [BN00], timed commitment schemes have to exhibit a *soundness property* which requires that at the end of the commit phase, the receiver is "convinced" that running the `forced-open` algorithm will produce the value v committed to. While not formally defined, the definition of [BN00] also requires valid commitments to be efficiently recognizable by the receiver.

Looking ahead to our construction, we do not need valid timed commitments to be efficiently recognizable. In particular, we can deal with the over-extraction of invalid commitments, i.e. the case where `forced-open` outputs a value $v \in M$, even if the commitment cannot be unveiled. We call this property *weak extractability* and will account for this in the following definition.

Also, the hiding property informally described in [BN00] seems to be relatively weak, considering honestly created commitments only. Moreover, the adversary's steps are only counted after it is provided the transcript of a successful commit phase. Our definition of timed hiding (Definition 7) is standard and stronger in the sense that the commitment *receiver* may act maliciously. Also, we count the adversary's steps from the very beginning on. It is easy to see that the scheme due to [BN00] satisfies this stronger notion.

With [BN00] not giving a formal definition, we define weakly extractable timed commitment schemes as follows.

Definition 1 (Weakly Extractable Timed Commitment Scheme). *A tuple of ITMs* $\mathsf{TCOM} = \langle \mathsf{C}, \mathsf{R} \rangle$ *is called a* (T, ℓ, ε)-weakly extractable timed commitment scheme *with message space* M *if* $\langle \mathsf{C}, \mathsf{R} \rangle$ *is a* (ℓ, ε)*-hiding commitment scheme for which there exists a deterministic algorithm* `forced-open` *that, given a transcript* c *of a successful commit phase, outputs the unique value* $v \in M$ *committed to in at most* T *steps.*

We say that TCOM is *perfectly correct* if for all $\kappa \in \mathbb{N}$ and all $v \in M$,

$$\Pr[v^\star = v' = v \mid (z_\mathsf{C}, z_\mathsf{R}, c) \leftarrow \mathrm{out}\langle \mathsf{C}(v), \mathsf{R}(\varepsilon)\rangle(1^\kappa, \texttt{Commit}), \\ v' \leftarrow \mathrm{out}_\mathsf{R}\langle \mathsf{C}(z_\mathsf{C}), \mathsf{R}(z_\mathsf{R})\rangle(\texttt{Unveil}), v^\star = \texttt{forced-open}(c)] = 1$$

The perfect correctness can be naturally relaxed to statistical correctness.

We say that TCOM is *perfectly binding* and *weakly extractable* if for all (malicious) committers C^*, all $\kappa \in \mathbb{N}$ and all $a \in \{0,1\}^*$, it holds that

$$\Pr[v^\star = v' \mid (z_{\mathsf{C}^*}, z_\mathsf{R}, c) \leftarrow \mathrm{out}\langle \mathsf{C}^*(a), \mathsf{R}(\varepsilon)\rangle(1^\kappa, \texttt{Commit}), \\ v' \leftarrow \mathrm{out}_\mathsf{R}\langle \mathsf{C}^*(z_{\mathsf{C}^*}), \mathsf{R}(z_\mathsf{R})\rangle(\texttt{Unveil}), v^\star = \texttt{forced-open}(c) \wedge v' \in M] = 1$$

While the aforementioned properties do not state any requirements for the output of `forced-open` on invalid commitments (i.e. allows over-extraction), it implies the soundness requirement of [BN00] for valid commitments.

While Definition 1 is not concerned with the committer's run-time, we note that it may depend on all parameters, in particular T and ℓ. This is important for (proving) security properties that consider more than one commitment, e.g. the timed simulation-soundness (Definition 3).

Boneh and Naor [BN00] also present a constant-round and black-box construction based on the generalized BBS assumption that is secure against parallel adversaries. Also, their construction admits a super-polynomial gap between the number of steps needed to perform the commitment and the number of steps ℓ the commitment is secure against.

While [BN00] consider a machine model that admits parallel computations, we consider (weaker) sequential models of computation only.

Recently, Ephraim et al. [Eph+20] and Katz, Loss, and Xu [KLX20] have re-visited timed commitment schemes, providing formal definitions and new constructions. However, as they consider (non-interactive) timed commitment schemes with setups, their definitions are not easily applicable to our setting.

Timed commitments can also be constructed by combinding *sequential functions* [MMV13] and universal hash functions. However, such a construction has the drawback that both commit and unveil phase are computation-intensive. Still, it suffices for a feasibility result with a symmetric assumption.

4 Timed Simulation-Sound Commitment Schemes

Looking ahead to our construction of a composable commitment scheme (Section 6), we need a commitment scheme that is equivocal for a polynomial-time simulator. At the same time, commitments created by a malicious committer (e.g. by the UC environment) must remain binding sufficiently long. To this end, we first define the security notion of *timed simulation-soundness*. Also, we present the construction SSCOM that combines a possibly malleable timed commitment scheme with a non-timed pCCA-secure commitment scheme and satisfies the notion of timed simulation-soundness.

4.1 Timed Simulation-Soundness

Based on the established notion of simulation-soundness [MY04; GMY03] and inspired by the non-malleability notion of Dachman-Soled et al. [Dac+13], we define a concurrent and timed variant of simulation-soundness that is suitable for commitments where the binding property only holds temporarily (Definition 3). Intuitively, this *timed* simulation-soundness ensures that commitments produced by a malicious committer remain binding for a bounded adversary even if it concurrently receives equivocated commitments. While somewhat similar to the notion of *non-malleability with respect to unveil* or *opening* or *decommitment* ([DIO98; PR05; OPV08]), our definition is stronger in the sense that commit and unveil phases may overlap (similar to the definition of [Dac+13]).

The Experiment. In the experiment for *timed simulation-soundness*, a man-in-the-middle adversary acts as receiver in an unbounded number of instances ("left sessions") of some trapdoor commitment scheme. The adversary starts left sessions by providing a tag of its choice, along with an efficiently samplable and length-normal (cf. Definition 2) distribution. Only considering distributions facilitates easier proofs and more general definitions and is sufficient for our application. In each left session, the code of the trapdoor committer C_{trap} is executed. After the commit phase of a session has finished, the adversary may, at some point of its choice, start the unveil phase. At its onset, a value from the provided distribution is sampled and unveiled by the trapdoor committer.

In addition, the adversary acts as committer in one session ("right session"), again using a tag of its choice that must be unique compared to all other tags that will eventually be used in the experiment. The scheduling between all sessions and their messages is fully controlled the adversary.

When the commit phase of the single right session has finished, the experiment determines the value committed to. The commitment scheme is secure if the adversary cannot unveil its single commitment to a value different from the committed one, even when presented with equivocated commitments.

Timer-Related Parameters. In our setting, we do not consider simulation-soundness against arbitrary polynomial-time adversaries. Indeed, our construction SSCOM is (intentionally) not simulation-sound or even binding against polynomial-time adversaries: If a corrupted receiver manages to break a timed commitment it receives from the (honest) sender early enough, the commitment becomes equivocal. In our setting, protocol parties may set up *timers* and inquire at some point whether the timer has expired. The timed simulation-soundness experiment is thus parameterized with a timed security parameter ℓ. This timed security parameter denotes how many steps experiment and adversary may perform before the a timer set up by the honest receiver in the right session is considered to have timed out. If no timeout occurs, the binding property of the single right commitment should hold, even if left commitments are equivocated.

Timed simulation-sound commitments that use timed primitives such as timed commitments as building block must choose their timed security parameter ℓ'

relative to ℓ. Usually, to account for reduction overhead, e.g. to the timed hiding property of a timed commitment scheme, ℓ' must be chosen sufficiently large to account for this overhead. As the reduction overhead may depend on the security parameter κ but $\ell(\kappa)$ might be constant, ℓ' is also parameterized with κ. Depending on the commitment scheme, increasing ℓ may lead to the timer *always* expiring, e.g. because a sub-protocol protected by the timer requires more than ℓ steps to execute (e.g. the commit phase of a timed commitment scheme, which may take longer for larger ℓ) for some values of ℓ. In this case, proving security becomes trivial as the adversary cannot win the game. However, this also implies that scheme is secure in this case. When using appropriate building blocks, e.g. non-interactive timed commitments or a timed commitment scheme with a sufficiently large gap (e.g. the scheme of [BN00] has an exponential gap between the time needed to create the commitment and its timed security), this problem does not occur for sufficiently large ℓ'.

In order to notify parties about timeouts, we require the adversary to obey the following rules: When receiving a message $(\texttt{notify}, id)$ for some ID id, it must immediately answer $(\texttt{notify}, id, 1)$ if it has previously received $(\texttt{timeout}, id)$ and the whole execution experiment, including the adversary and committers in left sessions, has performed ℓ or more steps, where ℓ is the timed security parameter. For our construction, this can be easily calculated as the run-time of the involved algorithms do not depend on their internal randomness or secrets. If an exact calculation is not possible, the adversary must use an appropriate upper bound.

This is in contrast to e.g. Definition 7 where only the steps of the adversary are counted. There, this is possible as only one commitment session is considered. Here, we consider an unbounded number of sessions. In a reduction to some timed property, all the left sessions will have to be emulated by the reduction adversary, counting against its time limit in the reduction.

As the algorithms C and C_{trap} may require a different number of steps during their execution, the adversary must count the maximum number of steps either algorithm might need in order for reductions to go through. In our construction, it would be sufficient to only count the steps of the honest committer C_{trap}.

As the guarantees of timed cryptographic assumptions are only for honest parties, the experiment does not answer `notify` messages.

In real life, one can of course not expect that a possibly malicious party obeys these rules. However, if a timed primitive is believed to be secure for e.g. several days considering the computation power available to the other party, assuming a timeout after, say, one minute, should be sufficiently secure.

Relationship to Other Non-Malleability Notions. Similar to the simulation-based non-malleability notion of [Dac+13], security must hold if the commit and unveil phases on the left side are interleaved with the right session. However, in contrast to [Dac+13], we do not require the commitment on the right side to be concurrently extractable and also do not consider adaptive corruptions, leading to a different security notion.

Formal Definition. First, we define length-normal probability distributions as distributions where all elements of the sample space are of equal length.[6]

Definition 2 (Length-normal Probability Distribution). *Let $\mathcal{D}$ be a probability distribution over $\{0,1\}^*$ with sample space Ω. $\mathcal{D}$ is called* length-normal *if for all $x, y \in \Omega$, it holds that $|x| = |y|$. Let $|\mathcal{D}|$ denote $|x|$ for $x \in \Omega$.*

An example for a length-normal distribution is the uniform distribution $\mathcal{U}_n$ over $\{0,1\}^n$ with $|\mathcal{U}_n| = n$.

Definition 3 (Timed Simulation-Soundness). *A trapdoor commitment scheme* TRAPCOM *with message space $M \subseteq \{0,1\}^*$ is called $\ell(\kappa)$-*timed simulation-sound *if for all legal PPT adversaries $\mathcal{A}$, there exists a negligible function* negl *such that for all $\kappa \in \mathbb{N}$ and for all $z \in \{0,1\}^*$, it holds that*

$$\mathrm{Adv}^{SIMSOUND}_{\mathcal{A},\mathsf{TRAPCOM}}(\kappa, \ell(\kappa), z) := \Pr[\mathrm{Exp}^{SIMSOUND}_{\mathcal{A},\mathsf{TRAPCOM}}(\kappa, \ell(\kappa), z) = 1] \leq \mathsf{negl}(\kappa)$$

where the probability is over the random coins of the experiment and the adversary. An adversary $\mathcal{A}$ is called legal *if i) it immediately sends the message $(\texttt{notify}, id, 1)$ after receipt of $(\texttt{notify}, id)$ and the experiment (including the adversary) has performed more than or equal to $\ell(\kappa)$ steps after having received a message $(\texttt{timer}, id)$, where steps performed by the committer on left sides are counted as the maximum of steps of either the honest committer C or the actually running trapdoor committer $\mathsf{C}_{\mathrm{trap}}$ and ii) $\mathcal{A}$ sends* `commit-left` *messages only parameterized with efficiently samplable and length-normal distributions (cf. Definition 2) where the sample space Ω is a subset of the message space M.*

The random variable $\mathrm{Exp}^{SIMSOUND}_{\mathcal{A},\mathsf{TRAPCOM}}(\kappa, \ell(\kappa), z)$ is defined as follows:

1. *Start the adversary $\mathcal{A}$ with input $(1^\kappa, \ell(\kappa), z)$.*
2. *Upon receiving $(\texttt{commit-left}, tag, \mathcal{D}_{tag})$ from the adversary: Start the commit phase of* TRAPCOM *with common input $(1^\kappa, \texttt{commit}, tag, \ell(\kappa))$, acting as trapdoor committer $\mathsf{C}_{\mathrm{trap}}$ with private input $|\mathcal{D}_{tag}|$ unless there already is a session with tag tag.*
3. *Upon receiving $(\texttt{commit-right}, tag)$ from the adversary: Start the commit phase of the right session with common input $(1^\kappa, \texttt{commit}, tag, \ell(\kappa), \kappa)$, acting as honest receiver R, unless the right session already exists or there is a left session with tag tag. Let $v' \in M$ denote the* unique *value committed to in the right session. If no such unique value exists, set $v' = \perp$.*
4. *Upon receiving $(\texttt{unveil-left}, tag)$ from the adversary: Sample $v_{tag} \leftarrow \mathcal{D}_{tag}$ and start the unveil phase of the i-th left session with common input $(\texttt{unveil}, tag)$ and private input v_{tag} for the trapdoor committer, unless the commit phase with tag tag has not finished or the unveil phase has already started.*
5. *Upon receiving $(\texttt{unveil-right})$ from the adversary: Start the unveil phase of the right session with common input $(\texttt{unveil}, tag)$, acting as honest receiver where tag is the tag specified in the commit phase. Let u denote the value accepted by the receiver or $\perp$ in case of an abort.*

[6] When considering an appropriate encoding, the definition can be extended to e.g. group elements.

6. *Upon receiving* $(\texttt{message}, tag, m)$ *from the adversary, forward the message* m *to the session with tag* tag*. Conversely, forward messages to the adversary.*
7. *After the right unveil phase has finished, output 1 if the receiver in the right session has accepted and* $u \neq v' \wedge u \neq \bot \wedge v' \neq \bot$*, where* v' *is the value committed to in the right session. Otherwise, output* 0*.*

We also say that a commitment scheme fulfilling the above definition is $\ell(\kappa)$-simulation-sound.

Like [Dac+13], we call an adversary that wins the above experiment with at most negligible probability *non-abusing*, i.e. if its commitments remain binding even when presented with equivocated commitments. Note that this notion is only meaningful for commitments where the value committed to is uniquely determined (except with negligible probability) if the receiver accepts. To capture the general case, the definition has to be changed slightly.

4.2 Construction SSCOM

In the following, we present the construction SSCOM (Construction 2) for a timed simulation-sound string commitment scheme, which is based on the commitment scheme due to [Bro+17], which is inspired by [DS13]. Roughly, the scheme works as follows: Committer and receiver perform a commitment to a random index vector $I \in \{0,1\}^\kappa$ chosen by the receiver. They then perform 2κ commitments to pair-wise shares of the secret. In the unveil phase, the committer first sends its shares without unveiling the share commitments. Then, the receiver unveils the commitment to I. Finally, the committer unveils the share commitments denoted by I, while the other commitments remain unopened. If the commitment scheme used for I is extractable, the commitment is equivocal. As inconsistent share commitments remain unopened and hiding, a malicious receiver cannot distinguish between an equivocated and a honest commitment. In order to achieve concurrent security, we require the share commitment scheme to be pCCA-secure (Definition 9).

In contrast to the original construction of [Bro+17], we use a timed commitment scheme for the commitment to the index vector I, which allows polynomial-time equivocation of SSCOM commitments. Also, we move this timed commitment to I to the end of the commit phase. For the sake of simpler proofs, we assume that the commitment scheme for the shares is perfectly binding. However, this requirement can be relaxed to statistically binding.

To facilitate easy integration with our composable commitment scheme and the timed simulation-soundness definitions, SSCOM includes explicit messages to set up timers and to check if they have expired. Again, the party answering the timer status inquiry checks if both parties have performed ℓ or more steps since the timer has been set up and answers accordingly. In the simulation-soundness experiment, the answer is given by the adversary that is required to answer truthfully. Again, it would have been possible to only count steps by the party that has *not* set up the timer. However, counting the steps of both parties is more consistent with our other definitions and more convenient in reductions.

Construction 2 (Commitment Scheme SSCOM)**.** *Parameterized by a security parameter κ, a timed security parameter $\ell(\kappa)$, a pCCA-secure and perfectly binding commitment scheme $\mathsf{COM}_{\mathsf{pCCA}}$ and a $(T, \ell'(\ell(\kappa), \kappa), \mathsf{negl}(\kappa))$-weakly extractable timed commitment scheme TCOM.*

Commit Phase. On common input $(1^\kappa, \texttt{commit}, tag, \ell(\kappa))$, committer and receiver interact as follows:

1. *The committer creates 2κ shares $s_{1,0}, s_{1,1}, \ldots, s_{\kappa,0}, s_{\kappa,1}$ of its private input v by sampling $s_{m,0} \xleftarrow{\$} \{0,1\}^{|v|}$ and setting $s_{m,1} = v \oplus s_{m,0}$, $m = 1, \ldots, \kappa$.*
2. *For $m = 1, \ldots, \kappa$, $n = 0, 1$, committer and receiver start 2κ instances of $\mathsf{COM}_{\mathsf{pCCA}}$ on input common input $(1^\kappa, \texttt{commit}, (tag, m, n))$ in parallel. The committer's private input in the instance with tag (tag, m, n) is $s_{m,n}$.*
3. *The receiver samples an index vector $I \xleftarrow{\$} \{0,1\}^\kappa$ and sends $(\texttt{timer}, tag)$ to the committer. Then, committer and receiver start an instance of TCOM with common input $(1^\kappa, \texttt{commit}, \ell(\ell'(\kappa), \kappa))$. The receiver of SSCOM acts as committer with private input I.*

Unveil Phase. On common input $(\texttt{unveil}, tag)$, committer and receiver interact as follows:

1. *The committer sends the shares $(s_{1,0}, \ldots, s_{\kappa,1})$ to the receiver.*
2. *The receiver sends $(\texttt{notify}, tag)$ to the committer, which the receiver answers with $(\texttt{notify}, tag, b)$ where $b = 1$ if committer and receiver have spent more than or equal to $\ell(\kappa)$ steps since the timer has been set up. Otherwise, $b = 0$ indicates that less than $\ell(\kappa)$ steps in total have elapsed. If the committer answers with $(\texttt{notify}, tag, 1)$, the receiver aborts. Otherwise, the receiver checks that $s_{1,0} \oplus s_{1,1} = \cdots = s_{\kappa,0} \oplus s_{\kappa,1}$ and aborts if this does not hold. Otherwise, committer and receiver perform the unveil phase of TCOM. Additionally, the committer checks that the TCOM commitment can be opened in at most T steps by using the $\texttt{forced-open}$ algorithm. If this check fails, the committer aborts. If the unveil phase or the check fail, the receiver aborts.*
3. *Committer and receiver perform κ unveil phases of $\mathsf{COM}_{\mathsf{pCCA}}$ as follows: If $I[m] = 0$ for $m = 1, \ldots, \kappa$, unveil the commitment to $s_{m,0}$ with tag $(tag, m, 0)$. Otherwise, unveil the commitment to $s_{m,1}$ with tag $(tag, m, 1)$. Let $s'_{m,n}$ denote the value decommitted to in the session with index (m, n).*
4. *After all unveil phases have finished, the receiver checks that $s'_{m,I[m]} = s_{m,I[m]}$, $m = 1, \ldots, \kappa$. If this holds, the receiver outputs $s_{1,0} \oplus s_{1,1}$. Otherwise, it aborts.*

Algorithm of the Trapdoor Committer $\mathsf{C}_{\mathrm{trap}}$.

1. *On private input l in the commit phase, commit honestly to 0^l.*
2. *On private input $v \in \{0,1\}^l$ in the unveil phase, extract the timed commitment using the $\texttt{forced-open}$ algorithm to obtain the index vector I. If $\texttt{forced-open}$ fails, sample $I \xleftarrow{\$} \{0,1\}^\kappa$ uniformly at random. For $m = 1, \ldots, \kappa$, send $s_{m,1-I[m]} = v \oplus s_{m,I[m]}$ as shares that will not be unveiled. Otherwise, perform the unveil phase honestly.*

Theorem 3. *Let* $\mathsf{COM}_{\mathsf{pCCA}}$ *be a pCCA-secure and perfectly binding commitment scheme with message space* $M \subseteq \{0,1\}^*$. *Let* TCOM *be a* $(T, \ell'(\ell(\kappa),\kappa), \mathsf{negl}(\kappa))$*-weakly extractable timed commitment scheme for some polynomially bounded* $T > \ell'(\ell(\kappa),\kappa)$*, sufficiently large timed security parameter* $\ell'(\ell(\kappa),\kappa)$ *and negligible function* negl *with message space* $\{0,1\}^\kappa$. *Then,* SSCOM *is an* $\ell(\kappa)$*-simulation-sound and trapdoor commitment scheme with message space* M.

It is easy to see that a successful commit phase of SSCOM statistically determines the value committed to. Looking ahead to the security proof of our composable commitment scheme, we will additionally need this value to be extractable in the presence of concurrently equivocated left sides. For the definition of extractability and the proof of Theorem 3, see Appendix B.

Possible Instantiations. Our construction SSCOM makes use of a weakly extractable timed commitment scheme TCOM as well as a pCCA-secure and perfectly binding commitment scheme $\mathsf{COM}_{\mathsf{pCCA}}$. A possible instantiation for the latter is the commitment scheme of Goyal et al. [Goy+14] which is pCCA-secure [Bro+17], constant-round, non-black-box, parallel extractable and perfectly binding if using e.g. the commitment scheme due to Blum [Blu81] based on one-way permutations as elementary commitment. By using a perfectly binding and homomorphic commitment scheme, the construction becomes perfectly binding and black-box [Bre+15; Bro+17].

Corollary 1. *Assume that constant-round, black-box perfectly binding and homomorphic commitment schemes exist. Assume that constant-round, black-box timed commitment schemes exist. Then,* SSCOM *is a constant-round, black-box timed simulation-sound commitment scheme from standard assumptions.*

An example for a constant-round black-box homomorphic commitment scheme is the ElGamal commitment scheme based on the DDH assumption [ElG84]. With respect to the timed commitment scheme, we can e.g. use the scheme due to Boneh and Naor [BN00] based on the generalized BBS assumption, which is black-box and constant-round.

Corollary 2. *Assume that the DDH assumption and the generalized BBS assumption hold. Then, there exists a constant-round, black-box timed simulation-sound commitment scheme.*

5 TLUC Security in a Nutshell

Timed primitives such as timed commitment schemes can be meaningfully used in practice. Consider performing a coin-toss using a timed commitment scheme secure for, say, $t = 10^{15}$ steps. Assuming that the adversary can perform at most 10^{10} steps per second (equating 10 GHz, assuming that steps equate cycles)[7],

[7] This is even more plausible when using cryptographic assumptions that are belived to be hard even for parallel adversaries

a coin-toss using this timed commitment should be considered secure if the adversary's second-round message comes within e.g. one second of receiving the timed commitment, with plenty time left as security margin.

TLUC Security. Unfortunately, this intuition is not easily captured in the UC framework, which neither offers a notion of time nor makes assumptions with respect to the (concrete) computational power of entities. Instead of considering a model with time or modifying the framework, we propose a variant of UC security, called TLUC security, that enables honest parties to check if more than ℓ steps have been performed since a certain point in the execution. This allows to capture the security guarantees of timed primitives and to use them in protocols.

With TLUC, parties can set up *timers* parameterized by an ID and a number of computation steps ℓ by sending $(\texttt{timer}, id, \ell)$ to the adversary[8]. At any point, a party that has set up a timer may check if it has expired, i.e. if the whole execution experiment has performed ℓ or more steps since the timer has been set up. This is done by sending $(\texttt{notify}, id)$ to the adversary. The adversary queries the environment if the timer has expired answers with $(\texttt{notify}, id, b)$, where $b = 1$ denotes an expired timer and $b = 0$ an unexpired one.

Mechanisms. The correct handling of timers is ensured by considering only *legal environments* and *legal adversaries*. Intuitively, legal environments correctly account for timers set up by honest parties by never under-estimating the number of computation steps performed by the execution experiment relative to a presumptive execution of a protocol π (counting obliviously of the parties' inputs and outputs) and adversary $\mathcal{A}$, denoted by $\mathcal{Z}[\pi, \mathcal{A}]$. This guarantees that timed assumptions protect against environment and adversary, but can be broken by the simulator in polynomial time (as the environment $\mathcal{Z}[\pi, \mathcal{A}]$ always counts relative to π and $\mathcal{A}$, even when interacting with ϕ and $\mathcal{S}$). For technical reasons, we require handling of timers and inquiries to go through the adversary. An adversary is legal if it immediately and correctly forwards timer setup messages or status inquiries by honest parties, as well as the environment's responses. Based on this, we define TLUC emulation as a special case of UC emulation, and consider legal adversaries and environments only. At first glance, this might seem restrictive, but when considering standard UC protocols without timers, then all UC environments and adversaries are legal under our definition. Thus, the restrictions only apply for classes of protocols that are not considered by UC security.

Properties of TLUC Security. As we consider only a subset of the UC environments and adversaries, properties of UC security do not necessarily carry over to TLUC security, at least for protocols using timers. To the contrary, even

[8] In contrast to stand-alone experiments where `timer` messages are not parameterlized with the timed security parameter, we have chosen to do so in the TLUC setting because the mechanism should be agnostic of the currently executed protocol and its timed security parameter.

properties such as the completeness of the dummy adversary are difficult to prove if concrete time bounds must be adhered to. We show several properties such as transitivity with UC protocols, i.e. protocols whose security does not rely on timers[9], completeness of the dummy adversary or full compatibility with UC security as well as UC reusability, meaning that all UC-secure protocols are also TLUC-secure and can be composed with TLUC protocols without loss of security. With respect to the latter, we state the single instance composition theorem.

The ability of the simulator to break timed assumptions while environment and real-world adversary are unable to do is sufficient to construct a composable commitment scheme in the plain model. When, e.g., combining our commitment scheme with a UC-secure generic MPC protocol in the $\mathcal{F}_{\text{COM}}$- or $\mathcal{F}_{\text{MCOM}}$-hybrid model[10], we obtain a composable generic MPC protocol in the plain model.

While composable MPC in the plain model is already possible in a number of other frameworks, previous approaches rely on some sort of super-polynomial or non-uniform simulation. The first may affect the security of concurrently executed protocols relying on polynomial-time hardness assumptions, resulting in limited *environmental friendliness* as defined by [CLP13] or limited UC reusability. TLUC security only considers entities that run in strict polynomial time. The second may affect the security of protocols that have been previously started, even ones that are secure against non-uniform adversaries. Our feasibility results also hold for uniform PPT environments and adversaries. Thus, TLUC security is the first notion that features composable constant-round black-box MPC in the plain model from standard (timed) assumptions as well as full environmental friendliness and does not hurt the security of previously started protocols relying on polynomial-time assumptions.

Guide for the Reader. A full treatment of TLUC security and the proofs for its properties can be found in Appendix C. However, the above informal description is sufficient to understand the commitment construction in Section 6

5.1 Protocol Emulation

We define TLUC emulation in analogy to UC emulation.

Definition 4 (TLUC Emulation). *Let π and ϕ be protocols. We say that π TLUC-emulates ϕ if for all legal PPT adversaries $\mathcal{A}$, there exists a PPT simulator $\mathcal{S}$ such that for all legal PPT environments $\mathcal{Z}[\pi, \mathcal{A}]$ there exists a negligible function* negl *such that for all $\kappa \in \mathbb{N}, a \in \{0,1\}^*$ it holds that*

$$|\Pr[\mathsf{Exec}\big(\pi, \mathcal{A}, \mathcal{Z}[\pi, \mathcal{A}]\big)(\kappa, a) = 1] - \Pr[\mathsf{Exec}\big(\phi, \mathcal{S}, \mathcal{Z}[\pi, \mathcal{A}]\big)(\kappa, a) = 1]| \leq \mathsf{negl}(\kappa)$$

[9] A UC protocol π that UC-emulates an ideal functionality $\mathcal{F}$ may of course send `timer` messages. However, as UC emulation also considers environments that handle these messages arbitrarily, the security of π cannot rely on them.

[10] $\mathcal{F}_{\text{MCOM}}$ and the multi-session extension $\hat{\mathcal{F}}_{\text{COM}}$ of $\mathcal{F}_{\text{COM}}$ are equivalent [CR03].

If π TLUC-emulates ϕ, we write $\pi \geq_{\text{TLUC}} \phi$. When omitting the non-uniform input a, the notion of protocol emulation is uniform.

Note that in Definition 4, the environment $\mathcal{Z}$ is supposed to count the steps according to the execution with π and $\mathcal{A}$ even if it is actually interacting with ϕ and $\mathcal{S}$. This allows the PPT-bounded simulator $\mathcal{S}$ to perform more steps than the adversary $\mathcal{A}$ without triggering a time-out, allowing it to break timed assumptions. If ϕ is an UC protocol, its security is not affected by such a powerful simulator. In contrast, if ϕ is a protocol making use of timers, honest parties of the protocol ϕ may not rely on timing assumptions as the adversary $\mathcal{S}$ is allowed to violate them unnoticed.

Meaningfulness of TLUC Security. When introducing a new security notion, it is important to argue that it does not allow to prove the security of "obviously" insecure protocols. The basic idea behind TLUC security is the very same as behind established simulation-based security notions, where a protocol's security is defined through the ideal functionality it realizes. For simulation-based security notions, care has to be taken that the simulator's capabilities do not affect the security guarantees of the ideal functionality. For example, SPS security is not meaningful for ideal functionalities that use a polynomial-time hardness assumption like a signature scheme that can be broken by the super-polynomial simulator. As the TLUC simulation is always polynomial-time, is does not affect an ideal functionality that makes use of computational assumptions. What is more, we show that non-trivial functionalities can be realized using a *uniform* polynomial-time simulation.

In total analogy to both UC security and other composable security notions that admit general MPC in the plain model, we can show strong impossibility results. This underlines that the new mechanism of timers does not help the simulator *per se*.

5.2 Properties of TLUC Security

Having defined protocol emulation, we can state important properties of TLUC security in analogy to properties of UC security.

Proposition 2. *The dummy adversary $\mathcal{D}$ is legal.*

Proposition 2 immediately follows from the definition of the dummy adversary in the UC framework.

Proposition 3 (Completeness of the Dummy Adversary). *Let π and ϕ be protocols. Then, $\pi \geq_{\text{TLUC}} \phi$ if and only if π TLUC-emulates ϕ with respect to the dummy adversary.*

TLUC security is also compatible with UC security, meaning that UC-secure protocols are also TLUC-secure.

Proposition 4 (Compatibility with UC Security). *Let π, ϕ be protocols such that $\pi \geq_{\text{UC}} \phi$. Then, $\pi \geq_{\text{TLUC}} \phi$.*

Transitivity. In contrast to UC security, TLUC security is not transitive. This means that there exist protocols π_1, π_2, π_3 such that $\pi_1 \geq_{\text{TLUC}} \pi_2$ and $\pi_2 \geq_{\text{TLUC}} \pi_3$, but $\pi_1 \not\geq_{\text{TLUC}} \pi_3$. For an example, see Appendix C.5. However, we can state the following weaker and useful properties.

Also, TLUC emulation is transitive in conjunction with UC emulation.

Proposition 5 (TLUC-UC Transitivity). *Let π_1, π_2, π_3 be protocols. If $\pi_1 \geq_{\text{TLUC}} \pi_2$ and $\pi_2 \geq_{\text{UC}} \pi_3$, then it holds that $\pi_1 \geq_{\text{TLUC}} \pi_3$.*

Composition. In the following, we consider the case of a protocol ρ that makes one subroutine call to a protocol ϕ.

Theorem 4 (Single Instance Composition Theorem). *Let π, ϕ be subroutine-respecting protocols such that $\pi \geq_{\text{TLUC}} \phi$. Let ρ be a protocol that makes one subroutine call to ϕ. Then, $\rho^\pi \geq_{\text{TLUC}} \rho^\phi$.*

Let ρ be a protocol that UC-realizes the ideal protocol IDEAL($\mathcal{G}$) of some ideal functionality $\mathcal{G}$ and makes one subroutine call to the ideal protocol IDEAL($\mathcal{F}$) of some ideal functionality $\mathcal{F}$. Using Propositions 4 and 5 and Theorem 4, we can import ρ into TLUC, replace IDEAL($\mathcal{F}$) with an appropriate TLUC protocol while preserving security and conclude that the resulting composite protocol TLUC-realizes IDEAL($\mathcal{G}$).

Corollary 3 (UC Reusability). *Let ρ be a protocol that makes one subroutine call to a protocol ϕ such that $\rho^\phi \geq_{\text{UC}} \sigma$ for some protocol σ. Let π be a protocol such that $\pi \geq_{\text{TLUC}} \phi$. Then, $\rho^\pi \geq_{\text{TLUC}} \sigma$.*

Unfortunately, TLUC security is not closed under general composition. More concretely, this means that there exist protocols π and ϕ such that $\pi \geq_{\text{TLUC}} \phi$ holds, but $\rho^\pi \not\geq_{\text{TLUC}} \rho^\phi$, where ρ makes *multiple* subroutine calls to ϕ. For an example, see Appendix C.6.

Environmental Friendliness. UC security has the desirable property of *environmental friendliness* [CLP13], which informally ensures that game-based security properties of protocols running along UC protocols ("in the environment") are not impacted by the UC execution. Unfortunately, this property does not hold for *all* game-based security properties for many notions that allow generic MPC in the plain model due to the use of super-polynomial simulation. What is more, determining whether the game-based property holds may be non-trivial, requiring e.g. to consider the security proof of the protocol in question. However, as TLUC security is a special case of UC security and considers polynomial-time simulation only, it inherits the environmental friendliness of UC security.

For an explanation and definition of environmental friendliness, see Appendix C.3.

Proposition 6 (Environmental Friendliness of TLUC Security). *Let π be a protocol that TLUC-emulates the ideal protocol of some functionality $\mathcal{G}$. Then π is friendly to every (non-timed) game-based property P of a protocol Π with property P.*

Protocols running alongside composable MPC protocols may not only be affected by super-polynomial simulation, but also by non-uniform simulation. For example, Lin, Pass, and Venkitasubramaniam [LPV09] propose a variant of UC security where the environment runs in uniform polynomial-time, while the simulator runs in non-uniform polynomial-time. The non-uniform input of the simulator may impact the security of protocols that have started before the input is given to the simulator—even if these protocols are secure against non-uniform adversaries. As the definition of environmental friendliness is non-uniform, it does not capture this property.

Both the simulation and the reductions for our composable commitment scheme (Section 6) are uniform. Our constructions thus do not adversely affect security properties of previously started protocols that hold against polynomial-time adversaries.

Remark 1. Environmental friendliness as defined by [CLP13] is not meaningful for *timed* game-based properties such as the timed hiding property of a timed commitment scheme.

When considering an ideal functionality $\mathcal{F}$ and a concurrently executed protocol π using timed assumptions, the functionality $\mathcal{F}$ may already be unfriendly to timed properties of π. For example, $\mathcal{F}$ may perform computations that break time-lock puzzles used in π.

In the experiment of environmental friendliness, no simulator is not used. The (presumptive) simulator is only used to show that a protocol π is as friendly as a functionality $\mathcal{F}$ (which may already be unfriendly in our setting). Thus, the problems of environmental friendliness to protocols using timed assumptions start well before considering the effects of the simulation, which additionally affect the environmental friendliness.

To the best of our knowledge, this novel environmental friendliness for timed game-based properties is not fulfilled by *any* security notion for composable MPC—not even for UC security.

Non-Triviality. While there exists no general and formal definition of non-triviality in the UC framework, Canetti et al. [Can+02] consider a protocol π to be a non-trivial realization of $\mathcal{F}$ if $\pi \geq_{\mathrm{UC}} \mathrm{IDEAL}(\mathcal{F})$ and for all adversaries $\mathcal{A}$ that deliver all messages and do not corrupt any party, the simulator $\mathcal{S}$ allows all outputs generated by $\mathcal{F}$.

With TLUC security, this notion is not sufficient as it does not consider the possibility that a protocol aborts due to timeouts, which may, depending e.g. on the environment, occur even if the adversary delivers all messages.

As an example, let π be a protocol that non-trivially UC-emulates $\mathcal{F}_{\mathrm{COM}}$ and takes $t(\kappa)$ steps to execute successfully if all parties are honest. Now, let π' be the protocol that is identical to π, with the following exception. When receiving its input, the honest committer sets up a timer with $10t(\kappa)$ steps. At the onset of the unveil phase, it checks if the timer has expired and halts upon expiration. Clearly, π' should be considered non-trivial.

However, there exists a legal environment such that π' never generates output even if the legal adversary delivers all messages. As we do not want π' to be considered trivial if there also exists a legal environment $\mathcal{Z}$ for which π' always generates an output under the conditions outlined in [Can+02], we thus consider an appropriate notion that accounts for this[11].

Note that non-triviality may be lost under composition. To this end, take a protocol ρ^{ϕ} that makes one subroutine call to some protocol ϕ and is non-trivial. Replacing ϕ with its realization π that takes more steps than ϕ may make the composed protocol ρ^{π} trivial as timers in ρ may *always* be triggered due to the additional steps performed by the protocol π. However, that this does *not* render ρ^{π} insecure.

Impossibility Results. The well-known impossibility results due to Canetti and Fischlin [CF01] state that there is no bilateral (i.e. involving two communicating parties) and terminating (in the sense of correctness for honest parties) protocol π that UC-realizes $\mathcal{F}_{\text{COM}}$ in the plain model. This is due to the fact that if a protocol π is in the plain model, an environment is able to internally emulate every (presumptive) UC simulator for π.

We state the following variant of the impossibility result of [CF01] for TLUC-realizing $\mathcal{F}_{\text{COM}}$ in the plain model:

Theorem 5. *There exists no bilateral, non-trivial protocol π in the plain model where only one party sets up timers such that $\pi \geq_{\text{TLUC}} \mathcal{F}_{\text{COM}}$.*

By introducing a temporary asymmetry between simulator and environment, e.g. when the environment counts the steps relative to the real-world adversary, non-trivial and environmentally friendly realizations of UC-complete functionalities in the plain model using timed assumptions become possible.

6 Composable Commitments in the Plain Model

We are now ready to present our construction π_{MCOM} that TLUC-realizes the ideal functionality $\mathcal{F}_{\text{MCOM}}$ (Fig. 1) and prove its security. Our construction is based on the $\mathsf{UCC}_{\mathsf{OneTime}}$ commitment scheme in the $\mathcal{F}_{\text{CRS}}$-hybrid model due to Canetti and Fischlin [CF01], which is a variant of the trapdoor commitment scheme due to Di Crescenzo, Ishai, and Ostrovsky [DIO98], which is in turn based based on the commitment scheme due to Naor [Nao90].

In the original scheme $\mathsf{UCC}_{\mathsf{OneTime}}$, which is suitable for a single commitment only, the CRS consists of two parts: a pair of public keys (pk_0, pk_1) for a trapdoor PRG PRG (cf. [CF01]) as well as a uniformly random string $\sigma \in \{0,1\}^{4\kappa}$. With the knowledge of the associated secret keys (sk_0, sk_1) for the trapdoor PRG, it is possible to extract commitments. By changing the distribution of σ in an indistinguishable way, the commitment scheme becomes equivocal.

[11] As legal environments count the number of computation steps independent of the input, the existence of *one* environment with fixed input implies the existence of *other* environments with all possible inputs such that π always generates an output.

To enable simulation in the case of static corruptions, the knowledge of only *one* trapdoor, depending on which party is corrupted, is sufficient. The other trapdoor does not even have to exist. Assuming trapdoor one-way permutations with dense public description [DP92], we can perform two coin-tosses to generate (pk_0, pk_1) resp. σ. While our coin-toss protocol (see Section 6.1) is not fully simulatable, it is simulatable if the simulator plays the initiator. This suffices to set up the extraction trapdoor if the sender is corrupted by having the commitment receiver, played by the simulator, start the coin-toss for (pk_0, pk_1). The simulator can equivocate the result to public keys for which it knows the secret keys. Conversely, the coin-toss for σ is started by the commitment sender. If it is honest, the simulator can simulate the coin-toss such that σ contains an equivocation trapdoor. From that point on, $\mathsf{UCC}_{\mathsf{OneTime}}$ can be executed as-is, using the values obtained by this preamble phase instead of the CRS as in the original protocol. For each new commitment between two parties, this preamble phase is re-executed. A similar approach is used in [Dac+13].

Our coin-toss protocol π_{CT} uses the trapdoor commitment scheme SSCOM (see Section 4.2) whose equivocation trapdoor is protected by a timed commitment that can be extracted by the simulator. As SSCOM is timed simulation-sound, SSCOM commitments of corrupted committers remain binding.

TLUC security does not imply concurrent self-composability. Thus, we cannot simply prove the security of a single commitment and conclude that it holds for multiple commitments performed concurrently. Indeed, when using weaker building blocks, our construction can be shown to securely realize one instance of $\mathcal{F}_{\mathrm{COM}}$, but not $\mathcal{F}_{\mathrm{MCOM}}$, where the latter captures concurrent self-composition.

In the following, we thus prove that π_{MCOM} TLUC-realizes the ideal functionality $\mathcal{F}_{\mathrm{MCOM}}$ for multiple commitments. Later on, we can plug π_{MCOM} into any (UC-secure) protocol making one subroutine call to $\mathcal{F}_{\mathrm{MCOM}}$ while maintaining security.

6.1 The Coin-Toss Protocol π_{CT}

One important building block towards constructing our TLUC-secure commitment scheme is the coin-toss protocol π_{CT} (Construction 3). It is essentially identical to the protocol due to Blum [Blu81], except for the use of a string commitment and with the addition of handling the timers of SSCOM.

Construction 3 (Coin-Toss Protocol π_{CT})**.** *Parameterized by a security parameter κ, a timed security parameter ℓ, a length parameter $s = s(\kappa)$ and a $\ell'(\ell, \kappa)$-simulation-sound commitment scheme* SSCOM *with message space $M \supseteq \{0,1\}^s$.*

1. *On input* $(\texttt{coin-toss}, sid, s)$*, the sender samples* $r \xleftarrow{\$} \{0,1\}^s$ *uniformly at random.*
2. *Sender and receiver start an instance of* SSCOM *on common input* $(1^\kappa, \texttt{commit}, sid, \ell(\kappa), \ell'(\ell(\kappa), \kappa))$*. The sender's private input for the commitment is r. All* `notify` *messages are forwarded between the adversary and the parties of* SSCOM*. Messages* $(\texttt{timer}, id)$ *coming from a* SSCOM *party are*

forwarded to the adversary as $(\texttt{timer}, id, \ell)$*, i.e. augmented with the timed security parameter* ℓ*.*

3. *After the commit phase has finished, the receiver samples* $r' \xleftarrow{\$} \{0,1\}^s$ *uniformly at random and sends* r' *to the sender.*
4. *Upon receiving* r'*, sender and receiver perform the unveil phase of* SSCOM*.*
5. *If the receiver accepts, sender and receiver output* $(\texttt{coin-toss}, sid, r \oplus r')$*. Otherwise, the execution halts.*

As SSCOM is not straight-line extractable, we cannot show that π_{CT} TLUC-realizes the coin-toss functionality $\mathcal{F}_{\mathrm{CT}}$. However, π_{CT} exhibits the following useful properties: If the commitment receiver is corrupted, the coin-toss is simulatable. If the sender is corrupted and does not abort, the result of the coin-toss is distributed uniformly at random. Due to the simulation-soundness of SSCOM, the result of one session is independent from all other instances of π_{CT} that may run concurrently, with the exception of aborts skewing the distribution.

We do not prove these properties on their own, but show them implicitly in the proof of the construction of the commitment scheme.

6.2 The Commitment Scheme π_{MCOM}

We now give the construction of the composable commitment scheme π_{MCOM}.

Construction 4 (Commitment Scheme π_{MCOM})**.** *Parameterized by a timed security parameter* $\ell(\kappa)$ *and a trapdoor PRG* PRG *with key space* $\{0,1\}^{l(\kappa)}$ *for some polynomial* l*, domain* $\{0,1\}^{\kappa}$ *and range* $\{0,1\}^{4\kappa}$*.*

Commit Phase.

1. *Upon receiving* $(\texttt{commit}, sid, cid, P_i, P_j, b)$ *as input for the committer* P_i*, committer* P_i *and receiver* P_j *execute two instances of* π_{CT} *with timed security parameter* $\ell(\kappa)$ *to generate*
   (a) $(pk_0, pk_1) \in \{0,1\}^{l(\kappa)} \times \{0,1\}^{l(\kappa)}$ *(the "extraction CRS") with the receiver acting as initiator in* π_{CT} *with session ID* $(sid, cid, 0)$*, where* $l(\kappa)$ *is the length of public keys of* PRG*.*
   (b) $\sigma \in \{0,1\}^{4\kappa}$ *(the "equivocation CRS") with the sender acting as initiator in* π_{CT} *with session ID* $(sid, cid, 1)$*.*

   If both instances of π_{CT} *terminate successfully, both parties store* $(sid, cid, (pk_0, pk_1, \sigma))$*. Otherwise, they halt the execution.*
2. *The committer samples* $r \xleftarrow{\$} \{0,1\}^{\kappa}$ *and sets* $c = \mathsf{PRG}(pk_0, r)$ *if* $b = 0$ *and* $c = \mathsf{PRG}(pk_1, r) \oplus \sigma$ *if* $b = 1$*. Then, the committer sends* $(\texttt{commitment}, sid, cid, c)$ *to the receiver. The committer stores* $(sid, cid, (b, r, c))$*, the receiver stores* (sid, cid, c) *and outputs* $(\texttt{committed}, sid, cid, P_i, P_j)$*.*

Unveil Phase.

1. *Upon receiving* $(\texttt{unveil}, sid, cid, P_i, P_j)$ *as input, the committer sends* $(\texttt{unveil}, sid, cid, (b, r))$ *to the receiver.*

2. *Upon receiving* $(\texttt{unveil}, sid, cid, (b, r))$ *from the sender, the receiver checks if* $c = \mathsf{PRG}(pk_0, r)$ *for* $b = 0$ *or if* $c = \mathsf{PRG}(pk_1, r) \oplus \sigma$ *for* $b = 1$*, relative to the values stored for this* (sid, cid)*. If the check is successful, the receiver outputs* $(\texttt{unveil}, sid, cid, P_i, P_j, b)$ *and halts otherwise.*

We discuss how to generalize our result to hold in different frameworks in Section 7.1.

7 Proof of Security

Theorem 6. *Assume that* PRG *is a trapdoor PRG with dense public description and that* SSCOM *is a (computationally) trapdoor, extractable and timed simulation-sound commitment scheme. Then,* $\pi_{\mathrm{MCOM}} \geq_{\mathrm{TLUC}} \mathrm{IDEAL}(\mathcal{F}_{\mathrm{MCOM}})$.

Proof. First, we define the simulator for π_{MCOM} and the dummy adversary.

Definition 5 (The Simulator $\mathcal{S}$).

Corrupted Receiver, Honest Sender.

1. *Upon receiving the delayed output* $(\texttt{committed}, sid, cid, P_i, P_j)$ *from* $\mathcal{F}_{\mathrm{MCOM}}$*, play the coin-toss for the extraction CRS honestly and report all messages. In case of an error or timeout, halt.*
2. *Sample* $r_0, r_1 \xleftarrow{\$} \{0,1\}^\kappa$ *and play the coin-toss for the equivocation CRS using the algorithm of the trapdoor committer* $\mathsf{C}_{\mathrm{trap}}$*. Equivocate the commitment such that the result of the coin-toss is* $\sigma = \mathsf{PRG}(pk_0, r_0) \oplus \mathsf{PRG}(pk_1, r_1)$ *for* $r_0, r_1 \xleftarrow{\$} \{0,1\}^\kappa$*. In case of an error, halt.*
3. *Perform the commit phase by reporting* $c = \mathsf{PRG}(pk_0, r_0)$ *as message from the sender. Upon successful message delivery, allow the delayed output.*
4. *Upon receiving the delayed output* $(\texttt{unveil}, sid, cid, P_i, P_j, b)$ *from* $\mathcal{F}_{\mathrm{MCOM}}$*, report* $(\texttt{unveil}, sid, cid, (0, r_0))$ *if* $b = 0$ *and* $(\texttt{unveil}, sid, cid, (1, r_1))$ *if* $b = 1$*. Upon successful message delivery, allow the delayed output.*

Corrupted Sender, Honest Receiver.

1. *Generate keys* $(pk_i, sk_i) \leftarrow \mathsf{PRG.TGen}(1^\kappa)$ *for* $i = 0, 1$ *and perform* π_{CT} *for the extraction CRS using the algorithm of the trapdoor committer* $\mathsf{C}_{\mathrm{trap}}$*. Equivocate the commitment such that the coin-toss result is* (pk_0, pk_1)*. In case of an error, halt.*
2. *Play the coin-toss for the equivocation CRS honestly and report all messages. In case of an error or timeout, halt.*
3. *Upon receiving* $(\texttt{commitment}, sid, cid, c)$ *from the environment:*
   - *If* c *is in the range of* $\mathsf{PRG}(pk_0, \cdot)$*, send* $(\texttt{commit}, sid, cid, P_i, P_j, 0)$ *to* $\mathcal{F}_{\mathrm{MCOM}}$ *on behalf of the corrupted sender* P_i*.*
   - *Otherwise, send* $(\texttt{commit}, sid, cid, P_i, P_j, 1)$ *to* $\mathcal{F}_{\mathrm{MCOM}}$ *on behalf of the corrupted sender* P_i*.*

Subsequently, allow the public delayed output of $(\texttt{committed}, sid, cid, P_i, P_j)$ *of the honest receiver.*

4. *Eventually, receive* $(\texttt{unveil}, sid, cid, (b, r))$ *from the environment. If* c *is a valid commitment to* b *and* b *agrees with the value sent to* $\mathcal{F}_{\text{MCOM}}$*, send* $(\texttt{unveil}, sid, cid, P_i, P_j)$ *to* $\mathcal{F}_{\text{MCOM}}$ *and allow the public delayed output. Otherwise, if* $b = 1$ *does not agree with the value sent to* $\mathcal{F}_{\text{MCOM}}$ *but* r *is in the range of* $\mathsf{PRG}(pk_1, \cdot)$*, output a special error symbol* $\perp_{\text{FAIL}}$*. Otherwise, i.e. if the commitment is invalid, halt.*

Both Parties Honest.
Identical to the case of the corrupted receiver.

We prove Theorem 6 by showing that the simulator given in Definition 5 is valid for the dummy adversary, which is sufficient (see Proposition 3). First, we define a series of hybrids for the proof, ordered by the start of the individual commitments. In order to better distinguish between elementary commitments in π_{CT} and commitments to be performed with π_{MCOM}, we also refer to the latter as *sessions* of π_{MCOM}. Starting with an all-real execution, we consecutively introduce the simulator into the execution.

For each hybrid, we show the indistinguishability of the environment's output compared to the previous hybrid. We also show that the environment is non-abusing, i.e. does not equivocate commitments, despite the equivocations performed by the simulator. While both properties are related, they require separate proofs at first. The hybrids are defined as follows:

- H_0: The real execution with the dummy adversary $\mathcal{D}$ and protocol π_{MCOM}.
- H_1^i: The execution with a simulator $\mathcal{S}_1^i$ and the protocol IDEAL$(\mathcal{F}_1)$ for the ideal functionality $\mathcal{F}_1$ that gives the inputs of all honest parties to the simulator and lets it determine all outputs. $\mathcal{S}_1^i$ honestly executes π_{MCOM} but uses the code of the trapdoor committer C_{trap} of SSCOM whenever the simulator $\mathcal{S}$ would do so in the first i sessions. Outputs of simulated honest parties are forwarded to $\mathcal{F}_1$.
- H_2^i: Identical to H_1^q with $\mathcal{F}_2 = \mathcal{F}_1$ and $\mathcal{S}_2^i = \mathcal{S}_1^i$, apart from the following changes for sessions $k \leq i$:
  - If, in the k-th session, only the sender (of π_{MCOM}) is corrupted, $\mathcal{S}_2^i$ generates $(sk_j, pk_j) \leftarrow \mathsf{PRG.TGen}(1^\kappa)$ for $j \in \{0, 1\}$ and equivocates the first coin-toss to (pk_0, pk_1).
  - If, in the k-th session, only the receiver (of π_{MCOM}) is corrupted or both sender and receiver are honest, $\mathcal{S}_2^i$ samples $r_0, r_1 \xleftarrow{\$} \{0,1\}^\kappa$, equivocates the second coin-toss to $\sigma = \mathsf{PRG}(pk_0, r_0) \oplus \mathsf{PRG}(pk_1, r_1)$ and commits by sending $c = \mathsf{PRG}(pk_0, r_0)$ as commitment.
- H_3^i: Identical to H_2^q, but $\mathcal{F}_3^i$ behaves like $\mathcal{F}_{\text{MCOM}}$ for the first i sessions and like $\mathcal{F}_2$ otherwise. $\mathcal{S}_3^i$ is identical to $\mathcal{S}_2^q$, except that it runs $\mathcal{S}$ for the first i sessions. If $\mathcal{S}$ outputs $\perp_{\text{FAIL}}$ in the j-th session, output $\perp_{\text{FAIL}}^j$.
- H_4: The ideal execution.

We now present the full proof.

Lemma 1. *If* SSCOM *is (computationally) trapdoor, then* out_1^i *and* out_1^{i+1} *are (computationally) indistinguishable. If* SSCOM *is timed simulation-sound, then, for an appropriate (polynomial) timed security parameter* ℓ' *for* SSCOM*, the environment is non-abusing in* H_1^i*.*

Proof. We prove the indistinguishability by a reduction to the trapdoorness property of SSCOM. Then, we show that the environment is non-abusing.

Indistinguishability. Let $\mathcal{A}'$ be the following adversary against the trapdoorness of SSCOM:

1. On input $(1^\kappa, z)$, $\mathcal{A}'$ internally starts an instance of H_1^i with input $(1^\kappa, z)$ for the environment.
2. If there is a commitment in the $i+1$-th session for which the trapdoor committer C_{trap} would be used, perform this commitment externally with the trapdoorness experiment.
3. Eventually, $\mathcal{A}'$ outputs what the environment outputs.

As the trapdoorness property is not timed, $\mathcal{A}'$, running in polynomial time, constitutes a valid adversary. By definition of $\mathcal{A}'$, its advantage in the oracle trapdoor game is identical to the distinguishing advantage of the environment between H_1^i and H_1^{i+1}, i.e. $\text{Adv}^{\text{OTD}}_{\mathcal{A}',\mathsf{SSCOM},\mathcal{O}} = |\Pr[\text{out}_1^i = 1] - \Pr[\text{out}_1^{i+1} = 1]|$, which is negligible as SSCOM is oracle-trapdoor. It follows that $|\Pr[\text{out}_1^0 = 1] - \Pr[\text{out}_1^q = 1]| \leq q(\kappa) \cdot \mathsf{negl}_{\text{OTD}}(\kappa)$.

Non-Abusing. For the use in later hybrids, we need to show that the environment does not equivocate commitments where it is the sender, even if it receives equivocated commitments (cf. Definition 3). In analogy to [Dac+13], we call this property *non-abusing*. In order to show that the environment is non-abusing, we perform a reduction to the timed simulation-soundness of SSCOM. In the reduction, the reduction adversary $\mathcal{A}'$ internally executes an instance of the UC experiment where the legal environment is abusing. We have to ensure that $\mathcal{A}'$ is able to internally simulate the execution without triggering a timeout in the reduction when there is no such timeout triggered by the TLUC environment. This is generally not trivial since the TLUC environment $\mathcal{Z}$ and $\mathcal{A}'$ may count the number of performed computation steps differently. For example, $\mathcal{Z}$ may (legally) not count simulation overhead resulting from e.g. simulating other environments and protocols. Of course, $\mathcal{A}'$ would have to count these steps. To address this caveat, $\mathcal{A}'$ runs, if applicable, the corresponding unrolled TLUC execution (cf. Definition 13) where all steps are counted correctly, including the emulation overhead resulting from $\mathcal{A}'$ internally emulating said execution (cf. Proposition 7).

Also, $\mathcal{Z}$ counts the steps relative to a (presumptive) execution of π_{MCOM} and the dummy adversary $\mathcal{D}$, while $\mathcal{A}'$ emulates an UC execution with an instance of IDEAL($\mathcal{F}_1$) and a simulator $\mathcal{S}_1^i$. For the reduction to go through even in the presence of these discrepancies, the timed security parameter ℓ' of SSCOM has to be chosen sufficiently large to account for these differences.

Let $\text{E}_{\text{ABUSE}_1^i}$ denote the event that the environment is abusing in H_1^i.

Suppose for the sake of contradiction that $\Pr[\mathrm{E}_{\mathrm{ABUSE}_1^i}]$ is non-negligible, i.e. $\mathcal{Z}$ is abusing. Then, we can construct an adversary $\mathcal{A}'$ against the $\ell'(\ell(\kappa),\kappa)$-simulation-soundness of SSCOM using the following reduction:

1. On input $(1^\kappa, z)$, $\mathcal{A}'$ chooses $j \xleftarrow{\$} \{1,\ldots,q\}$ uniformly at random, where $q = q(\kappa)$ is an upper bound for the number of sessions. Then, $\mathcal{A}'$ internally executes an instance of H_1^i. `timer` and `notify` messages are forwarded between the internal execution of the UC experiment and the timed simulation-soundness game. To this end, the timed security parameter ℓ is removed from `timer` messages coming from UC protocol parties and added to `timer` messages for UC parties.
2. The commitments of the sessions that are played using $\mathsf{C}_{\mathrm{trap}}$ in H_1^i are played as left sides with the timed simulation-soundness game, using the uniform distribution on bitstrings of length $l(\kappa)$ resp. 4κ.
3. If all or no parties in the j-th session are corrupted, abort.
4. If there is a corrupted party in the j-th session, play the single SSCOM commitment where the committer is corrupted as the right side in the simulation-soundness experiment.

Clearly, $\mathcal{Z}$'s view is identically distributed in the reduction and in H_1^i, unless all parties in the j-th session are honest or corrupted. If the timed security parameter ℓ' is sufficiently large, $\mathcal{A}'$ never has to send a $(\texttt{notify}, \star, 1)$ message before $\mathcal{Z}$ (when asked through a `notify` message) does. Thus, any attack carried out by $\mathcal{Z}$ can be performed by $\mathcal{A}'$ without having to trigger a timeout prematurely. In particular, $\ell'(\ell(\kappa),\kappa)$ has to be sufficiently large to account for the following differences between the reduction and the presumptive execution according to which $\mathcal{Z}$ counts the number of steps performed:

- Overhead due to the reduction itself, e.g. interaction with the experiment and relaying of messages between the internal UC execution and the experiment.
- Overhead due to changes in the internally emulated UC execution, e.g. additional steps for $\mathrm{IDEAL}(\mathcal{F}_1)$, as well as possible overhead for the trapdoor committer algorithm $\mathsf{C}_{\mathrm{trap}}$. However, if $\mathsf{C}_{\mathrm{trap}}$ requires less steps than C, $\mathcal{A}'$ has to count according to C, as the simulation-soundness experiment requires to count the maximum.

We want ℓ' to not depend on the internally emulated environment $\mathcal{Z}$ or the maximum number of sessions q, but only on the reduction as well as the timeout parameter ℓ[12]. To this end, we observe the following: $\mathcal{A}'$ relays messages between its internal UC execution and the experiment. For each commitment session, the overhead is independent of the number of sessions q.

$\mathcal{Z}$ may schedule the sessions in a way such that when the timer in the j-th session is active, multiple other sessions are activated. In the real execution, unless a timeout occurs, the number of active sessions is trivially upper-bounded

[12] This does of course not rule out that ℓ may depend on $\mathcal{Z}$ and q. However, $\mathcal{Z}$ and q do not have to be *additionally* considered for ℓ'.

by $\ell(\kappa)$, as each activation will consume at least one computation step. Thus, the maximum number of active sessions is independent from the bound for the number of challenge sessions $q(\kappa)$. We have to consider the case that $\mathcal{Z}$ is a routing environment. In this case, the steps needed by $\mathcal{A}'$ to emulate the UC experiment may be more compared to the steps reported by $\mathcal{Z}$ due to unaccounted emulation overhead. However, according to Proposition 7, there exists an "unrolled" execution (cf. Definition 13) where there is no such emulation overhead and the environment's view is identically distributed. Due to the structure of routing environments, $\mathcal{A}'$ is able to "unroll" the execution in polynomial time before starting the internal emulation. At this point, no timers are active. Alternatively, we may provide $\mathcal{A}'$ with the necessary information via its advice.

Thus, setting ℓ' as $\ell(\kappa)$ plus the overhead for one single session plus some additional (small) overhead polynomial in κ, suffices. It follows that ℓ' is polynomially bounded, independent of q and the particular environment. As $\mathcal{A}'$ internally executes an instance of the UC experiment with a legal environment and adversary and timeout parameter ℓ, $\mathcal{A}'$ is also a legal adversary against the $\ell'(\ell(\kappa), \kappa)$-simulation-soundness of SSCOM for appropriately chosen ℓ'. By definition of $\mathcal{A}'$, its advantage in the simulation-soundness experiment is equal to $\Pr[\mathrm{E}_{\mathrm{ABUSE}_1^i}]/q(\kappa)$ in H_1^i. Thus, a non-negligible probability of $\mathrm{E}_{\mathrm{ABUSE}_1^i}$ contradicts the simulation-soundness of SSCOM. It follows that $\Pr[\mathrm{E}_{\mathrm{ABUSE}_1^i}] \leq q(\kappa) \cdot \mathsf{negl}_{\mathrm{SIMSOUND}}(\kappa)$ and that $\mathcal{Z}$ is non-abusing in H_1^i. □

Lemma 2. *If* $\mathsf{PRG.TGen}$ *outputs public keys that are computationally indistinguishable from uniformly random strings of the same length,* PRG *is a PRG and* SSCOM *is extractable, then* out_2^i *and* out_2^{i+1} *are computationally indistinguishable. Moreover, the environment is non-abusing in* H_2^i.

Proof. We start by proving the non-abusing property.

Non-Abusing. In order to show that the environment is non-abusing in H_2^i (the argument for $H_2^{i,\#}$ is analogous), we cannot directly reduce to the simulation-soundness of SSCOM as in the previous lemma. This is because we do not know the correct distribution of the equivocated commitments beforehand, as it depends on the second-round message of the coin-toss.

For $i = 0$, we have shown that the environment is non-abusing in $H_2^0 = H_1^q$. If the environment were to become abusing in H_2^1, we could use the environment to break the assumptions stated in Lemma 2, namely the indistinguishability of keys output by $\mathsf{PRG.TGen}$ (pseudorandom strings) and $\mathsf{PRG.Gen}$ (uniformly random strings) or the PRG property of PRG, respectively. To this end, it is important that the reduction adversary is able to extract commitments created by the environment. While SSCOM is not straight-line extractable, it can be made parallel-extractable via rewinding in strict polynomial-time when using appropriate building blocks (cf. Corollary 5), allowing the reduction adversary to determine if the environment is abusing or not. As the properties we reduce to have non-interactive challenge phases, the reduction adversary is able to execute the extractor without having to rewind the reduction. Thus, a strict

polynomial-time reduction with only a polynomial loss of security is possible. As the aforementioned properties are not timed but hold for polynomial-time adversaries in general, we do not have to argue that the reduction adversary is able to provide its answer in time.

For $i > 1$, we can use this same strategy to create a contradiction to the assumptions of Lemma 2 by using the fact that in the previous hybrid (i.e. H_2^i or $H_2^{i,\#}$), the environment was non-abusing. To this end, the adversary $\mathcal{A}'$ chooses a random index $j \xleftarrow{\$} [q]$ and extracts the environment's commitment in the j-th session. All messages in other threads are answered as usual. If no party is corrupted or the extraction fails, the adversary outputs a random bit b. If the environment is abusing in the j-th session (determined by the same criterion as in Definition 3), it outputs 1, otherwise it outputs 0.

Thus, it holds that $\Pr[\mathrm{E}_{\mathrm{ABUSE}_2^i}] \leq \Pr[\mathrm{E}_{\mathrm{ABUSE}_2^0}] + i \cdot (2 \cdot \mathsf{negl}_{\mathrm{PRGKEY}}(\kappa) + q \cdot \mathsf{negl}_{\mathrm{PRGVAL}}(\kappa) + 2 \cdot \mathsf{negl}_{\mathrm{EXT}}(\kappa))$ for negligible functions $\mathsf{negl}_{\mathrm{PRGKEY}}, \mathsf{negl}_{\mathrm{PRGVAL}}$ and $\mathsf{negl}_{\mathrm{EXT}}$.

Note that in order to show the non-abusing propery, it was not necessary to first show the indistinguishability of the adjacent hybrids. We will do this in the following, making use of the non-abusing property.

Indistinguishability, Corrupted Sender. Suppose for the sake of contradiction that out_2^i and out_2^{i+1} are not computationally indistinguishable. We can construct an adversary against the indistinguishability of uniformly random values and public keys originating from $\mathsf{PRG.TGen}$. First, we consider a sub-hybrid $H_2^{i,\#}$ between H_2^i and H_2^{i+1} where only the first public key pk_0 is replaced with a value originating from $\mathsf{PRG.TGen}$.

The reduction adversary $\mathcal{A}'$ works as follows:

1. On input $(1^\kappa, z)$, internally start an execution of H_2^i on input $(1^\kappa, z)$.
2. Obtain the challenge pk_0 from the experiment and also sample $r \xleftarrow{\$} \{0,1\}^{l(\kappa)}$.
3. Equivocate the commitment in the i-th coin-toss such that it has the result (pk_0, r) and continue the execution of H_i^0 as specified.
4. Output what the environment outputs.

If pk_0 is a uniformly random string, then the environment's view is identically distributed as in H_2^i. If pk_0 originates from $\mathsf{PRG.TGen}$, then the environment's view is identically distributed as in $H_2^{i,\#}$. Thus, the distinguishing advantage of $\mathcal{A}'$ is the same as the environment's, contradicting the indistinguishability of PRG keys with trapdoor and uniformly random strings.

As the proof for the indistinguishability of $H_2^{i,\#}$ and H_2^{i+1} is similar, we omit it. All in all, as PRG keys with and without trapdoor are indistinguishable, we conclude $|\Pr[\mathrm{out}_2^i = 1] - \Pr[\mathrm{out}_2^{i+1} = 1]| = 2 \cdot \mathsf{negl}_{\mathrm{PRGKEY}}(\kappa)$, which is negligible.

Indistinguishability, Corrupted Receiver. Suppose for the sake of contradiction that out_2^i and out_2^{i+1} are not computationally indistinguishable, i.e. there exists a non-negligible function $\nu(\kappa)$ that lower-bounds $\mathcal{Z}$'s distinguishing advantage. We

construct an adversary $\mathcal{A}'$ against the PRG property of PRG with non-negligible advantage $\nu'(\kappa)$.

$\mathcal{A}'$, on input $(1^\kappa, z)$, acts as follows:

1. Obtain the challenge public key pk (which is distributed uniformly at random over $\{0,1\}^{l(\kappa)}$ from the PRG experiment as well as a challenge value s. Also, sample $pk' \xleftarrow{\$} \{0,1\}^{l(\kappa)}$
2. Internally, start an instance of H_2^i with input $(1^\kappa, z)$. At the beginning of the i-th session, play the π_{CT} instance for the extraction trapdoor as follows: First, use the extractor E of SSCOM (cf. Corollary 5) to obtain the value v committed to by the environment. Simulate all protocol messages not related to SSCOM as in H_2^i. If E outputs $v = \perp_E$, output a uniformly random bit to the PRG experiment. Otherwise, send $v \oplus (pk_0, pk_1)$ with $pk_b = pk'$ and $pk_{1-b} = pk$ as second-round message. Let v' denote the value unveiled by the environment. Let $\mathrm{E}_{\mathrm{INCON}}$ denote the event that $v \neq v'$. If $\mathrm{E}_{\mathrm{INCON}}$ occurs, or $\mathcal{Z}$ does not open the commitment, output a uniformly random bit b'.
3. Sample $r \xleftarrow{\$} \{0,1\}^\kappa$ and equivocate the equivocation CRS σ to $\mathsf{PRG}(pk_b, r) \oplus s$.
4. Continue the execution, but commit as follows: If $b = 0$, send $\mathsf{PRG}(pk_0, r)$ as c. If $b = 1$, send $\mathsf{PRG}(pk_1, r) \oplus \sigma = s$ as c.
5. Finally, obtain the output b' of $\mathcal{Z}$ and output b'.

If s is a uniformly random string, then $\mathcal{Z}$'s view is distributed as in H_2^i:

- σ is a uniformly random element (and so is $\sigma \oplus c$).
- If $b = 0$, c is always in the range of $\mathsf{PRG}(pk_0, \cdot)$.
- If $b = 1$, $c \oplus \sigma = s \oplus (\mathsf{PRG}(pk_1, r) \oplus s) = \mathsf{PRG}(pk_1, r)$ is always in the range of $\mathsf{PRG}(pk_1, \cdot)$.

If s is distributed pseudorandomly, i.e. is in the range of $\mathsf{PRG}(pk, \cdot)$, then $\mathcal{Z}$'s view is distributed as in H_2^{i+1}:

- There exists $r' \in \{0,1\}^\kappa$ such that $\sigma = \mathsf{PRG}(pk_b, r) \oplus \mathsf{PRG}(pk, r')$
- If $b = 0$, c is always in the range of $\mathsf{PRG}(pk_0, \cdot)$.
- If $b = 1$, c is always in the range of $\mathsf{PRG}(pk_0, \cdot)$ and $c \oplus \sigma = s \oplus (\mathsf{PRG}(pk_1, r) \oplus s) = \mathsf{PRG}(pk_1, r)$ is always in the range of $\mathsf{PRG}(pk_1, \cdot)$.

If $\mathrm{E}_{\mathrm{INCON}}$ does not occur, $\mathcal{A}'$'s advantage in the PRG game is identical to $\mathcal{Z}$'s distinguishing advantage between H_2^i and H_2^{i+1}. If $\mathrm{E}_{\mathrm{INCON}}$ occurs, $\mathcal{A}'$'s advantage is 0. We can thus use $\mathcal{A}'$ to bound the distinguishing advantage of $\mathcal{Z}$ as follows, using the security of PRG and the extractability of SSCOM:

$$
\begin{aligned}
&|\Pr[\mathrm{out}_2^i = 1] - \Pr[\mathrm{out}_2^{i+1} = 1]| \leq \Pr[\mathrm{E}_{\mathrm{INCON}}] + |\Pr[\mathrm{out}_2^i = 1 \wedge \neg \mathrm{E}_{\mathrm{INCON}}] - \Pr[\mathrm{out}_2^{i+1} = 1 \wedge \neg \mathrm{E}_{\mathrm{INCON}}]| \\
&\leq \Pr[\mathrm{E}_{\mathrm{INCON}}] + |\Pr[\mathrm{out}_{\mathcal{A}'}^{s \text{ is uniformly random}} = 1 \wedge \neg \mathrm{E}_{\mathrm{INCON}}] - \Pr[\mathrm{out}_{\mathcal{A}'}^{s \text{ is a PRG image}} = 1 \wedge \neg \mathrm{E}_{\mathrm{INCON}}]| \\
&\leq \Pr[\mathrm{E}_{\mathrm{INCON}}] + \mathsf{negl}_{\mathrm{PRGVAL}}(\kappa)
\end{aligned}
$$

The last inequality holds due to the security of PRG.

As SSCOM is simulation-sound and extractable with overwhelming probability, we can bound $\Pr[\mathrm{E}_{\mathrm{INCON}}]$ by $\Pr[\mathrm{E}_{\mathrm{ABUSE}_2^i}] + \mathsf{negl}_{\mathrm{EXT}}(\kappa)$ (as the execution is identically distributed to H_2^i when the extraction occurs), where $\mathsf{negl}_{\mathrm{EXT}}(\kappa)$ is a bound for the extraction error. It thus holds that

$$
\begin{aligned}
&|\Pr[\mathrm{out}_2^i = 1] - \Pr[\mathrm{out}_2^{i+1} = 1]| \leq \Pr[\mathrm{E}_{\mathrm{INCON}}] + \mathsf{negl}_{\mathrm{PRGVAL}}(\kappa) \\
&\leq \Pr[\mathrm{E}_{\mathrm{ABUSE}_2^0}] + i \cdot (2 \cdot \mathsf{negl}_{\mathrm{PRGKEY}}(\kappa) + q \cdot \mathsf{negl}_{\mathrm{PRGVAL}}(\kappa) + 2 \cdot \mathsf{negl}_{\mathrm{EXT}}(\kappa)) + \mathsf{negl}_{\mathrm{EXT}}(\kappa) + \mathsf{negl}_{\mathrm{PRGVAL}}(\kappa) \\
&\leq \Pr[\mathrm{E}_{\mathrm{ABUSE}_2^0}] + q \cdot (2 \cdot \mathsf{negl}_{\mathrm{PRGKEY}}(\kappa) + q \cdot \mathsf{negl}_{\mathrm{PRGVAL}}(\kappa) + 2 \cdot \mathsf{negl}_{\mathrm{EXT}}(\kappa)) + \mathsf{negl}_{\mathrm{EXT}}(\kappa) + \mathsf{negl}_{\mathrm{PRGVAL}}(\kappa)
\end{aligned}
$$

which is negligible as the environment is non-abusing in H_2^0.

In particular, it follows that $|\Pr[\mathrm{out}_2^0 = 1] - \Pr[\mathrm{out}_2^q = 1]|$ is negligible too, as there exists a common bound between adjacent hybrids.

□

Lemma 3. *If the environment is non-abusing in H_2^q, then out_3^i and out_3^{i+1} are computationally indistinguishable. Moreover, the environment is non-abusing in H_3^i.*

Proof. As the only possible difference between H_3^i and H_3^{i+1} is the simulator outputting a special error symbol $\perp_{\mathrm{FAIL}}^{i+1}$ if the environment is able to open a commitment of a corrupted sender to a value different than the one extracted by the simulator in the $i+1$-th session, the proof is straight-forward. Let $\mathrm{E}_{\mathrm{FAIL}}^{i+1}$ denote this event.

Non-Abusing. If $\mathcal{Z}$ is non-abusing in H_2^q, then it is also non-abusing in H_3^{i+1}. This is due to the fact that out_3^i and out_3^{i+1} are identically distributed unless $\mathrm{E}_{\mathrm{FAIL}}^{i+1}$ occurs. If $\mathrm{E}_{\mathrm{FAIL}}^{i+1}$ occurs, the execution halts without giving $\mathcal{Z}$ the opportunity to equivocate a commitment that it has not been able to equivocate in the previous hybrid.

Indistinguishability. If $\mathrm{E}_{\mathrm{FAIL}}^{i+1}$ does not occur, H_3^i and H_3^{i+1} are identically distributed. It thus suffices to bound $\Pr[\mathrm{E}_{\mathrm{FAIL}}^{i+1}]$.

Corrupted Sender. If $\mathrm{E}_{\mathrm{FAIL}}^{i+1}$ occurs, then there exist $r_0, r_1 \in \{0,1\}^\kappa$ such that $\sigma = \mathsf{PRG}(pk_0, r_0) \oplus \mathsf{PRG}(pk_1, r_1)$ in the $i+1$-th session, which we call a collision. Let $S = \{\sigma' \mid \exists r_0, r_1 : \sigma' = \mathsf{PRG}(pk_0, r_0) \oplus \mathsf{PRG}(pk_1, r_1)\}$ denote the set of collisions and let $\mathrm{E}_{\mathrm{COL}}$ denote the event that $\sigma \in S$. As the size of PRG's image is upper-bounded by the size of its domain, i.e. 2^κ, we can bound $|S|$ by $2^{2\kappa}$. Thus, in a coin-toss for σ where the commitment is not equivocated and the second-round message is chosen uniformly at random, the probability $\Pr[\mathrm{E}_{\mathrm{COL}}] \leq 2^{2\kappa}/2^{4\kappa}$ is negligible.

In order for $\Pr[\mathrm{E}_{\mathrm{FAIL}}^{i+1}]$ to be non-negligible, the environment must bias the coin-toss either by only finishing sessions where the result is a collision or by equivocating its commitment such that a collision occurs. Let $\mathrm{E}_{\mathrm{INCON}}^{i+1}$ denote the event that the environment equivocates the commitment in π_{CT} in the $i+1$-th

session. We can upper-bound bound $\Pr[\mathrm{E}^{i+1}_{\mathrm{FAIL}}]$ by $\Pr[\mathrm{E}^{i+1}_{\mathrm{COL}} \wedge \neg \mathrm{E}^{i+1}_{\mathrm{INCON}} \vee \mathrm{E}^{i+1}_{\mathrm{INCON}}]$. As $\mathcal{Z}$ is non-abusing in H^i_3, it follows that $\Pr[\mathrm{E}^{i+1}_{\mathrm{INCON}}] \leq \Pr[\mathrm{E}_{\mathrm{ABUSE}^q_2}]$ and is negligible. As $\Pr[\mathrm{E}_{\mathrm{COL}}] \leq 2^{-2\kappa}$, it follows that $\Pr[\mathrm{E}_{\mathrm{COL}} \wedge \neg \mathrm{E}_{\mathrm{INCON}}] \leq 2^{-2\kappa}$. All in all, we have $|[\Pr[\mathrm{out}^i_3 = 1] - \Pr[\mathrm{out}^{i+1}_3 = 1]| = \Pr[\mathrm{E}^{i+1}_{\mathrm{FAIL}}] \leq 2^{-2\kappa} + \Pr[\mathrm{E}_{\mathrm{ABUSE}^q_2}]$, which is negligible as the environment is non-abusing.

As there exists a common bound for the distinguishing advantage between the hybrids H^i_3 and H^{i+1}_3 for all $i \in \{0, \ldots, q-1\}$, it follows that $|\Pr[\mathrm{out}^0_3 = 1] - \Pr[\mathrm{out}^q_3 = 1]$ is negligible.

Corrupted Receiver. If the receiver is corrupted, the simulation is identical to the previous hybrid. Thus, $\mathcal{Z}$'s view is identically distributed and its distinguishing advantage is 0. □

As H_4 is identical to H^q_3, except for syntactical changes (the final simulator outputs $\perp_{\mathrm{FAIL}}$ when the previous one would output $\perp^i_{\mathrm{FAIL}}$) and it holds that $\Pr[\mathrm{out}_4 = 1] = \Pr[\mathrm{out}^q_3 = 1]$.

As there exists a common bound for the distinguishing advantage between the individual hybrids, out_0 and out_4 are computationally indistinguishable and the claim follows. □

Remark 2. Our proof differs from the one in [CF01] in a number of ways:

- In the proof for the $\mathsf{UCC}_{\mathsf{OneTime}}$ scheme, the reduction adversary has to guess the bit committed to by the honest committer when embedding the public key from the PRG reduction into the CRS. In our reduction, this is different as the reduction adversary knows the bit committed to by the honest party beforehand.
- The $\mathsf{UCC}_{\mathsf{OneTime}}$ scheme requires a PRG with a stretch of 3κ that expands strings of length κ to strings of length 4κ. This is due to the fact that the $\mathsf{UCC}_{\mathsf{OneTime}}$ simulator always embeds an equivocation trapdoor into the CRS, even if the receiver is honest. In our case, this is not only impossible if the committer is corrupted, but the analysis is also somewhat different, allowing the use PRGs with a stretch of 2κ only.

7.1 Generalizing Our Result

Our protocol π_{MCOM} uses a timed simulation-sound commitment scheme SSCOM to TLUC-realize $\mathcal{F}_{\mathrm{MCOM}}$. However, it can be generalized to realize $\mathcal{F}_{\mathrm{MCOM}}$ for other security notions, depending on the properties of SSCOM and the existence of certain complexity assumptions.

If one desires a setting where there is no (even temporary) complexity asymmetry between simulator and environment, one can replace SSCOM with a UC-secure commitment scheme and obtains a commitment scheme that UC-realizes $\mathcal{F}_{\mathrm{MCOM}}$. Following the impossibility results of [CF01], this requires a setup.

The other extreme would be to consider large asymmetries that hold throughout the whole execution. This is the general setting of $(\mathcal{C}_{env}, \mathcal{C}_{sim})$-security introduced by [LPV09]. In this setting, $\mathcal{C}_{env}$ denotes the complexity class of the

environment and $\mathcal{C}_{sim}$ the complexity class of the simulator. If $\mathcal{C}_{env} = \mathcal{C}_{sim}$, one achieves a framework with universal composition. In the case of UC security, the complexity class considered is (non-uniform) probabilistic polynomial-time.

If $\mathcal{C}_{env}$ is e.g. (non-uniform) polynomial-time and $\mathcal{C}_{sim}$ is quasi-polynomial time, then we can realize $\mathcal{F}_{\text{MCOM}}$ in the plain model, assuming that trapdoor one-way permutations with dense public description and subexponential hardness exist. Indistinguishability is even guaranteed if the environment passes its view after the execution to a distinguisher with complexity $\mathcal{C}_{sim}$, which is not guaranteed by other approaches.

In between these extremes, more fine-grained solutions are possible, e.g. by considering different security parameters for environment and simulator. In a very practical setting, one might believe that e.g. RSA-4096 is secure even for adversaries able to break RSA-1024. Our protocol can be adapted to this setting, using a simulator that is able to efficiently break RSA-1024 but not RSA-4096.

7.2 Obtaining a Timer-Obeying Simulator Using a Setup

We have stated in Section 4.2 that a potential candidate for the weakly extractable timed commitment scheme TCOM used in SSCOM is the timed commitment scheme of Boneh and Naor [BN00], which is in the plain model.

In order to argue for the plausibility of their plain-model MPC protocol, Prabhakaran and Sahai [PS04] outline how to realize their angel in the $\mathcal{F}_{\text{CRS}}$-hybrid model, essentially resulting in a UC protocol.

We discuss how to cast our protocol π_{MCOM} in the $\mathcal{F}_{\mathsf{KRK}}$-hybrid model when using the timed commitment scheme of [BN00], resulting in a composable commitment that is simulatable even for simulators that obey to timers.

The changes to the commitment scheme of [BN00] are as follows: Instead of having the committer sample primes p_1, p_2 such that $p_1 \equiv p_2 \equiv 3 \mod 4$ and send $N = p_1 \cdot p_2$ to the receiver, we use the $\mathcal{F}_{\mathsf{KRK}}$ functionality to provide a public key N for each committer (to all parties) as well as the secret $\varphi(N)$ (to the committer only) according to the aforementioned distribution.

Informally, the extraction trapdoor $\varphi(N)$ is protected by the BBS assumption, which is believed to be hard for polynomial-time adversaries with only the knowledge of N. Otherwise, the protocol remains unmodified. Each N can even be used for multiple TCOM commitments [BN00], i.e. as sole setup for π_{MCOM}.

The resulting protocol in the $\mathcal{F}_{\mathsf{KRK}}$-hybrid model is straight-line extractable. The new simulator does not have to use the costly (but still polynomial-time) `forced-open` algorithm, but can use $\varphi(N)$ to very efficiently extract timed commitments from malicious senders. Otherwise, the simulation remains unmodified. It is easy to see that the resulting simulator can obey all timers if the timed security parameter ℓ is chosen such that there is sufficient time for the timed commitment to be extracted using $\varphi(N)$ while still guaranteeing its timed hiding property without the knowledge of $\varphi(N)$.

While this modified protocol still crucially relies on timers to prevent the environment from extracting timed commitments created by the simulator and is

thus not UC-secure, the simulation strategy is as "plausible" as any UC simulation strategy relying on a trusted setup.

8 Constant-Round Black-Box Composable General MPC

In order to achieve composable general MPC, we can plug the construction π_{MCOM} into any UC-secure generic MPC protocol in the $\mathcal{F}_{\text{MCOM}}$-hybrid model while maintaining security (using Corollary 3).

Hazay and Venkitasubramaniam [HV15] have presented a constant-round and black-box general MPC protocol in the $\mathcal{F}_{\text{CRS}}$-hybrid model based on public-key encryption and semi-honest oblivious transfer. We can generate the CRS of the [HV15] protocol using a simulatable coin-toss, assuming that trapdoor one-way permutations with dense public description exist, thus casting the protocol in the $\mathcal{F}_{\text{MCOM}}$-hybrid model.

Theorem 7. *Assume that timed commitment schemes and perfectly binding homomorphic commitment schemes exist. Also, assume that enhanced trapdoor one-way permutations with dense public descriptions exist. Then, for every well-formed functionality $\mathcal{F}$, there exists a constant-round black-box protocol $\pi_{\mathcal{F}}^{BB}$ in the plain model such that $\hat{\pi}_{\mathcal{F}}^{BB} \geq_{\text{TLUC}} \text{IDEAL}(\hat{\mathcal{F}})$.*

In Theorem 7, $\hat{\mathcal{F}}$ denotes the multi-session existence of $\mathcal{F}$ (cf. [CR03]) that naturally captures concurrent self-composition.

Considering possible candidates for timed commitments and perfectly binding homomorphic commitment schemes, we obtain the following corollary.

Corollary 4. *Assume that the generalized BBS assumption and the DDH assumption hold and that enhanced trapdoor one-way permutations with dense public description exist. Then, for every well-formed functionality $\mathcal{F}$, there exists a constant-round black-box protocol $\pi_{\mathcal{F}}^{BB}$ in the plain model such that $\hat{\pi}_{\mathcal{F}}^{BB} \geq_{\text{TLUC}} \text{IDEAL}(\hat{\mathcal{F}})$.*

Alternative and more practically efficient constructions for composable generic MPC in the plain model are possible. For example, one can use our composable commitment scheme to bootstrap the CRS of the OT protocol of Peikert, Vaikuntanathan, and Waters [PVW08]. The resulting protocol can then be combined with an arbitrary (efficient) UC-secure generic MPC protocol in the $\mathcal{F}_{\text{OT}}$-hybrid model, resulting in a TLUC-secure generic MPC protocol.

9 Conclusion

We constructed a composable constant-round black-box general MPC protocol in the plain model from standard (timed) assumptions only. In contrast to previous techniques for generic MPC in the plain model, our approach fully fulfills the notion of environmental friendliness.

Looking ahead, it remains to investigate if these results can be obtained more efficiently and from weaker or more generic assumptions and if stronger properties, e.g. with respect to transitivity or composition, can be achieved. With the recent popularity of timed assumptions, it is necessary to define a meaningful extension of environmental friendliness for timed game-based security properties.

References

[Bau+20a] Carsten Baum et al. *CRAFT: Composable Randomness and Almost Fairness from Time*. Cryptology ePrint Archive, Report 2020/784. 2020.

[Bau+20b] Carsten Baum et al. *TARDIS: Time And Relative Delays In Simulation*. Cryptology ePrint Archive, Report 2020/537. 2020.

[Blu81] Manuel Blum. "Coin Flipping by Telephone". In: *CRYPTO'81*. Ed. by Allen Gersho. Vol. ECE Report 82-04. Santa Barbara, CA, USA: U.C. Santa Barbara, Dept. of Elec. and Computer Eng., 1981, pp. 11–15.

[BN00] Dan Boneh and Moni Naor. "Timed Commitments". In: *CRYPTO 2000*. Ed. by Mihir Bellare. Vol. 1880. LNCS. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2000, pp. 236–254.

[Bre+15] Hai Brenner et al. "Fast Non-Malleable Commitments". In: *ACM CCS 2015*. Ed. by Indrajit Ray, Ninghui Li, and Christopher Kruegel. Denver, CO, USA: ACM Press, Oct. 2015, pp. 1048–1057.

[Bro+17] Brandon Broadnax et al. "Concurrently Composable Security with Shielded Super-Polynomial Simulators". In: *EUROCRYPT 2017, Part I*. Ed. by Jean-Sébastien Coron and Jesper Buus Nielsen. Vol. 10210. LNCS. Paris, France: Springer, Heidelberg, Germany, Apr. 2017, pp. 351–381.

[BS05] Boaz Barak and Amit Sahai. "How To Play Almost Any Mental Game Over The Net - Concurrent Composition via Super-Polynomial Simulation". In: *46th FOCS*. Pittsburgh, PA, USA: IEEE Computer Society Press, Oct. 2005, pp. 543–552.

[Can+02] Ran Canetti et al. "Universally composable two-party and multi-party secure computation". In: *34th ACM STOC*. Montréal, Québec, Canada: ACM Press, May 2002, pp. 494–503.

[Can+07] Ran Canetti et al. "Universally Composable Security with Global Setup". In: *TCC 2007*. Ed. by Salil P. Vadhan. Vol. 4392. LNCS. Amsterdam, The Netherlands: Springer, Heidelberg, Germany, Feb. 2007, pp. 61–85.

[Can01] Ran Canetti. "Universally Composable Security: A New Paradigm for Cryptographic Protocols". In: *42nd FOCS*. Las Vegas, NV, USA: IEEE Computer Society Press, Oct. 2001, pp. 136–145.

[CF01] Ran Canetti and Marc Fischlin. "Universally Composable Commitments". In: *CRYPTO 2001*. Ed. by Joe Kilian. Vol. 2139. LNCS.

Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2001, pp. 19–40.

[CKL03] Ran Canetti, Eyal Kushilevitz, and Yehuda Lindell. "On the Limitations of Universally Composable Two-Party Computation without Set-up Assumptions". In: *EUROCRYPT 2003*. Ed. by Eli Biham. Vol. 2656. LNCS. Warsaw, Poland: Springer, Heidelberg, Germany, May 2003, pp. 68–86.

[CLP10] Ran Canetti, Huijia Lin, and Rafael Pass. "Adaptive Hardness and Composable Security in the Plain Model from Standard Assumptions". In: *51st FOCS*. Las Vegas, NV, USA: IEEE Computer Society Press, Oct. 2010, pp. 541–550.

[CLP13] Ran Canetti, Huijia Lin, and Rafael Pass. "From Unprovability to Environmentally Friendly Protocols". In: *54th FOCS*. Berkeley, CA, USA: IEEE Computer Society Press, Oct. 2013, pp. 70–79.

[CR03] Ran Canetti and Tal Rabin. "Universal Composition with Joint State". In: *CRYPTO 2003*. Ed. by Dan Boneh. Vol. 2729. LNCS. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2003, pp. 265–281.

[Dac+13] Dana Dachman-Soled et al. "Adaptive and Concurrent Secure Computation from New Adaptive, Non-malleable Commitments". In: *ASIACRYPT 2013*. Ed. by Kazue Sako and Palash Sarkar. Vol. 8269. Lecture Notes in Computer Science. Springer, 2013, pp. 316–336. URL: https://doi.org/10.1007/978-3-642-42033-7%5C_17.

[DDN00] Danny Dolev, Cynthia Dwork, and Moni Naor. "Nonmalleable Cryptography". In: *SIAM Journal on Computing* 30.2 (2000), pp. 391–437.

[DIO98] Giovanni Di Crescenzo, Yuval Ishai, and Rafail Ostrovsky. "Non-Interactive and Non-Malleable Commitment". In: *30th ACM STOC*. Dallas, TX, USA: ACM Press, May 1998, pp. 141–150.

[DP92] Alfredo De Santis and Giuseppe Persiano. "Zero-Knowledge Proofs of Knowledge Without Interaction (Extended Abstract)". In: *33rd FOCS*. Pittsburgh, PA, USA: IEEE Computer Society Press, Oct. 1992, pp. 427–436.

[DS13] Ivan Damgård and Alessandra Scafuro. "Unconditionally Secure and Universally Composable Commitments from Physical Assumptions". In: *ASIACRYPT 2013, Part II*. Ed. by Kazue Sako and Palash Sarkar. Vol. 8270. LNCS. Bengalore, India: Springer, Heidelberg, Germany, Dec. 2013, pp. 100–119.

[ElG84] Taher ElGamal. "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms". In: *CRYPTO'84*. Ed. by G. R. Blakley and David Chaum. Vol. 196. LNCS. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 1984, pp. 10–18.

[Eph+20] Naomi Ephraim et al. *Non-Malleable Time-Lock Puzzles and Applications*. Tech. rep. 2020.

[Gar+12] Sanjam Garg et al. “Concurrently Secure Computation in Constant Rounds”. In: *EUROCRYPT 2012*. Ed. by David Pointcheval and Thomas Johansson. Vol. 7237. LNCS. Cambridge, UK: Springer, Heidelberg, Germany, Apr. 2012, pp. 99–116.

[GKP18] Sanjam Garg, Susumu Kiyoshima, and Omkant Pandey. “A New Approach to Black-Box Concurrent Secure Computation”. In: *EUROCRYPT 2018, Part II*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Vol. 10821. LNCS. Tel Aviv, Israel: Springer, Heidelberg, Germany, Apr. 2018, pp. 566–599.

[GMY03] Juan A. Garay, Philip D. MacKenzie, and Ke Yang. “Strengthening Zero-Knowledge Protocols Using Signatures”. In: *EUROCRYPT 2003*. Ed. by Eli Biham. Vol. 2656. LNCS. Warsaw, Poland: Springer, Heidelberg, Germany, May 2003, pp. 177–194.

[Gol08] Oded Goldreich. *Computational complexity - a conceptual perspective*. Cambridge University Press, 2008. URL: https://doi.org/10.1017/CBO9780511804106.

[Goy+14] Vipul Goyal et al. “An Algebraic Approach to Non-malleability”. In: *55th FOCS*. Philadelphia, PA, USA: IEEE Computer Society Press, Oct. 2014, pp. 41–50.

[HV15] Carmit Hazay and Muthuramakrishnan Venkitasubramaniam. “On Black-Box Complexity of Universally Composable Security in the CRS Model”. In: *ASIACRYPT 2015, Part II*. Ed. by Tetsu Iwata and Jung Hee Cheon. Vol. 9453. LNCS. Auckland, New Zealand: Springer, Heidelberg, Germany, Nov. 2015, pp. 183–209.

[IPS08] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. “Founding Cryptography on Oblivious Transfer - Efficiently”. In: *CRYPTO 2008*. Ed. by David Wagner. Vol. 5157. LNCS. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2008, pp. 572–591.

[KL11] Dafna Kidron and Yehuda Lindell. “Impossibility Results for Universal Composability in Public-Key Models and with Fixed Inputs”. In: *Journal of Cryptology* 24.3 (July 2011), pp. 517–544.

[KLP05] Yael Tauman Kalai, Yehuda Lindell, and Manoj Prabhakaran. “Concurrent general composition of secure protocols in the timing model”. In: *37th ACM STOC*. Ed. by Harold N. Gabow and Ronald Fagin. Baltimore, MA, USA: ACM Press, May 2005, pp. 644–653.

[KLX20] Jonathan Katz, Julian Loss, and Jiayu Xu. “On the Security of Time-Lock Puzzles and Timed Commitments”. In: *TCC 2020, Part III*. Ed. by Rafael Pass and Krzysztof Pietrzak. Vol. 12552. LNCS. Durham, NC, USA: Springer, Heidelberg, Germany, Nov. 2020, pp. 390–413.

[Lin03] Yehuda Lindell. “General Composition and Universal Composability in Secure Multi-Party Computation”. In: *44th FOCS*. Cambridge, MA, USA: IEEE Computer Society Press, Oct. 2003, pp. 394–403.

[LPV09] Huijia Lin, Rafael Pass, and Muthuramakrishnan Venkitasubramaniam. “A unified framework for concurrent security: universal composability from stand-alone non-malleability”. In: *41st ACM*

STOC. Ed. by Michael Mitzenmacher. Bethesda, MD, USA: ACM Press, May 2009, pp. 179–188.

[MMV13] Mohammad Mahmoody, Tal Moran, and Salil P. Vadhan. "Publicly verifiable proofs of sequential work". In: *ITCS 2013*. Ed. by Robert D. Kleinberg. Berkeley, CA, USA: ACM, Jan. 2013, pp. 373–388.

[MPR06] Silvio Micali, Rafael Pass, and Alon Rosen. "Input-Indistinguishable Computation". In: *47th FOCS*. Berkeley, CA, USA: IEEE Computer Society Press, Oct. 2006, pp. 367–378.

[MY04] Philip D. MacKenzie and Ke Yang. "On Simulation-Sound Trapdoor Commitments". In: *EUROCRYPT 2004*. Ed. by Christian Cachin and Jan Camenisch. Vol. 3027. LNCS. Interlaken, Switzerland: Springer, Heidelberg, Germany, May 2004, pp. 382–400.

[Nao90] Moni Naor. "Bit Commitment Using Pseudo-Randomness". In: *CRYPTO'89*. Ed. by Gilles Brassard. Vol. 435. LNCS. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 1990, pp. 128–136.

[OPV08] Rafail Ostrovsky, Giuseppe Persiano, and Ivan Visconti. *Constant-Round Concurrent Non-Malleable Commitments and Decommitments*. Cryptology ePrint Archive, Report 2008/235. `https://eprint.iacr.org/2008/235`. 2008.

[Pas03] Rafael Pass. "Simulation in Quasi-Polynomial Time, and Its Application to Protocol Composition". In: *EUROCRYPT 2003*. Ed. by Eli Biham. Vol. 2656. LNCS. Warsaw, Poland: Springer, Heidelberg, Germany, May 2003, pp. 160–176.

[PR05] Rafael Pass and Alon Rosen. "New and improved constructions of non-malleable cryptographic protocols". In: *37th ACM STOC*. Ed. by Harold N. Gabow and Ronald Fagin. Baltimore, MA, USA: ACM Press, May 2005, pp. 533–542.

[PR08] Manoj Prabhakaran and Mike Rosulek. "Cryptographic Complexity of Multi-Party Computation Problems: Classifications and Separations". In: *CRYPTO 2008*. Ed. by David Wagner. Vol. 5157. LNCS. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2008, pp. 262–279.

[PS04] Manoj Prabhakaran and Amit Sahai. "New notions of security: Achieving universal composability without trusted setup". In: *36th ACM STOC*. Ed. by László Babai. Chicago, IL, USA: ACM Press, June 2004, pp. 242–251.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. "A Framework for Efficient and Composable Oblivious Transfer". In: *CRYPTO 2008*. Ed. by David Wagner. Vol. 5157. LNCS. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2008, pp. 554–571.

[PW09] Rafael Pass and Hoeteck Wee. "Black-Box Constructions of Two-Party Protocols from One-Way Functions". In: *TCC 2009*. Ed. by Omer Reingold. Vol. 5444. LNCS. Springer, Heidelberg, Germany, Mar. 2009, pp. 403–418.

[RSW96] Ronald L Rivest, Adi Shamir, and David A Wagner. *Time-lock puzzles and timed-release crypto*. 1996.

Appendix

A Definitions

A.1 (Interactive) Turing machines

In the following, we re-visit some notation for interactive Turing machines in the stand-alone setting.

Let A and B be interactive Turing machines. Then $\tau \leftarrow \langle \mathsf{A}(\alpha), \mathsf{B}(\beta)\rangle(\gamma)$ denotes the transcript of the interaction between A and B on private input α resp. β and common input γ. Similarly, $z_{\mathsf{P}} \leftarrow \mathrm{out}_{\mathsf{P}}\langle \mathsf{A}(\alpha), \mathsf{B}(\beta)\rangle(\gamma)$ denotes the output of party $\mathsf{P} \in \{\mathsf{A}, \mathsf{B}\}$. By $(z_{\mathsf{A}}, z_{\mathsf{B}}, \tau) \leftarrow \mathrm{out}\langle \mathsf{A}(\alpha), \mathsf{B}(\beta)\rangle(\gamma)$, we denote the outputs of parties A and B as well as the transcript of their interaction.

A.2 Commitment Schemes

In this section, we provide basic definitions of commitment schemes and their properties. Depending on their use later on, the security definitions are either asymptotic or concrete. If necessary, we consider tag-based commitment schemes (cf. [DDN00]).

Definition 6 (Interactive Commitment Scheme). *An* interactive *commitment scheme* $\mathsf{COM} = \langle \mathsf{C}, \mathsf{R}\rangle$ *with message space* M *is a two-phase protocol between two interactive Turing machines* C *and* R*. Both parties get the phase* `Commit` *or* `Unveil` *as well as the security parameter* κ *as common input. The committer* C *additionaly gets a value* $m \in M$ *as input for the commit phase. We call the transcript*

$$c \leftarrow \langle \mathsf{C}(m), \mathsf{R}(\varepsilon)\rangle(1^{\kappa}, \texttt{Commit})$$

the commitment c *between* C *and* R *(to* $m \in M$*).*

We say that COM *is perfectly correct if for all* $\kappa \in \mathbb{N}$ *and all* $m \in M$*,*

$$\Pr[m' = m \mid (z_{\mathsf{C}}, z_{\mathsf{R}}, c) \leftarrow \mathit{out}\langle \mathsf{C}(m), \mathsf{R}(\varepsilon)\rangle(1^{\kappa}, \texttt{Commit}), m' \leftarrow \mathit{out}_{\mathsf{R}}\langle \mathsf{C}(z_{\mathsf{C}}), \mathsf{R}(z_{Rec})\rangle(\texttt{Unveil})] = 1$$

where the probability is over the random coins used by C *and* R*. Statistical correctness is defined analogously and admits a negligible error probability.*

Definition 7 (Hiding). *For an interactive commitment scheme* $\mathsf{COM} = \langle \mathsf{C}, \mathsf{R}\rangle$*, the hiding experiment is defined as:*

Experiment $\mathrm{Exp}^{\mathit{Hiding}}_{\mathcal{A},\mathsf{COM}}(\kappa, a)$

$(m_0, m_1, \mathit{state}) \leftarrow \mathcal{A}(1^{\kappa}, \texttt{find}, a)$
$b \xleftarrow{\$} \{0,1\}$
if $|m_0| \neq |m_1|$
  return b
$b' \leftarrow \mathit{out}_{\mathcal{A}}\langle \mathsf{C}(m_b), \mathcal{A}(\texttt{guess}, \mathit{state})\rangle(1^{\kappa}, \texttt{Commit})$
return $b = b'$

The advantage of a possibly malicious receiver $\mathcal{A}$ is given by

$$\mathrm{Adv}^{\mathit{Hiding}}_{\mathcal{A},\mathsf{COM}}(\kappa, a) := 2\left|\Pr[\mathrm{Exp}^{\mathit{Hiding}}_{\mathcal{A},\mathsf{COM}}(\kappa, a) = 1] - \frac{1}{2}\right| .$$

The probability is over the randomness of $\mathcal{A}$, R and the choice bit b. An adversary $\mathcal{A}$ is called valid *if $m_0, m_1 \in M$ and $\mathcal{A}$ eventually outputs a single bit. We say that COM is $(\ell(\kappa), \varepsilon(\kappa))$-hiding if $\ell(\kappa)$ is an upper bound for the number of steps performed by $\mathcal{A}$ on input* `guess` *and for all $\kappa \in \mathbb{N}$ and $\ell(\kappa)$-bounded valid $\mathcal{A}$ and for all $a \in \{0,1\}^*$, $\mathrm{Adv}^{\mathit{Hiding}}_{\mathcal{A},\mathsf{COM}}(\kappa, a) \leq \varepsilon(\kappa)$.*

Definition 8 (Binding). *For an interactive commitment scheme $\mathsf{COM} = \langle \mathsf{C}, \mathsf{R} \rangle$, the binding experiment is defined as*

Experiment $\mathrm{Exp}^{\mathit{Binding}}_{\mathcal{A},\mathsf{COM}}\ (\kappa, a)$

$(z_{\mathcal{A}}, z_{\mathsf{R}}, c) \leftarrow \mathit{out}\langle \mathcal{A}(a), \mathsf{R}(\varepsilon)\rangle(1^{\kappa}, \texttt{Commit})$
$m_0 \leftarrow \mathit{out}_{\mathsf{R}}\langle \mathcal{A}(z_{\mathcal{A}}, 0), \mathsf{R}(z_{\mathsf{R}})\rangle(\texttt{Unveil})$
$m_1 \leftarrow \mathit{out}_{\mathsf{R}}\langle \mathcal{A}(z_{\mathcal{A}}, 1), \mathsf{R}(z_{\mathsf{R}})\rangle(\texttt{Unveil})$
return $m_0 \in M \wedge m_1 \in M \wedge m_0 \neq m_1$

where $\mathcal{A}$ and R use the same coins in each run of the unveil phase. The advantage of a possibly malicious committer $\mathcal{A}$ is given by

$$\mathrm{Adv}^{\mathit{Binding}}_{\mathcal{A},\mathsf{COM}}(\kappa, a) := \Pr[\mathrm{Exp}^{\mathit{Binding}}_{\mathcal{A},\mathsf{COM}}(\kappa, a) = 1] .$$

The probability is over the randomness of $\mathcal{A}$ and R. We say that COM is computationally binding if for all PPT adversaries $\mathcal{A}$, there exists a negligible function ε such that for all $\kappa \in \mathbb{N}$ and all $a \in \{0,1\}^*$, $\mathrm{Adv}^{\mathit{Binding}}_{\mathcal{A},\mathsf{COM}}(\kappa, a) \leq \varepsilon(\kappa)$. We say that COM is perfectly binding if for all unbounded adversaries $\mathcal{A}$, it holds that $\mathrm{Adv}^{\mathit{Binding}}_{\mathcal{A},\mathsf{COM}}(\kappa, a) = 0$

Next, we consider a stronger notion of the hiding property called pCCA security. In the pCCA hiding experiment, the adversary may additionally interact with an (inefficient) oracle $\mathcal{O}$ to perform an unbounded number of commitments *in parallel*, with $\mathcal{O}$ acting as receiver. After all commit phases with $\mathcal{O}$ have finished, $\mathcal{O}$ outputs, for each commitment, the unique value committed to. If no such value exists, a special symbol $\perp$ is returned for this commitment. The challenge commitment where the adversary acts as receiver must remain hiding, even with access to $\mathcal{O}$. pCCA security constitutes a stronger variant of parallel one-left many-right non-malleability.

Definition 9 (pCCA security). *For a commitment scheme* $\mathsf{COM} = \langle \mathsf{C}, \mathsf{R} \rangle$*, the pCCA hiding experiment is defined as*

Experiment $\mathrm{Exp}^{pCCA\text{-}Hiding}_{\mathcal{A},\mathsf{COM},\mathcal{O}}(\kappa, a)$

$(m_0, m_1, tag, state) \leftarrow \mathcal{A}^{\mathcal{O}}(1^\kappa, \texttt{find}, a)$
$b \xleftarrow{\$} \{0,1\}$
if $|m_0| \neq |m_1|$
  return b
$b' \leftarrow out_{\mathcal{A}}\langle \mathsf{C}(m_b), \mathcal{A}^{\mathcal{O}}(\texttt{guess}, state)\rangle(\texttt{Commit}, tag)$
return $b = b'$

$\mathcal{O}$ acts as honest receiver R *for multiple sessions* in parallel. *When all commit phases have finished, the oracle returns the unique values committed to. If no such unique value exists, a special symbol $\perp$ is output for these commitments. An adversary $\mathcal{A}$ is valid if it eventually outputs a bit and never interacts with $\mathcal{O}$ on the challenge tag. We say that* COM *is pCCA-secure if for all valid PPT adversaries $\mathcal{A}$, there exists a negligible function* negl *such that for all $\kappa \in \mathbb{N}$ and all $a \in \{0,1\}^*$,*

$$\mathrm{Adv}^{pCCA\text{-}Hiding}_{\mathcal{A},\mathsf{COM}}(\kappa, a) := 2\left|\Pr[\mathrm{Exp}^{pCCA\text{-}Hiding}_{\mathcal{A},\mathsf{COM}}(\kappa, a) = 1] - \frac{1}{2}\right| \leq \mathsf{negl}(\kappa)$$

A.3 A Brief Introduction to Universal Composability

In the following, we give a very brief introduction of the Universal Composability framework due to [Can01] and the corresponding security notion, called UC security. It focuses only on the aspects pertaining to this paper, leaving out other important aspects and details. Unless stated otherwise, this paper uses UC2013.

The Setting. Extending the notion of the real-ideal paradigm, which considers a single protocol execution and non-reactive functionalities only, Universal Composability models the distributed execution of protocols in a network environment by multiple parties. Like in the real-ideal paradigm, the security of protocols is defined through *ideal functionalities* that model a trusted party carrying out some task honestly, e.g. commitments, authenticated message transmission or secure function evaluation. UC security considers the security a protocol not only when a single instance is executed in isolation, but in a setting where multiple, possibly different, protocols are executed concurrently.

The Machine Model. Protocol parties and other entities of the framework are modeled as *interactive Turing machines* that feature several communication tapes. Parties may give or receive input via the *input tape*, receive computation results of subroutines via the *subroutine output tape* and receive general messages (from the adversary) via the *incoming communication tape*. Sending a message is performed by writing the message, along with the recipient's identity, to the

outgoing message tape and issuing a special *external write* message. All entities in the UC execution experiment run in (non-uniform) probabilistic polynomial time.

The Execution Experiment. The UC execution is, apart from the challenge protocol and its parties, defined using two further entities: An adversary that may corrupt parties and, depending on the communication model, alter, inject or drop messages sent between protocol parties via the incoming communication tape. The adversary may freely interact with the environment $\mathcal{Z}$ that (adaptively) provides input to all protocol parties and the adversary. As environment and adversary may freely communicate throughout the execution, the environment is able to incorporate other protocols executed concurrently next to the challenge protocol, capturing the setting of general composition. In order to show the UC security of a protocol, it is thus not necessary to explicitly consider different protocols running alongside as they can, without loss of generality, be considered to be part of the environment.

Protocol Emulation. UC security is based on the notion of protocol emulation: Let π and ϕ be protocols. We say that π *emulates* ϕ (denoted by $\pi \geq_{\mathrm{UC}} \phi$) if, informally, for all adversaries $\mathcal{A}$, there exists a PPT simulator $\mathcal{S}$ such that no environment $\mathcal{Z}$ can distinguish if it interacts with π and $\mathcal{A}$ or with ϕ and $\mathcal{S}$. In contrast to the stand-alone real-ideal paradigm, UC security does not distinguish between a real and an ideal execution experiment but provides a uniform treatment of protocol emulation. To this end, there exists a special class of protocols called *ideal protocols*. For an ideal functionality $\mathcal{F}$, let IDEAL($\mathcal{F}$) denote the corresponding ideal protocol. If $\pi \geq_{\mathrm{UC}} \mathrm{IDEAL}(\mathcal{F})$, we say that π UC-emulates or UC-realizes $\mathcal{F}$.

Properties of UC Security. UC security exhibits several properties such as reflexivity or transitivity and is closed under general composition. Using these properties, one can show that one instance of π UC-realizes a single instance of the protocol ϕ to argue that one may replace all instances of ϕ in a protocol ρ that makes multiple subroutine calls to ϕ (we say that ρ is in the ϕ-hybrid model) with instances of π without losing security. Usually, this is used to conclude that for a protocol ρ in the $\mathcal{F}$-hybrid model that UC-realizes $\mathcal{G}$, one can replace $\mathcal{F}$ by its realization π and the resulting protocol ρ^{π} still UC-realizes $\mathcal{G}$.

As the dummy adversary that reports all messages between protocol parties to the environment and delivers all messages sent from the environment is complete, it is sufficient to show UC security only for this dummy adversary, leading to easier proofs. Also, UC security is transitive, i.e. if $\pi_1 \geq_{\mathrm{UC}} \pi_2$ and $\pi_2 \geq_{\mathrm{UC}} \pi_3$, then $\pi_1 \geq_{\mathrm{UC}} \pi_3$.

Impossibility Results. While UC security is a very strong notion, there exist a number of impossibility results such as the ones of [CF01] that imply that non-trivial functionalities such as the commitment functionality $\mathcal{F}_{\mathrm{COM}}$ can only be realized using some kind of setup, e.g. a common reference string, a public key

infrastructure or a random oracle. Lindell [Lin03] has shown that these impossibilities are not due to the particular definition of UC security, but universally apply to general composition.

Ideal Functionality for Multiple Commitments. The ideal functionality for multiple commitments $\mathcal{F}_{\text{MCOM}}$ in Fig. 1, introduced by [CF01], models ideal bilateral commitments for multiple parties and instances. Individual commitments are distinguished by their commitment ID *cid*.

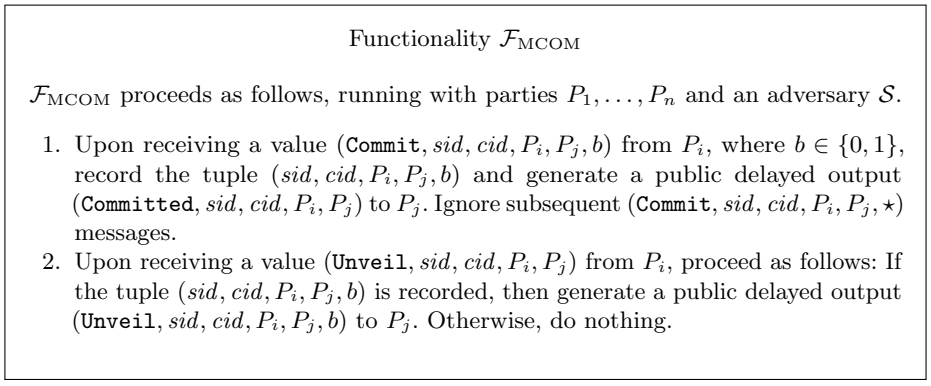

Fig. 1. The ideal commitment functionality for multiple commmitments $\mathcal{F}_{\text{MCOM}}$ (adapted from [CF01])

B Timed Simulation-Sound Commitment Schemes

In the following, we prove the security of the SSCOM commitment scheme.

First, we define oracle trapdoor commitment schemes (Definition 10), which are similar to standard trapdoor commitment schemes except that adversary is additionally given access to an oracle $\mathcal{O}$. Later on, in our constructions, this will be the pCCA oracle for some pCCA-secure commitment scheme.

B.1 Oracle Trapdoor Commitment Schemes

Previous definitions of trapdoor commitments such as the ones of [MY04; GMY03] do not provide the adversary with additional oracles, e.g. a (p)CCA oracle in order to strengthen security. In the following, we define the stronger notion of *oracle trapdoorness*.

Definition 10 (Oracle Trapdoor Commitment Scheme). *A commitment scheme* $\mathsf{TRAPCOM} = \langle \mathsf{C}, \mathsf{R}, \mathsf{C}_{\text{trap}} \rangle$ *is called* oracle trapdoor *for oracle* $\mathcal{O}$ *and message space* M *if* $\langle \mathsf{C}, \mathsf{R} \rangle$ *and* $\langle \mathsf{C}_{\text{trap}}, \mathsf{R} \rangle$ *are commitment schemes with message*

space M such that for all PPT adversaries $\mathcal{A}$, there exists a negligible function negl *such that for all $\kappa \in \mathbb{N}$ and all $z \in \{0,1\}^*$,*

$$\mathrm{Adv}^{OTD}_{\mathcal{A},\mathcal{O},\mathsf{TRAPCOM}}(\kappa, z) := |\Pr[\mathrm{Exp}^{OTD}_{\mathcal{A},\mathcal{O},\mathsf{TRAPCOM}}(\kappa, z) = 1] - \frac{1}{2}| \leq \mathsf{negl}(\kappa)$$

where the probability is over the randomness of $\mathcal{A}$ and the randomness of the OTD experiment. The random variable $\mathrm{Exp}^{OTD}_{\mathcal{A},\mathcal{O},\mathsf{TRAPCOM}}$ is defined as follows:

1. *Sample $b \xleftarrow{\$} \{0,1\}$ uniformly at random.*
2. *Run $\mathcal{A}$ on input $(1^\kappa, z)$. $\mathcal{A}$ interacts with the experiment by first sending $(\mathtt{start}, tag, v)$ to start the commit phase of* TRAPCOM*, acting as receiver. If $b = 0$, the experiment runs the code of the honest committer C on input $(1^\kappa, tag, v)$. If $b = 1$, the experiment runs the code of the trapdoor committer $\mathsf{C}_{\mathrm{trap}}$ on input $(1^\kappa, tag, |v|)$. After the commit phase has finished, the unveil phase is performed. At any time, $\mathcal{A}$ may interact with the (possibly stateful) oracle $\mathcal{O}(\cdot,\cdot)$ which takes a tag as first argument. Finally, $\mathcal{A}$ outputs a bit b'.*
3. *Output b if $\mathcal{A}$ has queried $\mathcal{O}$ with a tag that has the tag of the challenge commitment as prefix, if $v \notin M$ or if $b' \notin \{0,1\}$. Otherwise, output 1 if $b = b'$ and 0 otherwise.*

If $\mathrm{Adv}^{\mathrm{OTD}}_{\mathcal{A},\mathcal{O},\mathsf{TRAPCOM}}(\kappa, z)$ is negligible for unbounded adversaries, then TRAPCOM is called *statistically oracle trapdoor*. If $\mathcal{A}$ is unbounded and its advantage is 0, then TRAPCOM is called *perfectly oracle trapdoor*. A trapdoor commitment scheme TRAPCOM is called *unconditionally oracle trapdoor* if it is perfectly or statistically oracle trapdoor.

In the following, we parameterize oracle trapdoor commitment schemes with additional parameters related to timed security properties. However, as the oracle-trapdoorness is a polynomial-time property, we have chosen to omit these parameters here.

It directly follows that a commitment scheme satisfiying the above notion is also hiding.

B.2 Construction SSCOM

Proof for the Oracle Trapdoorness of SSCOM. We first prove that SSCOM is oracle-trapdoor for the pCCA oracle of $\mathsf{COM}_{\mathsf{pCCA}}$.

Theorem 8. *Let $\mathsf{COM}_{\mathsf{pCCA}}$ be a pCCA-secure and perfectly binding commitment scheme with message space M. Let* TCOM *be a perfectly binding and weakly extractable timed commitment scheme. Then, the commitment scheme* SSCOM *is an oracle-trapdoor commitment scheme with message space M for the pCCA oracle of $\mathsf{COM}_{\mathsf{pCCA}}$.*

Proof. We prove Theorem 8 by a reduction to the pCCA hiding property of $\mathsf{COM}_{\mathsf{pCCA}}$. To this end, we first define a series of hybrids. Let H_i denote the following hybrid:

1. Interact with the adversary $\mathcal{A}$ on input $(1^\kappa, z)$ as in the oracle trapdoorness game. Initially, receive a message $(\texttt{start}, tag, v)$.
2. Create the shares $s_{m,n} \in \{0,1\}^{|v|}$ according to the protocol description of SSCOM. Continue the execution as in the oracle trapdoorness experiment, except with the following changes:
   (a) Perform the commit phase of SSCOM as follows: For $j = 1, \ldots, i$, commit to $s_{j,0}$ using $\mathsf{COM}_{\mathsf{pCCA}}$ for both the commitment with tag $(tag, j, 0)$ and tag $(tag, j, 1)$. For $j > i$, commit to $s_{j,0}$ for the commitment with tag $(tag, j, 0)$ and to $s_{j,1}$ for the commitment with tag $(tag, j, 1)$, i.e. commit honestly.
   (b) At the beginning of the unveil phase, extract the index vector I stored in the timed commitment using the `forced-open` algorithm.
   (c) In the unveil phase, send $s_{m,n}$ for $m > i$. For $m \leq i$, send shares that are consistent with v and I.
3. Continue the internal execution of the oracle trapdoorness game, eventually output what $\mathcal{A}$ outputs.

By definition, H_0 is identical to an execution of the oracle trapdoorness game where $b = 0$, i.e. the commit and unveil phase are performed honestly. Similarly, H_κ is identical to an execution with $b = 1$, i.e. the commit and unveil phase are performed like the trapdoor algorithm would.

Suppose for the sake of contradiction that there exsits an adversary that can distinguish between H_0 and H_κ with non-negligible probability, i.e. $|\Pr[\text{out}_0 = 1] - \Pr[\text{out}_\kappa = 1]| = \nu(\kappa)$ is non-negligible. Then, there exists an index i such that the adversary can distinguish between H_i and H_{i+1} with non-negligible probability of at least $\nu(\kappa)/\kappa$. We show that this contradicts the pCCA hiding property of $\mathsf{COM}_{\mathsf{pCCA}}$.

Lemma 4. *If $\mathsf{COM}_{\mathsf{pCCA}}$ is pCCA-secure, then H_i and H_{i+1} are (computationally) indistinguishable.*

Proof. Suppose for the sake of contradiction that out_i and out_{i+1} are not computationally indistinguishable, i.e. $p := |\Pr[\text{out}_i = 1] - \Pr[\text{out}_{i+1} = 1]|$ is non-negligible. Then, we can construct an adversary $\mathcal{A}'$ against the pCCA-hiding property of $\mathsf{COM}_{\mathsf{pCCA}}$ with non-negligible advantage as follows:

On input $(1^\kappa, z)$, $\mathcal{A}'$ externally runs an instance of the pCCA hiding game. Internally, it runs an instance of the oracle trapdoorness game with adversary $\mathcal{A}$ on input $(1^\kappa, z)$ as follows:

1. Forward oracle queries by $\mathcal{A}$ to the pCCA oracle of the pCCA hiding game.
2. Sample $b' \xleftarrow{\$} \{0,1\}$ uniformly at random.
3. Play H_i honestly, but play the commitment with tag $(tag, i+1, b')$ as follows:
   - Send $(s_{i+1,0}, s_{i+1,1})$ and tag $(tag, i+1, b')$ to the pCCA hiding game as challenge.
   - Perform the commit phase for the commitment with tag $(tag, i+1, b')$ with the pCCA hiding game.

4. Continue the execution of H_i but abort the unveil phase if $I[i+1] = b'$. In this case, send b' to the hiding experiment.
5. Otherwise, continue the execution and output whatever $\mathcal{A}$ outputs.

If $\mathcal{A}'$ does not abort and the pCCA hiding game has the choice bit 0, i.e. commits to $s_{i+1,0}$, then $\mathcal{A}$'s view is distributed as in H_i. Otherwise, it is distributed as in H_{i+1}. Let p denote $\mathcal{A}$'s distinguishing advantage between out_i and out_{i+1}. By definition of $\mathcal{A}'$, its advantage in the pCCA hiding game is $p/2$, which is non-negligible if p is non-negligible, contradicting the pCCA hiding property of $\mathsf{COM}_{\mathsf{pCCA}}$. □

We can conclude that $|\Pr[\text{out}_0 = 1] - \Pr[\text{out}_\kappa = 1]| \leq 2\kappa \cdot \text{Adv}^{\text{pCCA hiding}}_{\mathcal{A}', \mathsf{COM}_{\mathsf{pCCA}}, \mathcal{O}}(\kappa, z)$, which is negligible as $\mathsf{COM}_{\mathsf{pCCA}}$ is pCCA-secure. □

Theorem 9. *Let* SSCOM *be an oracle-trapdoor commitment scheme for the pCCA oracle of* $\mathsf{COM}_{\mathsf{pCCA}}$. *Let* TCOM *be a* $(T, \ell'(\ell(\kappa), \kappa), \mathsf{negl}(\kappa))$*-weakly extractable timed commitment scheme for some polynomially bounded* $T > \ell'(\ell(\kappa), \kappa)$*, sufficiently large* $\ell'(\ell(\kappa), \kappa)$ *and negligible function* negl. *Then,* SSCOM *is an* $\ell(\kappa)$*-simulation-sound commitment scheme.*

Proof for the Timed Simulation-Soundness of SSCOM. In the following, we show that SSCOM (Construction 2) is timed simulation-sound.

Proof. We prove Theorem 9 using a hybrid argument. Let $q = q(\kappa)$ be a bound for the number of left commitment sessions in the timed simulation-soundness experiment.

Let H_i denote the execution of the simulation-soundness experiment where the first i left sessions are equivocated and the remaining sessions are honest, i.e. use the code of the honest committer C instead of C_{trap}. (For the honest sessions, the committer's private input is sampled at commit time.) H_0 refers to an execution where all sessions are honest, H_q to an execution where all sessions are equivocated, i.e. the standard execution of the simulation-soundness experiment.

Let $H_i^\star$ be identical to H_i, but stop the execution before the TCOM commitment to I in the single right session is unveiled. Let $S_{ext} = (s^e_{0,0}, s^e_{0,1}, \ldots, s^e_{\kappa,0}, s^e_{\kappa,1})$ denote the set of shares committed to in the right session. As $\mathsf{COM}_{\mathsf{pCCA}}$ is perfectly binding, S_{ext} is well-defined. Let S denote the set of shares sent in the first-round message of the unveil phase. We say that S is *consistent* relative to I (which is well-defined as TCOM is perfectly binding) if the following holds:

- S is locally consistent, i.e. $s_{0,0} \oplus s_{0,1} = \cdots = s_{\kappa,0} \oplus s_{\kappa,1}$ and
- $s_{0,I[0]} = s^e_{0,I[0]} \wedge \cdots \wedge s_{\kappa,I[\kappa]} = s^e_{\kappa,I[\kappa]}$, i.e. the shares are consistent with commitments that would be unveiled. As TCOM is perfectly binding, I is well-defined.

The adversary wins in $H_i^\star$ if S is consistent relative to I and reconstructs to a different value, i.e. $s^0_{0,0} \oplus s^0_{0,1} \neq s^e_{0,0} \oplus s^e_{0,1}$.

We first show that in H_0, the execution where no commitments are equivocated, the adversary $\mathcal{A}$ is non-abusing, i.e. does not equivocate on the right side. Starting from H_0, we will show that this property is retained in subsequent hybrids. As a non-abusing adversary does not satisfy the winning condition, the claim follows.

Lemma 5. *If* TCOM *is* $(T, \ell'(\ell(\kappa), \kappa), \mathsf{negl})$*-hiding for polynomial* T*, sufficiently large* $\ell'(\ell(\kappa), \kappa)$ *and perfectly binding and if* $\mathsf{COM}_{\mathsf{pCCA}}$ *is perfectly binding, then* $\mathcal{A}$ *is non-abusing in* H_0*.*

Proof. We prove Lemma 5 by a reduction to the timed hiding property of TCOM. As we cannot simulate H_0 in the reduction to the hiding property, we instead show that $\mathcal{A}$ is non-abusing in $H_0^\star$. It then follows that $\mathcal{A}$ is also non-abusing in H_0.

Assume for the sake of contradiction that there exists an adversary $\mathcal{A}$ that is abusing in $H_0^\star$. First, we fix the coins of $\mathcal{A}$ and $H_0^\star$ up to the point before the TCOM commitment starts such that $\mathcal{A}$ has maximum success probability. We also fix $\mathcal{A}$'s advice z. Let $r_\mathcal{A}$ denote these coins of the adversary and r_H denote the coins of the hybrid. As the execution of $H_0^\star$ using these coins is deterministic until the TCOM commitment starts and $\mathsf{COM}_{\mathsf{pCCA}}$ is perfectly binding, the shares $s_{m,n}$ committed to in the right session are well-defined given the coins and the advice. As before, let S_{ext} denote the set containing these shares. We prove Lemma 5 by a reduction to the non-uniform timed hiding property of TCOM. To this end, we construct an adversary $\mathcal{A}'$ against the hiding property of TCOM that is successful if $\mathcal{A}$ is abusing. $\mathcal{A}'$ works as follows:

1. On input $(1^\kappa, (r_\mathcal{A}, r_H, S_{ext}, z))$, internally start an instance of $H_0^\star$ using coins $r_\mathcal{A}$ for the adversary and r_H for $H_0^\star$. Externally, interact with the TCOM hiding experiment. Run the adversary $\mathcal{A}$ on input $(1^\kappa, z)$.
2. Sample $I \xleftarrow{\$} \{0,1\}^\kappa$ and send $(I, \overline{I})$ to the TCOM hiding experiment.
3. Perform the commit phase for TCOM in the right execution with the hiding experiment.
4. Receive the shares S in the unveil phase.
5. If $\mathcal{A}$ has sent a $(\texttt{notify}, id, 1)$ message where id is the tag of the right commitment in either thread, output a random bit b.
6. If S is locally consistent with I and S_{ext}, output 0. If S is locally consistent with $\overline{I}$ and S_{ext}, output 1. Otherwise, output a random bit b.

In order to abuse, $\mathcal{A}$ must send shares in the unveil phase that are consistent with the index vector I or $\overline{I}$ committed to with TCOM as well as the extracted shares. Thus, an abusing $\mathcal{A}$ can be used to distinguish between a commitment to I resp. $\overline{I}$. If $\mathcal{A}$ is unsuccessful, i.e. the sent shares are inconsistent, we output a uniformly random bit.

It remains to show that $\mathcal{A}'$ is $\ell'(\ell(\kappa), \kappa)$-bounded. By assumption, $\mathcal{A}$ is legal. Thus, between the beginning of the timed commitment and the transmission of the shares $s_{m,n}$, the whole experiment, including the adversary $\mathcal{A}$, has spent at most ℓ steps, unless $\mathcal{A}$ has notified the receiver of a timeout. If ℓ' is sufficiently large to account for i) the overhead from internally emulating $H_0^\star$ and $\mathcal{A}$, ii) the

computations performed by $\mathcal{A}'$ after the challenge commitment has been sent and iii) possible (polynomial) speed-up of the internally emulated $\mathcal{A}$ due to changes in the machine model of $\mathcal{A}'$, which may be different, then $\mathcal{A}'$ is legal and its advantage in the hiding game is equal to the probability that $\mathcal{A}'$ is abusing, which contradicts the timed hiding property of TCOM if $\mathcal{A}$ is abusing with non-negligible probability. By definition of $H_0^\star$ and H_0, $\mathcal{A}$ cannot win in H_0 if it cannot win in $H_0^\star$, as $\mathsf{COM}_{\mathsf{pCCA}}$ is perfectly binding. Thus, if $\mathcal{A}$ is non-abusing in $H_0^\star$, then it is also non-abusing in H_0 and the claim follows. □

Remark 3. In order to obtain a uniform reduction, changes to the proof and an additional assumption are necessary. With respect to the assumptions, the pCCA-secure commitment scheme has to be extractable by the reduction adversary, e.g. via rewinding, which is usually the case for non-malleable commitment schemes in the plain model. In particular, the scheme due to Goyal et al. [Goy+14] satisfies this property, regardless of the instantiation of the base commitment. As the commit phases of $\mathsf{COM}_{\mathsf{pCCA}}$ have finished before the challenge phase of the reduction starts, rewinding does not interfere with the reduction.

Also, the timed hiding definition (Definition 7) has to be split in two phases. In a first phase before the actual commit phase starts, the steps performed by the polynomial-time adversary $\mathcal{A}$ are not counted. Second, the share commitments are extracted, e.g. via rewinding, instead of providing their value via the non-uniform advice. In the second phase, the timed commitment is performed and the adversary's steps are counted as usual. As Definition 7 requires security against non-uniform adversaries, any timed commitment scheme secure according to this definition is also secure when considering the outlined variant of the hiding experiment.

Lemma 6. *If $\mathcal{A}$ is non-abusing in H_0 and if SSCOM is an oracle trapdoor commitment scheme for the pCCA oracle of $\mathsf{COM}_{\mathsf{pCCA}}$ and $\mathsf{COM}_{\mathsf{pCCA}}$ is a perfectly binding and pCCA-secure commitment scheme, then $\mathcal{A}$ is non-abusing in H_i for $i > 0$, i.e. $\Pr[out_i = 1] \leq \mathsf{negl}(\kappa)$.*

Proof. We prove Lemma 6 by induction. We have shown that $\mathcal{A}$ is non-abusing in H_0, except with negligible probability. We assume that $\mathcal{A}$ is non-abusing in H_{i-1} for some fixed $i \in [q(\kappa)]$. Then, we can show that $\mathcal{A}$ is also non-abusing in H_i by a reduction to the oracle trapdoor property of SSCOM for the pCCA oracle of $\mathsf{COM}_{\mathsf{pCCA}}$. To this end, we construct an adversary $\mathcal{A}'$ against the oracle trapdoorness of SSCOM as follows:

1. On input $(1^\kappa, z)$, sample a bit $b \xleftarrow{\$} \{0,1\}$ and internally start an execution of H_{i-1}.
2. In the i-th left session, sample $v_i \leftarrow \mathcal{D}_i$ and play the commitment with the oracle trapdoorness experiment on challenge value v_i.
3. Play the single right commitment honestly, but perform the share commitments with the pCCA oracle. After these commit phases have finished, obtain the $s_{m,n}$ from the pCCA oracle. Then, continue the execution.

4. Upon receiving the $s'_{m,n}$ in the right session, check if they are locally consistent and consistent with the index vector I and the extracted shares. If they reconstruct to a different value than the extracted shares, i.e. $\mathcal{A}$ has equivocated the right session, output 1.
5. Otherwise (i.e. if $\mathcal{A}$ has not equivocated), output 0.

Let p_{i-1} resp. p_i denote the probability that $\mathcal{A}$ is abusing in H_{i-1} resp. H_i. The advantage of $\mathcal{A}'$ in the oracle trapdoorness game is $p := |p_i - p_{i-1}|/2$. If p is non-negligible, then $\mathcal{A}'$ wins the trapdoorness game with non-negligible probability, contradicting the oracle trapdoorness of SSCOM. It follows that the advantage of $\mathcal{A}'$ in H_i for $i \geq 1$ can be bounded by $\mathrm{Adv}^{\mathrm{OTD}}_{\mathcal{A},\mathsf{SSCOM},\mathcal{O}}(\kappa)$, which is negligible by assumption. □

All in all, we have $\mathrm{Adv}^{\mathrm{SIMSOUND}}_{\mathcal{A},\mathsf{TRAPCOM},\ell}(\kappa) \leq \mathsf{negl}^{hiding}_{\mathsf{TCOM}}(\kappa) + (q(\kappa)-1) \cdot \mathsf{negl}^{OTD}_{\mathsf{SSCOM},\mathcal{O}}(\kappa)$, which is negligible. □

Looking at the proof of our composable commitment scheme (Lemma 2), we need SSCOM commitments to be (not necessarily straight-line) extractable. The general setting is the following: The environment concurrently receives commitments from the simulator. At the same time, it commits in parallel. In the reduction, the reduction adversary can play all commitments the environment receives as they are unrelated to the reduction. However, it has to extract the parallel commitments by the environment in order to embed a challenge. As the "left sides" can be played honestly and can be rewound, we do not need the strong extractability guaranteed by some non-malleable commitments (e.g. [Goy+14]) where the left sides have to be simulated by the extractor without rewinding them. Thus, extractability similar to the notion defined by Pass and Wee [PW09] is sufficient. We also recall that extractability for one commitment already implies parallel extractability in this setting [PW09]).

Corollary 5. *Let E' be a polynomial-time extractor for $\mathsf{COM}_{\mathsf{pCCA}}$ with overwhelming success probability (over the coins of the extractor and the adversary), then there exists an extractor E for SSCOM with overwhelming success probability (over the coins of the extractor and the adversary).*

Sketch. We prove Corollary 5 by constructing an extractor E for SSCOM as follows. Let E' denote the extractor for $\mathsf{COM}_{\mathsf{pCCA}}$.

- Internally execute E' in parallel and forward all messages related to $\mathsf{COM}_{\mathsf{pCCA}}$ instances where the adversary is the committer between E' and the adversary.
- Execute all instances of $\mathsf{COM}_{\mathsf{pCCA}}$ where the adversary is receiver honestly, also when rewinding.
- Emulate all other messages of SSCOM relative to the current state of the sessions resp. to the protocol description if no such state exists for the current session.
- After the commitment to I has finished, run the `forced-open` algorithm to obtain I.

– Determine the unique value committed to relative to I and the extracted shares and output the reconstructed value. In case of failure, output $\perp$.

□

Depending on the setting, E may also use the code of the trapdoor committer for the left sides.

For example, the pCCA-secure commitment scheme due to Goyal et al. [Goy+14] admits such an extractor E'. In contrast to their construction in the context of non-malleability, our composite extractor honestly answer challenges of concurrently executed commitments where the adversary acts as receiver.

Corollary 5 follows implicitly from the proof of Theorem 9.

C TLUC Security

In the following, we give a full treatment of TLUC security. We re-state the properties of the notion unchanged and present their proofs.

Remark 4. We have chosen to model TLUC security using the mechanisms currently available in the UC framework. However, it is conceivable to achieve a similar notion of security using different mechanisms. To this end, an anonymous reviewer has suggested the use of *shells*, which have seen heavy use in recent iterations of the UC paper. This would necessitate changes to the framework, i.e. the introduction of a shell for the environment, which is currently not available in the UC framework.

Unfortunately, there is no indication that resorting to shells would lead to easier definition or improve the notion's properties. Looking ahead, composition is limited because environments cannot internally execute simulators that break timed assumptions without affecting the number of counted steps. In any setting where the steps of the environment are counted correctly, this would affect timed assumptions of the challenge protocol, leading to limited composition.

Legal Adversaries. We first define *legal* adversaries, i.e. adversaries that correctly handle timer-related messages:

Definition 11 (Legal Adversaries). *An adversary $\mathcal{A}$ is called* legal *if*

1. *upon receiving* $(\texttt{timer}, \mu, id, t)$ *from some ITI with extended identity μ via the incoming communication tape, $\mathcal{A}$ immediately sends* $(\texttt{timeout}, \mu, id, t)$ *to the environment.*
2. *it does not sends messages* $(\texttt{timer}, \mu, id, t)$ *to the environment if the party with extended ID μ is honest and has not sent* $(\texttt{timer}, \mu, id, t)$ *to the adversary.*
3. *upon receiving* $(\texttt{notify}, \mu, id)$ *from some ITI with extended identity μ via the incoming communication tape, $\mathcal{A}$ immediately sends* $(\texttt{notify}, \mu, id)$ *to the environment.*
4. *upon receiving* $(\texttt{notify}, \mu, id, b)$ *from the environment, $\mathcal{A}$ immediately writes* $(\texttt{notify}, \mu, id, b)$ *on the incoming communication tape of the ITI with extended identity μ.*

Counting Computational Steps. When a honest party P sets up a timer with timeout ℓ, we want P to be able to learn when the timer has expired. To this end, it is first necessary to define how computation steps are counted against the timer by the environment, which is ultimately responsible to signal if a timeout has occurred.

Suppose that we want to prove that a protocol π making use of a timed assumption emulates some protocol ϕ. At some point in the reduction, we may have to construct a stand-alone adversary $\mathcal{A}'$ against the timed assumption that incorporates the whole TLUC execution and uses a distinguishing environment to contradict the security of the timed assumption. In this situation, the counting of steps by $\mathcal{Z}$ and by $\mathcal{A}'$ has to be compatible in the sense that if $\mathcal{Z}$ never triggers a timeout, neither does $\mathcal{A}'$.

While the intuition is clear, the definition must account for the fact that adversary and protocol under consideration might change, e.g. when considering protocol emulation. To this end, we explicitly parameterize environments with a protocol and adversary to make clear relative to which ones the number of performed steps is counted. Let $\mathcal{Z}[\pi, \mathcal{A}]$ denote the environment $\mathcal{Z}$ that expects to interact with protocol π and adversary $\mathcal{A}$ and counts its steps accordingly.

Note that an environment may not have enough information to precisely calculate the correct number of steps performed by other entities, e.g. if the protocol is probabilistic. Considering the security guarantee we want to capture, this is no problem as long as the environment does not *under-estimate* the number of steps performed. In particular, security is not affected if the number of steps performed is estimated too high, leading to a timeout triggered too early. This is accounted for in Definition 14.

We also require environments to perform the calculation independent of the protocol parties' inputs and outputs. Otherwise, we would introduce a side-channel (in the ideal execution) which might help the adversary to learn a party's secrets by observing if timeouts are triggered or not. In this setting, one can design clearly insecure protocols that e.g. realize the ideal commitment functionality $\mathcal{F}_{\text{COM}}$ and prove their security in TLUC. For discussion, see Appendix C.4.

Routing Environments. Before defining legal environments, we look ahead to the proofs of various framework properties. In the proof of the completeness of the dummy adversary (Proposition 9), an environments $\mathcal{Z}_{\mathcal{D}}$ interacting with a protocol π and the dummy adversary $\mathcal{D}$ internally runs an adversary $\mathcal{A}$ and a legal environment $\mathcal{Z}$ and appropriately routes all messages. As $\mathcal{Z}_{\mathcal{D}}$ does not perform any meaningful computations on its own but only routes messages, we call it a *routing environment*. In the proof, we have to show that $\mathcal{Z}_{\mathcal{D}}$ is a legal environment. Clearly, the number of steps performed in the interaction of $\mathcal{Z}_{\mathcal{D}}$ with $\mathcal{D}$ is greater than the number of steps in the execution of $\mathcal{Z}$ with $\mathcal{A}$. As $\mathcal{Z}$ is oblivious of the fact that it is emulated, it cannot possibly account for this difference. In particular, in the "outer" execution with $\mathcal{Z}_{\mathcal{D}}$, additional steps are performed by $\mathcal{Z}_{\mathcal{D}}$ due to emulation overhead as well as due to the additional steps performed by the dummy adversary $\mathcal{D}$. Still, we want both executions to count the same number of computation steps. Otherwise, the view of the "inner"

environment $\mathcal{Z}$ would not be identically distributed when emulated or $\mathcal{Z}_{\mathcal{D}}$ would not be a legal environment.

In the proof of the single-instance composition theorem (Theorem 10), a similar situation arises where an environment $\mathcal{Z}'$ internally emulates an environment $\mathcal{Z}$ and a protocol ρ.

In the following, we thus define the class of routing environments, which are allowed to count steps identically to their internally emulated environment (see Definition 14), even though more steps are actually performed. This is justified as in every execution with a routing environment, there exists a corresponding "unrolled" execution without routing environments (see Definition 13) for which the number of counted steps is correct. This "unrolled" execution can then be used e.g. in a reduction.

Definition 12 (Routing Environment). *$\mathcal{Z}'[\pi, \mathcal{D}]$ is a* type-1 routing environment *if it expects to interact with challenge protocol π and the dummy adversary $\mathcal{D}$ and internally emulates a non-routing environment $\mathcal{Z}[\pi, \mathcal{A}]$ and adversary $\mathcal{A}$ that is not the dummy adversary, and routes messages as follows:*

- *Outputs from the challenge protocol to the environment $\mathcal{Z}'$ are forwarded as outputs to the internal environment $\mathcal{Z}$.*
- *Inputs from the internal environment $\mathcal{Z}$ to the challenge protocol are forwarded to the challenge protocol as inputs of $\mathcal{Z}'$.*
- *Inputs from the internal environment $\mathcal{Z}$ to the adversary $\mathcal{A}$ are forwarded to the internal emulation of $\mathcal{A}$.*
- *Messages from the internal adversary $\mathcal{A}$ to the environment $\mathcal{Z}$ are forwarded to the internal environment $\mathcal{Z}$.*
- *Messages from the internal adversary $\mathcal{A}$ to the challenge protocol are forwarded to the external adversary as coming from $\mathcal{Z}'$.*
- *Messages from the adversary to the environment $\mathcal{Z}'$ are forwarded to the internal adversary $\mathcal{A}$ as coming from the challenge protocol π.*
- *Eventually, $\mathcal{Z}'$ outputs what $\mathcal{Z}$ outputs.*

$\mathcal{Z}'[\pi, \mathcal{D}]$ is a type-2 routing environment *if it expects to interact with a challenge protocol π and the dummy adversary $\mathcal{D}$, internally emulates an environment $\mathcal{Z}[\rho^{\pi}, \mathcal{D}]$ that is either a non-routing environment or a type-2 environment and protocol ρ that makes one subroutine call to π and routes messages as follows:*

- *Outputs from the challenge protocol to the environment $\mathcal{Z}'$ are forwarded as output to the appropriate party of ρ coming from π.*
- *Inputs from the protocol ρ to π are forwarded to the challenge protocol as input coming from $\mathcal{Z}'$.*
- *Inputs from $\mathcal{Z}$ to the adversary pertaining ρ are forwarded to ρ as coming from the dummy adversary $\mathcal{D}$.*
- *Inputs from $\mathcal{Z}$ to the adversary pertaining π are forwarded to the adversary as coming from the environment $\mathcal{Z}'$*
- *Messages from ρ to the adversary are forwarded as input to $\mathcal{Z}$ as coming from the dummy adversary $\mathcal{D}$.*

– *Messages from the adversary to $\mathcal{Z}'$ are forwarded to $\mathcal{Z}$.*
– *Eventually, $\mathcal{Z}'$ output what $\mathcal{Z}$ outputs.*

We will make use of type-1 environments in the proof of the completeness of the dummy adversary (Proposition 9). Type-2 routing environments will be used in the proof of the single-instance composition theorem (Theorem 10).

As TLUC inherits the notion of polynomial time of the UC framework, routing environments are inherently polynomially bounded. Thus, the nesting depth of routing environments is also polynomially bounded. Also, there always exists an innermost non-routing environment. As routing environments only route messages between the outside and internally emulated entities, there exists an "unrolled" execution without routing environments. In the unrolled execution, the innermost environment interacts with the actual challenge protocol (which may have been split into several parts hosted by different routing environments before) and an adversary which may not be the dummy adversary. We formally define the unrolled execution as follows:

Definition 13 (Unrolled Execution). *Let $\mathcal{Z}_k$ be a routing environment expecting to interact with protocol π_k and adversary $\mathcal{D}$ such that $\mathcal{Z}_k$ (transitively) incorporates k environments such that the innermost environment $\mathcal{Z}$, expecting to interact with adversary $\mathcal{A}$ and protocol $\rho^{\pi_1 \cdots \pi_k}$ (if $\mathcal{A} = \mathcal{D}$) resp. $\rho^{\pi_2 \cdots \pi_k}$ (if $\mathcal{A} \neq \mathcal{D}$), is no routing environment. For $i = 2, \ldots, k-1$, let $\mathcal{Z}_i$ denote the i-th routing environment (of type 2) that is emulated by the $i+1$-th routing environment, along with the protocol π_i. $\mathcal{Z}_i$ expects to interact with the protocol $\pi_i^{\cdots \pi_k}$ and the adversary $\mathcal{D}$. If $\mathcal{Z}_1$ is a type-1 routing environment, it expects to interact with $\rho^{\pi_2 \cdots \pi_k}$ and internally runs $\mathcal{Z}$ and $\mathcal{A}$. If $\mathcal{Z}_1$ is a type-2 routing environment, it expects to interact with $\pi_2 \cdots \pi_k$ and $\mathcal{D}$ and internally runs $\mathcal{Z}$ and π_1. The unrolled execution of $\mathcal{Z}_k$ is defined as the execution of $\mathcal{Z}$ interacting with the protocol $\rho^{\pi_2 \cdots \pi_k}$ resp. $\rho^{\pi_1 \cdots \pi_k}$ and the adversary $\mathcal{A}$.*

Legal Environments. We are now ready to define the class of *legal environments*, namely environments that correctly handle timers set up by honest parties.

Definition 14 (Legal Environment). *An environment $\mathcal{Z}$ expecting to interact with protocol π and adversary $\mathcal{A}$, denoted by $\mathcal{Z}[\pi, \mathcal{A}]$, is called* legal *if upon receiving $(\texttt{notify}, \mu, id)$ from the adversary, it immediately sends*

– $(\texttt{notify}, \mu, id, 0)$ *to the adversary if the party with extended identity μ is honest and the (presumptive) execution of the UC execution experiment with π and $\mathcal{A}$ will have performed less than ℓ steps since*
  - *$\mathcal{Z}$ has received $(\texttt{timer}, \mu, id, \ell)$ from the adversary and*
  - *$\mathcal{A}$ has written $(\texttt{notify}, \mu, id, \star)$ to the incoming communication tape of μ,*

  subject to the rules below[13]*. The calculation of all steps is performed obliviously of the protocol parties' inputs and relative to the first message $(\texttt{timer}, \mu, id, \star)$ from μ with ID id.*

[13] It is possible to admit a negligible error to these calculation to get a definition that captures more "intuitively secure" protocols.

– $(\texttt{notify}, \mu, id, 1)$ *to the adversary otherwise.*

Computation steps are counted as follows:

– *If* $\mathcal{Z}[\pi, \mathcal{A}]$ *is a non-routing environment,* $\mathcal{Z}[\pi, \mathcal{A}]$ *counts the steps as if the whole execution experiment were emulated by a single Turing machine.*
– *If* $\mathcal{Z}[\pi, \mathcal{D}]$ *is a routing environment,* $\mathcal{Z}[\pi, \mathcal{D}]$ *counts the steps as if the whole unrolled experiment (cf. Definition 13) were emulated by a single Turing machine.*

Note that while we specify that computation steps have to be counted as if the whole TLUC experiment were executed on a single Turing machine, we do not fix the concrete machine model (e.g. the alphabet or the number of tapes). The actual machine model we are interested in depends on the timing assumption used in the protocol π (if there are any) such that the environment $\mathcal{Z}$ always counts the correct number of steps if some reduction adversary $\mathcal{A}'$ internally emulates the TLUC execution experiment.

Proposition 7 (Properties of Routing Environments). *Let* $\mathcal{Z}$ *be the outermost legal non-routing environment that is (transitively) emulated by a routing environment* $\mathcal{Z}'$*. Then,*

– $\mathcal{Z}'$ *is a legal environment,*
– *the number of steps counted by* $\mathcal{Z}$ *is correct for the unrolled execution run on a single Turing machine and*
– *the view of* $\mathcal{Z}$ *in the unrolled execution is identically distributed as when (transitively) emulated by* $\mathcal{Z}'$*.*

Proof. We prove Proposition 7 by induction over the nesting depth. Let $\mathcal{Z}$ denote the outermost legal non-routing environment.

If $\mathcal{Z}$ expects to interact with an adversary $\mathcal{A}$ that is not the dummy adversary $\mathcal{D}$, then $\mathcal{Z}$ is internally executed by a type-1 routing environment that expects to interact with the same protocol and the dummy adversary and internally emulates $\mathcal{Z}$ and the adversary $\mathcal{A}$. Let $\mathcal{Z}_1$ denote this environment, where the index denotes the nesting depth. By Definition 12, $\mathcal{Z}_1$ relays all `timer` and `notify` messages between $\mathcal{Z}$, $\mathcal{A}$ and the external adversary. Thus, `notify` messages are handled according to the calculation of $\mathcal{Z}$. In order for $\mathcal{Z}_1$ to be legal, it suffices to show that the steps performed by the "unrolled" execution experiment are identical to the steps performed by the execution considered by $\mathcal{Z}$. As the unrolled execution is the very execution $\mathcal{Z}$ expects, i.e. the interaction between $\mathcal{Z}$, the protocol π and the adversary $\mathcal{A}$, it holds that the number of steps calculated by $\mathcal{Z}$ are also valid for $\mathcal{Z}_1$.

If $\mathcal{Z}$ expects to interact with the dummy adversary $\mathcal{D}$ and a protocol ρ^{π_1}, then $\mathcal{Z}$ is emulated by a type-2 routing environment $\mathcal{Z}_1$. $\mathcal{Z}_1$ internally executes $\mathcal{Z}$ and a protocol ρ, while externally expecting to interact with a protocol π_1 such that ρ makes exactly one subroutine call to π_1. Also, $\mathcal{Z}_1$ expects to interact with the dummy adversary. Again, $\mathcal{Z}_1$ relays all `timer` and `notify` messages. In the unrolled execution, $\mathcal{Z}$ interacts with the composed protocol ρ^{π_1} and the

dummy adversary, which is the very execution $\mathcal{Z}$ considers when counting the steps performed by the execution experiment. Thus, $\mathcal{Z}_1$ notifies timers correctly, making it a legal environment. Also, the number of steps counted by $\mathcal{Z}$ is correct for the unrolled execution.

In both cases, it follows from the definition of routing environments and the unrolled execution that $\mathcal{Z}$'s view in the unrolled execution is identically distributed and that the number of steps calculated by $\mathcal{Z}$ is also correct for the unrolled execution run on a single Turing machine.

Suppose that we have shown Proposition 7 to hold for an arbitrary but fixed nesting of depth $k \geq 1$. We show that it also holds for nestings of depth $k+1$. As $k \geq 1$, we know that $\mathcal{Z}_{k+1}$ is a type-2 routing environment and expects to interact with the dummy adversary. $\mathcal{Z}_{k+1}$ expects to externally interact with a protocol π_{k+1} and the dummy adversary. Internally, it emulates a routing environment $\mathcal{Z}_k[\pi_k^{\pi_{k+1}}, \mathcal{D}]$ or $\mathcal{Z}_k[\pi_{k+1}, \mathcal{D}]$ (if $k = 1$ and $\mathcal{Z}_k$ is of type 1). By using the same argument as in the base case, the claim for $k \geq 1$ follows. □

We are now ready to define TLUC protocol emulation. To this end, we only consider legal environments and adversaries. Otherwise, the definition of protocol emulation is equivalent to the standard notion of UC protocol emulation.

C.1 Distinguishing Between UC and TLUC Protocols

To facilitate an easier discussion, we would like to clearly and syntactically distinguish between UC and TLUC protocols where former are protocols that *do not* use timers and the latter are protocols that *may* use timers. As TLUC security is cast in the unmodified UC framework, such a syntactic distinction is impossible. In particular, UC protocols can use the same messages that TLUC protocols use to set up and check timers. While they have no meaning in the UC framework in the sense that a honest party can rely on them being handled in any particular way, we cannot rule out that the `timer` and `notify` messages are used UC protocols, too.

Still, we want to use the following *informal* distinction in the context of protocol emulation: Let π_1 and π_2 be protocols. If π_1 (non-trivially) TLUC-emulates π_2 but π_1 does not (non-trivially) UC-emulate π_2, then we consider π_1 not to be a UC protocol. If π_1 (non-trivially) UC-emulates the ideal protocol of some non-trivial functionality $\mathcal{F}$, then π_1 is always considered to be a UC protocol: While such a π_1 may send `timer` and `notify` messages, the definition of UC emulation makes no guarantee on how these messages are handled. Thus, intuitively, the security of π_1 cannot depend on timers being handled correctly. Of course, every UC protocol is also a TLUC protocol (but not vice versa).

In order to allow a clear and explicit distinction, a possible solution is to augment the model of interactive Turing machines to include a new *timeout tape* that handles timer-related messages. As a timeout tape does not exist in the UC framework, UC protocols never read its contents or write messages on it. In such a model, we can syntactically recognize TLUC protocols that are not UC protocols. However, we stress that this is not formally necessary and do not

assume such a change in the following. Alternatively, one could *assume* that UC protocols never use `timer` and `notify` (or some other unique) messages. However, as stated before, this not formally justified.

C.2 Protocol Emulation

We define TLUC emulation in analogy to UC emulation.

Definition 15 (TLUC Emulation). *Let π and ϕ be protocols. We say that π TLUC-emulates ϕ if for all legal PPT adversaries $\mathcal{A}$, there exists a PPT simulator $\mathcal{S}$ such that for all legal PPT environments $\mathcal{Z}[\pi, \mathcal{A}]$ there exists a negligible function* negl *such that for all $\kappa \in \mathbb{N}, a \in \{0,1\}^*$ it holds that*

$$|\Pr[\mathsf{Exec}\big(\pi, \mathcal{A}, \mathcal{Z}[\pi, \mathcal{A}]\big)(\kappa, a) = 1] - \Pr[\mathsf{Exec}\big(\phi, \mathcal{S}, \mathcal{Z}[\pi, \mathcal{A}]\big)(\kappa, a) = 1]| \leq \mathsf{negl}(\kappa)$$

If π TLUC-emulates ϕ, we write $\pi \geq_{\mathrm{TLUC}} \phi$. When omitting the non-uniform input a, the notion of protocol emulation is uniform.

Note that in Definition 15, the environment $\mathcal{Z}$ is supposed to count the steps according to the execution with π and $\mathcal{A}$ even if it is actually interacting with ϕ and $\mathcal{S}$. This allows the PPT-bounded simulator $\mathcal{S}$ to perform more steps than the adversary $\mathcal{A}$ without triggering a time-out, allowing it to break timed assumptions. If ϕ is an UC protocol, its security is not affected by such a powerful simulator. In contrast, if ϕ is a protocol making use of timers, honest parties of the protocol ϕ may not rely on timing assumptions as the adversary $\mathcal{S}$ is allowed to violate them unnoticed.

Meaningfulness of TLUC Security. When introducing a new security notion, it is important to argue that it does not allow to prove the security of "obviously" insecure protocols. The basic idea behind TLUC security is the very same as behind established simulation-based security notions, where a protocol's security is defined through the ideal functionality it realizes. For simulation-based security notions, care has to be taken that the simulator's capabilities do not affect the security guarantees of the ideal functionality. For example, SPS security is not meaningful for ideal functionalities that use a polynomial-time hardness assumption like a signature scheme that can be broken by the super-polynomial simulator. As the TLUC simulation is always polynomial-time, is does not affect an ideal functionality that makes use of computational assumptions. What is more, we show that non-trivial functionalities can be realized using a *uniform* polynomial-time simulation.

In total analogy to both UC security and other composable security notions that admit general MPC in the plain model, we can show strong impossibility results. This underlines that the new mechanism of timers does not help the simulator *per se*.

Plausibility of TLUC Simulation. Canetti et al. [Can+07] have raised the question of *simulation plausibility* with respect to different simulation strategies and deniability. They state that "if the resources required to simulate a protocol

session are readily available, then we say the protocol session is *plausibly deniable* (since it is plausible that information obtained from the protocol was the result of a simulation). If the resources required to simulate are difficult or impossible to obtain, then there is no guarantee of plausible deniability (since it will be difficult to convince others that an incriminating protocol transcript was the result of a simulation)" [Can+07]. In standard UC, the existence of a simulation strategy is "plausible" as in the presence of a trusted setup, the task of simulation does not asymptotically require any computational power that other entities are not capable of. In particular, protocol parties can provide "fake" transcripts of interactions that cannot be verified by other parties that do not have access to the setup used in the transcript. (Still, UC security does not rule out that a simulator has to perform an "efficient" computation requiring $\Theta(\kappa^{100})$ steps, while all other entities have run-time in e.g. $\Theta(\kappa)$ with a small constant factor only.)

In contrast, when considering e.g. SPS security, the simulator is required to have computational power (i.e. *online* super-polynomial computation capabilities) that is not believed to exist and which are not admitted to other entities such as protocol parties or the environment. In this setting, a protocol transcript is meaningful to other parties that have not participated in the protocol execution in question: Either, the transcript is the result of a real execution or its creation required super-polynomial powers, which are not believed to be available. Thus, a protocol participation cannot be plausibly denied.

In advanced settings such as the Angel-based security framework, the question of simulation plausibility is harder to answer as the power available to the simulator is less clear-cut. (Moreover, a complexity advantage can often be traded for the ability to rewind, e.g. in [CLP10; CLP13].)) Indeed, as Canetti et al. [Can+07] argue, if the angel of [PS04] "were to somehow be made practical in the real world, all security would be lost", making the simulation implausible. Still, Prabhakaran and Sahai [PS04] have argued the plausibility of their angel by showing that it can be realized in the $\mathcal{F}_{\text{CRS}}$-hybrid model from one-way functions. We show a similar result for our composable commitment scheme in Section 7.2.

In TLUC, we believe that the plausibility of the simulation depends largely on the size of the (only temporary) gap between simulator and environment. If this gap is very large, i.e. if the simulator has to perform many more computation steps than the environment is allowed to while a timer is active, simulation may be less plausible than in a setting where this gap is very small, consisting only of very few steps. To some extent, this can be adjusted via a protocol's parameters and the underlying timed assumption. However, the asymmetry can always be "caught up" by a polynomial-time protocol party—allowing it to fake a transcript given "enough time" to internally execute the simulator. In any case, TLUC simulators are asymptotically as efficient as UC simulators and their existence would *never* endanger other polynomial-time hardness assumptions.

C.3 Properties of TLUC Security

Having defined protocol emulation, we can state important properties of TLUC security in analogy to properties of UC security.

Proposition 8. *The dummy adversary $\mathcal{D}$ is legal.*

Proposition 8 immediately follows from the definition of the dummy adversary in the UC framework.

Proposition 9 (Completeness of the Dummy Adversary). *Let π and ϕ be protocols. Then, $\pi \geq_{\mathrm{TLUC}} \phi$ if and only if π TLUC-emulates ϕ with respect to the dummy adversary.*

Proof. Clearly, if π TLUC-emulates ϕ, then π TLUC-emulates ϕ with respect to the dummy adversary. We now show the converse. Let π TLUC-emulate ϕ with respect to the dummy adversary, i.e there exists a simulator $\mathcal{S}_{\mathcal{D}}$ such that for all legal environments $\mathcal{Z}[\pi, \mathcal{D}]$, it holds that

$$\mathsf{Exec}\big(\pi, \mathcal{D}, \mathcal{Z}[\pi, \mathcal{D}]\big) \approx \mathsf{Exec}\big(\phi, \mathcal{S}_{\mathcal{D}}, \mathcal{Z}[\pi, \mathcal{D}]\big) \tag{1}$$

Consider the type-1 routing environment (cf. Definition 12) $\mathcal{Z}_{\mathcal{D}}$ that expects to interact with the dummy adversary $\mathcal{D}$ and protocol π as the environment that internally runs $\mathcal{Z}[\pi, \mathcal{A}]$ and $\mathcal{A}$ and relays all messages between (its internal simulation of) $\mathcal{A}$ and the dummy adversary $\mathcal{D}$ as well as (its internal simulation of) $\mathcal{Z}[\pi, \mathcal{A}]$ and the challenge protocol π. Eventually, $\mathcal{Z}_{\mathcal{D}}$ outputs what $\mathcal{Z}$ outputs.

By Proposition 7, $\mathcal{Z}_{\mathcal{D}}$ is legal. Also, the views of $\mathcal{Z}$ and $\mathcal{A}$ remain identically distributed and timers set up in $\mathsf{Exec}\big(\pi, \mathcal{A}, \mathcal{Z}[\pi, \mathcal{A}]\big)$ are handled identically to timers set up in $\mathsf{Exec}\big(\pi, \mathcal{D}, \mathcal{Z}_{\mathcal{D}}\big[\pi, \mathcal{D}]\big)$ and vice versa.

It follows that

$$\mathsf{Exec}\big(\pi, \mathcal{A}, \mathcal{Z}[\pi, \mathcal{A}]\big) \equiv \mathsf{Exec}\big(\pi, \mathcal{D}, \mathcal{Z}_{\mathcal{D}}\big[\pi, \mathcal{D}]\big). \tag{2}$$

Now, consider the execution where $\mathcal{Z}_{\mathcal{D}}$ interacts with the simulator for the dummy adversary $\mathcal{S}_{\mathcal{D}}$ and protocol ϕ. Using Equation (1) and the fact that $\mathcal{Z}_{\mathcal{D}}$ is legal, we know that $\mathsf{Exec}\big(\pi, \mathcal{D}, \mathcal{Z}_{\mathcal{D}}[\pi, \mathcal{D}]\big)$ and $\mathsf{Exec}\big(\phi, \mathcal{S}_{\mathcal{D}}, \mathcal{Z}_{\mathcal{D}}[\pi, \mathcal{D}]\big)$ are (computationally) indistinguishable.

We now construct the simulator $\mathcal{S}$ for $\mathcal{A}$ and π from $\mathcal{S}_{\mathcal{D}}$: $\mathcal{S}$ internally runs instances of $\mathcal{A}$ and $\mathcal{S}_{\mathcal{D}}$ as follows:

- Messages from ϕ to the adversary are forwarded to $\mathcal{S}_{\mathcal{D}}$.
- Messages from $\mathcal{S}_{\mathcal{D}}$ to ϕ are forwarded to ϕ.
- Messages from $\mathcal{S}_{\mathcal{D}}$ to the environment are forwarded to $\mathcal{A}$.
- Messages from $\mathcal{A}$ to π are forwarded to $\mathcal{S}_{\mathcal{D}}$.
- Messages from $\mathcal{A}$ to the environment are forwarded to the environment.
- Messages from the environment to the adversary are forwarded to $\mathcal{A}$.

By applying Proposition 7 again, it follows that

$$\mathsf{Exec}\big(\phi, \mathcal{S}_{\mathcal{D}}, \mathcal{Z}_{\mathcal{D}}\big[\pi, \mathcal{D}]\big) \equiv \mathsf{Exec}\big(\phi, \mathcal{S}, \mathcal{Z}[\pi, \mathcal{A}]\big) \tag{3}$$

Combining Equations (1) to (3), the claim follows. □

TLUC security is also compatible with UC security, meaning that UC-secure protocols are also TLUC-secure.

Proposition 10 (Compatibility with UC Security). *Let π, ϕ be protocols such that $\pi \geq_{\mathrm{UC}} \phi$. Then, $\pi \geq_{\mathrm{TLUC}} \phi$.*

Proof. As TLUC emulation is a special case of UC emulation and the class of simulators is not restricted, Proposition 10 trivially follows. □

Transitivity. In contrast to UC security, TLUC security is not transitive. This means that there exist protocols π_1, π_2, π_3 such that $\pi_1 \geq_{\mathrm{TLUC}} \pi_2$ and $\pi_2 \geq_{\mathrm{TLUC}} \pi_3$, but $\pi_1 \not\geq_{\mathrm{TLUC}} \pi_3$. For an example, see Appendix C.5. However, we can state the following weaker and useful properties.

Corollary 6 (Transitivity for UC Protocols). *Let π_1, π_2, π_3 be protocols such that $\pi_1 \geq_{\mathrm{UC}} \pi_2$ and $\pi_2 \geq_{\mathrm{UC}} \pi_3$. Then, $\pi_1 \geq_{\mathrm{TLUC}} \pi_3$.*

Also, TLUC emulation is transitive in conjunction with UC emulation.

Proposition 11 (TLUC-UC Transitivity). *Let π_1, π_2, π_3 be protocols. If $\pi_1 \geq_{\mathrm{TLUC}} \pi_2$ and $\pi_2 \geq_{\mathrm{UC}} \pi_3$, then it holds that $\pi_1 \geq_{\mathrm{TLUC}} \pi_3$.*

Proof sketch. Let $\mathcal{A}$ be a legal adversary. Since $\pi_1 \geq_{\mathrm{TLUC}} \pi_2$, there exists a (possibly not legal) TLUC simulator $\mathcal{S}_1$ for every legal adversary $\mathcal{A}$ such that for every legal environment $\mathcal{Z}[\pi_1, \mathcal{A}]$, it holds that

$$\mathsf{Exec}\big(\pi_1, \mathcal{A}, \mathcal{Z}[\pi_1, \mathcal{A}]\big) \approx \mathsf{Exec}\big(\pi_2, \mathcal{S}_1, \mathcal{Z}[\pi_1, \mathcal{A}]\big) \tag{4}$$

and since $\pi_2 \geq_{\mathrm{UC}} \pi_3$, there exists a simulator $\mathcal{S}_2$ for all (not necessarily legal) adversaries $\mathcal{B}$ such that

$$\mathsf{Exec}\big(\pi_2, \mathcal{B}, \mathcal{Z}\big) \approx \mathsf{Exec}\big(\pi_3, \mathcal{S}_2, \mathcal{Z}\big) \tag{5}$$

As Equation (5) quantifies over all adversaries, it implies the existence of a simulator for the adversary / simulator $\mathcal{S}_1$.

We construct the simulator $\mathcal{S}_3$ for π_1 and $\mathcal{A}$ as follows: Internally, run $\mathcal{S}_2, \mathcal{S}_1$ and $\mathcal{A}$ as follows:

- Messages from π_3 to the adversary are forwarded to $\mathcal{S}_2$.
- Messages from $\mathcal{S}_2$ to the environment are forwarded to $\mathcal{S}_1$.
- Messages from $\mathcal{S}_2$ to π_3 are forwarded to π_3.
- Messages from $\mathcal{S}_1$ to the environment are forwarded to the environment.
- Messages from $\mathcal{S}_1$ to π_2 are forwarded to $\mathcal{S}_2$.
- Messages from the environment to the adversary are forwarded to $\mathcal{S}_1$.

In Equation (5), we have quantified over the environment $\mathcal{Z}[\pi_1, \mathcal{A}]$ as well as the simulator $\mathcal{S}_1$. Thus, it follows from Equations (4) and (5) and the definition of $\mathcal{S}_3$ that

$$\mathsf{Exec}\big(\pi_1, \mathcal{A}, \mathcal{Z}[\pi_1, \mathcal{A}]\big) \approx \mathsf{Exec}\big(\pi_3, \mathcal{S}_3, \mathcal{Z}[\pi_1, \mathcal{A}]\big)$$

□

Composition. In the following, we consider the case of a protocol ρ that makes one subroutine call to a protocol ϕ.

Theorem 10 (Single Instance Composition Theorem). *Let π, ϕ be subroutine-respecting protocols such that $\pi \geq_{\text{TLUC}} \phi$. Let ρ be a protocol that makes one subroutine call to ϕ. Then, $\rho^\pi \geq_{\text{TLUC}} \rho^\phi$.*

Let ρ be a protocol that UC-realizes the ideal protocol IDEAL($\mathcal{G}$) of some ideal functionality $\mathcal{G}$ and makes one subroutine call to the ideal protocol IDEAL($\mathcal{F}$) of some ideal functionality $\mathcal{F}$. Using Propositions 10 and 11 and Theorem 10, we can import ρ into TLUC, replace IDEAL($\mathcal{F}$) with an appropriate TLUC protocol while preserving security and conclude that the resulting composite protocol TLUC-realizes IDEAL($\mathcal{G}$).

Proof. In the following, we show that if π TLUC-emulates ϕ for the dummy adversary, then ρ^π emulates ρ^ϕ for the dummy adversary, i.e. for all legal environments, there exists a simulator $\mathcal{S}_\rho$ such that

$$\mathsf{Exec}\big(\rho^\pi, \mathcal{D}, \mathcal{Z}[\rho^\pi, \mathcal{D}]\big) \approx \mathsf{Exec}\big(\rho^\phi, \mathcal{S}_\rho, \mathcal{Z}[\rho^\pi, \mathcal{D}]\big). \tag{6}$$

By Proposition 9, this is sufficient and implies the general case. First, we construct a simulator $\mathcal{S}_\rho$ for ρ^π. Internally, $\mathcal{S}_\rho$ runs the simulator $\mathcal{S}_\mathcal{D}$ for π and handles messages as follows:

- Messages from protocol parties of ρ are sent to the environment.
- Messages from the environment to the protocol parties of ρ are sent to these parties.
- Messages from protocol parties of ϕ are sent to $\mathcal{S}_\mathcal{D}$.
- Messages from $\mathcal{S}_\mathcal{D}$ to ϕ are sent to ϕ.
- Messages from the environment to the protocol parties of π are sent to $\mathcal{S}_\mathcal{D}$.
- Messages from $\mathcal{S}_\mathcal{D}$ to the environment are sent to the environment.

We show that $\mathcal{S}_\rho$ is a valid simulator for ρ^π and $\mathcal{D}$ by contradiction: Suppose there exists an environment $\mathcal{Z}$ that can distinguish between the execution with ρ^π and $\mathcal{D}$ and the execution with ρ^ϕ and $\mathcal{S}_\rho$, i.e.

$$\mathsf{Exec}\big(\rho^\pi, \mathcal{D}, \mathcal{Z}[\rho^\pi, \mathcal{D}]\big) \not\approx \mathsf{Exec}\big(\rho^\phi, \mathcal{S}_\rho, \mathcal{Z}[\rho^\pi, \mathcal{D}]\big). \tag{7}$$

Let $\mathcal{Z}'[\pi, \mathcal{D}]$ be the type-2 routing environment (cf. Definition 12) that internally runs $\mathcal{Z}[\rho^\pi, \mathcal{D}]$ and ρ.

Using Proposition 7 and the definition of $\mathcal{S}_\rho$, it is easy to see that $\mathcal{Z}$'s view when emulated by $\mathcal{Z}'$ is identically distributed as in the execution of ρ^π and $\mathcal{A}$ resp. the execution of ρ^ϕ and $\mathcal{S}_\mathcal{D}$, depending on the challenge protocol of $\mathcal{Z}'$. By Proposition 7, $\mathcal{Z}'$ is legal for π and $\mathcal{D}$ if $\mathcal{Z}$ is legal for ρ^π and $\mathcal{D}$.

Thus, the distinguishing advantage of $\mathcal{Z}'$ for π and ϕ is identical to that of $\mathcal{Z}$ for ρ^π and ρ^ϕ and Equation (7) implies that

$$\mathsf{Exec}\big(\pi, \mathcal{D}, \mathcal{Z}'[\pi, \mathcal{D}]\big) \not\approx \mathsf{Exec}\big(\phi, \mathcal{S}_\mathcal{D}, \mathcal{Z}'[\pi, \mathcal{D}]\big), \tag{8}$$

contradicting the assumption that $\pi \geq_{\text{TLUC}} \phi$. Overall, it follows that if $\pi \geq_{\text{TLUC}} \phi$, then $\rho^\pi \geq_{\text{TLUC}} \rho^\phi$. □

Corollary 7 (UC Reusability). *Let ρ be a protocol that makes one subroutine call to a protocol ϕ such that $\rho^{\phi} \geq_{\mathrm{UC}} \sigma$ for some protocol σ. Let π be a protocol such that $\pi \geq_{\mathrm{TLUC}} \phi$. Then, $\rho^{\pi} \geq_{\mathrm{TLUC}} \sigma$.*

Unfortunately, TLUC security is not closed under general composition. More concretely, this means that there exist protocols π and ϕ such that $\pi \geq_{\mathrm{TLUC}} \phi$ holds, but $\rho^{\pi} \not\geq_{\mathrm{TLUC}} \rho^{\phi}$, where ρ makes *multiple* subroutine calls to ϕ. For an example, see Appendix C.6.

Environmental Friendliness. UC security has the desirable property of *environmental friendliness* [CLP13], which informally ensures that game-based security properties of protocols running along UC protocols ("in the environment") are not impacted by the UC execution. Unfortunately, this property does not hold for *all* game-based security properties for many notions that allow generic MPC in the plain model due to the use of super-polynomial simulation. What is more, determining whether the game-based property holds may be non-trivial, requiring e.g. to consider the security proof of the protocol in question. However, as TLUC security is a special case of UC security and considers polynomial-time simulation only, it inherits the environmental friendliness of UC security.

Environmental friendliness considers the validity of a game-based security property P when the presumptively P-secure cryptographic scheme Π, where P is defined via some security game, is executed alongside another protocol Π' with game-based security property Q. In particular, a UC-like execution running concurrently is considered.

First, we restate the notion of a *security game* as defined in [CLP13].

Definition 16 (Security Game, [CLP13]). *A* security game *(or game) consists of an ITM* Chal*, called the challenger, that is polynomial-time in the length of the messages it receives, and a constant $\tau_{\mathcal{C}}$, called the threshold, in the interval $[0, 1)$. In an execution of a security game, the challenger* Chal *interacts with an adversary A on common input 1^n and outputs* `accept` *or* `reject` *at the end of the interaction.*

We say that A_n breaks Chal_n with advantage ε, if A_n makes Chal_n accept with probability $\tau_{\mathcal{C}} + \varepsilon$. We say that A breaks Chal*, or the game-based assumption $\mathcal{C}$, if A_n breaks Chal_n with advantage $\varepsilon(n)$ for infinitely many $n \in \mathbb{N}$ for a non-negligible function ε. ε is the advantage of the adversary.*

An example for such a security game could be the IND-CPA game for encryption schemes with $\tau = 1/2$, i.e. the trivial winning probability of an adversary. However, Definition 16 is also valid for IND-CPA security with $\tau = 1$.

Based on *games*, one can define *assumptions*, which restrict the parameter τ such that there exists a trivial strategy for adversaries to win the game with probability τ.

Definition 17 (Game-Based Assumptions, [CLP13]). *A* game-based assumption *is simply a security game $\mathcal{C} = (\mathsf{Chal}, \tau)$, such that, there is a non-uniform PPT adversary A, called the* trivial strategy*, satisfying that A_n breaks*

Chal_n *with probability at least* τ *(possibly without any advantage) for all* $n \in \mathbb{N}$. *We say that assumption* $\mathcal{C}$ *holds if no non-uniform PPT adversary can break the game* (Chal, τ).

Definition 17 would rule out $\tau = 1$ for IND-CPA security, as there is no trivial winning strategy for the IND-CPA game with winning probability 1. However, the definition of game-based assumptions does not rule out the existence of insecure schemes Π for which the adversary has a non-negligible advantage over τ. This is covered in the following definition of *game-based security properties*.

Definition 18 (Game-Based Security Property, [CLP13]). *A* game-based security property *of a cryptographic scheme* Π *is simply a security game* $P_\Pi = (\mathsf{Chal}, \tau)$. *We say that the property* P_Π *holds if no non-uniform PPT adversary can break the game* (Chal, τ).

However, the game for IND-CPA security does not capture a setting where other protocols are executed concurrently. In order to argue that the security of Π is not impacted by a protocol ρ that runs concurrently and implements a functionality $\mathcal{G}$, the game P_Π has to be modified accordingly. The proceedings version [CLP13] gives an informal description of the associated game (see the full version[14] for a complete description):

Similar to UC security, an environment Z that gives input to the challenger Chal as well as ρ and may freely interact with the adversary A, is introduced. The adversary A not only interacts with Chal, but also with ρ (which may be a "real" protocol or the ideal protocol of some functionality $\mathcal{G}$). Z may, in particular, correlate the inputs of ρ and Chal. However, Π and ρ are never sub-routines of one another. As a consequence, environmental friendliness does not generally imply composability in the sense of subroutine replacement. Also, the adversary A does not "attack" the execution of ρ as in the UC execution experiment. Furthermore, there is no simulator. Like in the security game (Chal, τ), the adversary's success is determined by the output of Chal and not e.g. by the output of Z.

The game $\mathsf{Chal}^{\mathcal{G}/\rho}$ is defined similar to Chal, with the execption that (the ideal protocol of) $\mathcal{G}$ is replaced with ρ. In contrast to UC security, the environment Z and the adversary know whether $\mathcal{G}$ or ρ are executed.

The outlined game is (implicitly) considered in the following definition.

Definition 19 (Environmental Friendliness, [CLP13]). *Let* $P = (\mathsf{Chal}, \tau)$ *be a game-based security property of a cryptographic scheme* Π, *and* ρ *a protocol implementing a functionality* $\mathcal{G}$. *Then we say that* ρ *is* environmental friendly *to* Π *with property* P, *if the security property* $P^{\mathcal{G}/\rho} = (\mathsf{Chal}^{\mathcal{G}/\rho}, \tau)$ *holds.*

In order to prove the environmental friendliness of some protocol ρ that emulates an ideal functionality $\mathcal{G}$ according to some security notion (e.g. Angel-based security), the following general proof strategy is often applied (e.g. in [CLP10; CLP13]):

[14] `https://www.cs.cornell.edu/~rafael/papers/EnvFriendly-proc.pdf`

1. Define the game-based properties of Π that hold in an execution with $\mathcal{G}$.
2. Consider a property P that does not longer hold when $\mathcal{G}$ is replaced with ρ.
3. Use a presumptive simulator for ρ in the interaction with $\mathcal{G}$. Due to the simulatability, the adversary's success probability from Item 2 should be the same as in an interaction with ρ, except for a negligible difference. Depending on the security notion, the simulation may be inefficient (e.g. because the simulator has access to a super-polynomial helper).
4. Replace the inefficient simulation strategy with an efficient strategy (which is possible by assumption), e.g. by implementing the super-polynomial helper using rewinding that is possible when interacting with Chal.
5. Obtain a contradiction to Item 1.

Proposition 12 (Environmental Friendliness of TLUC Security). *Let π be a protocol that TLUC-emulates the ideal protocol of some functionality $\mathcal{G}$. Then π is friendly to every (non-timed) game-based property P of a protocol Π with property P.*

Proof. Proposition 12 state that UC security is environmentally friendly [CLP13] to all game-based properties. As TLUC security is a special case of UC security and all entities run in polynomial-time, the claim follows. □

Protocols running alongside composable MPC protocols may not only be affected by super-polynomial simulation, but also by non-uniform simulation. For example, Lin, Pass, and Venkitasubramaniam [LPV09] propose a variant of UC security where the environment runs in uniform polynomial-time, while the simulator runs in non-uniform polynomial-time. The non-uniform input of the simulator may impact the security of protocols that have started before the input is given to the simulator—even if these protocols are secure against non-uniform adversaries. As the definition of environmental friendliness is non-uniform, it does not capture this property.

Both the simulation and the reductions for our composable commitment scheme (Section 6) are uniform. Our constructions thus do not adversely affect security properties of previously started protocols that hold against polynomial-time adversaries.

Remark 5. Environmental friendliness as defined by [CLP13] is not meaningful for *timed* game-based properties such as the timed hiding property of a timed commitment scheme.

When considering an ideal functionality $\mathcal{F}$ and a concurrently executed protocol π using timed assumptions, the functionality $\mathcal{F}$ may already be unfriendly to timed properties of π. For example, $\mathcal{F}$ may perform computations that break time-lock puzzles used in π.

In the experiment of environmental friendliness, no simulator is not used. The (presumptive) simulator is only used to show that a protocol π is as friendly as a functionality $\mathcal{F}$ (which may already be unfriendly in our setting). Thus, the problems of environmental friendliness to protocols using timed assumptions start

well before considering the effects of the simulation, which additionally affect the environmental friendliness.

To the best of our knowledge, this novel environmental friendliness for timed game-based properties is not fulfilled by *any* security notion for composable MPC—not even for UC security.

Non-Triviality. While there exists no general and formal definition of non-triviality in the UC framework, Canetti et al. [Can+02] consider a protocol π to be a non-trivial realization of $\mathcal{F}$ if $\pi \geq_{\text{UC}} \text{IDEAL}(\mathcal{F})$ and for all adversaries $\mathcal{A}$ that deliver all messages and do not corrupt any party, the simulator $\mathcal{S}$ allows all outputs generated by $\mathcal{F}$.

With TLUC security, this notion is not sufficient as it does not consider the possibility that a protocol aborts due to timeouts, which may, depending e.g. on the environment, occur even if the adversary delivers all messages.

As an example, let π be a protocol that non-trivially UC-emulates $\mathcal{F}_{\text{COM}}$ and takes $t(\kappa)$ steps to execute successfully if all parties are honest. Now, let π' be the protocol that is identical to π, with the following exception. When receiving its input, the honest committer sets up a timer with $10t(\kappa)$ steps. At the onset of the unveil phase, it checks if the timer has expired and halts upon expiration. Clearly, π' should be considered non-trivial.

However, there exists a legal environment such that π' never generates output even if the legal adversary delivers all messages. As we do not want π' to be considered trivial if there also exists a legal environment $\mathcal{Z}$ for which π' always generates an output under the conditions outlined in [Can+02], we thus consider an appropriate notion that accounts for this[15].

Note that non-triviality may be lost under composition. To this end, take a protocol ρ^{ϕ} that makes one subroutine call to some protocol ϕ and is non-trivial. Replacing ϕ with its realization π that takes more steps than ϕ may make the composed protocol ρ^{π} trivial as timers in ρ may *always* be triggered due to the additional steps performed by the protocol π. However, that this does *not* render ρ^{π} insecure.

Impossibility Results. The well-known impossibility results due to Canetti and Fischlin [CF01] state that there is no bilateral (i.e. involving two communicating parties) and terminating (in the sense of correctness for honest parties) protocol π that UC-realizes $\mathcal{F}_{\text{COM}}$ in the plain model. This is due to the fact that if a protocol π is in the plain model, an environment is able to internally emulate every (presumptive) UC simulator for π.

We state the following variant of the impossibility result of [CF01] for TLUC-realizing $\mathcal{F}_{\text{COM}}$ in the plain model:

Theorem 11. *There exists no bilateral, non-trivial protocol π in the plain model where only one party sets up timers such that $\pi \geq_{\text{TLUC}} \mathcal{F}_{\text{COM}}$.*

[15] As legal environments count the number of computation steps independent of the input, the existence of *one* environment with fixed input implies the existence of *other* environments with all possible inputs such that π always generates an output.

Proof. The following proof follows the same outline as the proof of the impossibility result in [CF01]. Suppose that $\pi \geq_{\text{TLUC}} \text{IDEAL}(\mathcal{F}_{\text{COM}})$, i.e. for all legal adversaries, there exists a simulator $\mathcal{S}$ for all legal environments. We first consider the case where the committer is the party that sets up the timer(s). We use the presumptive simulator $\mathcal{S}$ to construct a legal environment $\mathcal{Z}[\pi, \mathcal{D}]$ expecting to interact with the dummy adversary for which there is no simulator, contradicting the assumption.

The environment $\mathcal{Z}[\pi, \mathcal{D}]$ works as follows: Initially, it orders the adversary to corrupt the committer and internally executes the simulator $\mathcal{S}$ for the corrupted *receiver* to generate an equivocal commitment. Prior to the beginning of the unveil phase, it samples a bit $b \xleftarrow{\$} \{0,1\}$ and instructs the simulator to decommit to b. `timer` and `notify` messages for / from the internal instance of $\mathcal{S}$ are handled by $\mathcal{Z}$ relative to a real execution where the committer is honest. This is possible without b as legal environments handle these message independent of parties' inputs and outputs.

As there are no "real" timers if the committer is corrupted and π is non-trivial, π is also non-trivial in this execution. Furthermore, $\mathcal{Z}[\pi, \mathcal{D}]$ is a legal environment.

If the environment is in the real execution of the protocol π, then the internally emulated simulator can, due to the correctness of the simulation, unveil the commitment to the bit b, except with negligible probability. In particular, the `timer` and `timeout` messages are handled as in an interaction with a legal adversary relative to the internally emulated instance of $\mathcal{S}$.

If the environment is in the ideal execution of $\mathcal{F}_{\text{COM}}$, the whole execution is independent of b until the onset of the unveil phase. Thus, *any* simulator will extract the wrong bit b' with probability $1/2$, where the probability is over the coins of the simulator and the environment. Even if the internally emulated simulator failed during the unveil phase with non-negligible probability, this would allow the environment to distinguish as such a failure does not occur in the real execution, except with negligible probability.

We now consider the case that the receiver is the only party setting up timer(s). At the onset of the execution, the environment $\mathcal{Z}[\pi, \mathcal{D}]$ samples $b \xleftarrow{\$} \{0,1\}$ and sends b as input to the honest committer. It also orders the adversary to corrupt the receiver and internally executes the simulator $\mathcal{S}$ for the corrupted *sender* to extract the commitment of the honest committer. Again, timers for the internally emulated simulator are handled relative to a presumptive real execution (independent of the honest committer's input b). In the real execution, the internally emulated simulator will succeed in extracting the correct bit with overwhelming probability. In the ideal execution, the extracted bit b' will either be independent of b or the extraction may fail altogether, allowing $\mathcal{Z}$ to distinguish. □

By introducing a temporary asymmetry between simulator and environment, e.g. when the environment counts the steps relative to the real-world adversary, non-trivial and environmentally friendly realizations of UC-complete functionalities in the plain model using timed assumptions become possible.

C.4 Example for Side-Channels

With TLUC security, we require legal environments to count steps performed by protocol parties independent of the parties' actual inputs, which may influence the number of steps taken. In the following, we show that this is necessary in order to achieve a meaningful notion of security. If environments were to count the actual number of steps depending on the input, this would introduce a side-channel that can leak the secret of honest parties even in the ideal execution. As a consequence, obviously insecure protocols could be proven secure.

As an example, we consider a commitment scheme in the plain model that is clearly insecure but can be shown to emulate $\text{IDEAL}(\mathcal{F}_{\text{COM}})$ if legal environments count the actually performed steps instead of an upper bound independent of a party's input.

The protocol π between a committer and a receiver is defined as follows:

- Upon receiving $(\texttt{commit}, sid, b)$, the committer sends a message $(\texttt{init}, sid)$ to the receiver. The receiver then samples $r_0, r_1 \xleftarrow{\$} \{0,1\}^\kappa$ and sends $(\texttt{timer}, r_0, 10000\kappa)$ to the adversary. Upon its next activation, the receiver sends $(\texttt{timer}, r_1, 1000\kappa)$ to the adversary. Upon its next activation, it sends $(\texttt{init}, sid)$ to the committer. If $b = 0$, the committer performs $1000\kappa + 1$ steps. If $b = 1$, the committer performs $10000\kappa + 1$ steps. Afterwards, it sends $(\texttt{done}, sid)$ to the receiver. The receiver checks the status of both timers, remembers the result and outputs $(\texttt{committed}, sid)$.
- Upon receiving $(\texttt{unveil}, sid)$ as input, the committer sends (sid, b) to the receiver. If $b = 0$ and the first timer has timed out but the second has not, it outputs $(\texttt{unveil}, sid, 0)$. If $b = 1$ and both timers have timed out, it outputs $(\texttt{unveil}, sid, 1)$. Otherwise, it halts.

Let $\mathcal{S}$ be the following simulator for π and the dummy adversary.

- If the committer is corrupted, internally run the protocol of the honest receiver. After having received $(\texttt{done}, sid)$, interact with $\mathcal{F}_{\text{COM}}$ as follows: If only first timer has timed out, send $(\texttt{commit}, sid, 0)$ to $\mathcal{F}_{\text{COM}}$ on behalf of the corrupted sender. Otherwise, send $(\texttt{commit}, sid, 1)$. Upon receiving the bit b' during the unveil phase, do the following checks: If $b' = 0$ and only the first timer has timed out, send $(\texttt{unveil}, sid)$ to $\mathcal{F}_{\text{COM}}$ on behalf of the corrupted committer and allow the receiver's output. If $b' = 1$ and both timers have timed out, send $(\texttt{unveil}, sid)$ to $\mathcal{F}_{\text{COM}}$ on behalf of the corrupted committer and allow the receiver's output. Otherwise, halt.
- If the receiver is corrupted, internally run the protocol of the honest committer on input $b = 0$ after getting the request for the receiver's output from $\mathcal{F}_{\text{COM}}$. When receiving the output request $(\texttt{unveil}, sid, b)$ from $\mathcal{F}_{\text{COM}}$, send (sid, b) to the corrupted receiver.
- If both parties are honest, follow the strategy for the corrupted receiver and additionally simulate the messages of the honest receiver. Eventually, allow the output of $\mathcal{F}_{\text{COM}}$ if the honest receiver would accept.

Now, consider that we require environments to always count the steps relative to the actual steps performed by the protocol parties. Clearly, the protocol is non-trivial (cf. Appendix C). If the sender is corrupted, any legal environment $\mathcal{Z}$ will handle the timers as in the real execution. Thus, the receiver's output is identically distributed as the simulator $\mathcal{S}$ can extract the bit that will be accepted by the receiver (if it exists). Conversely, if the receiver is corrupted, the environment $\mathcal{Z}$ does not know the number of steps performed by the simulator. Thus, the commitment is not binding and the simulator can always unveil the bit that the honest committer has sent to $\mathcal{F}_{\text{COM}}$.

It follows that any legal environment's view in an execution with π and $\mathcal{D}$ is identically distributed to an execution with IDEAL($\mathcal{F}_{\text{COM}}$) and $\mathcal{S}$ if the steps counted by $\mathcal{Z}$ depend on secret inputs.

C.5 Counterexample for Transitivity

In contrast to UC security, TLUC security is not transitive.

Proposition 13. *There exist protocols π_1, π_2, π_3 such that $\pi_1 \geq_{\text{TLUC}} \pi_2$ and $\pi_2 \geq_{\text{TLUC}} \pi_3$ but $\pi_1 \not\geq_{\text{TLUC}} \pi_3$.*

Proof. Let π_3 the ideal protocol of the ideal commitment functionality $\mathcal{F}_{\text{COM}}$. Let π_2 be the protocol π_{MCOM} (with slight syntactical modifications towards realizing $\mathcal{F}_{\text{COM}}$ instead of $\mathcal{F}_{\text{MCOM}}$) from Section 6. Let π_1 be the following protocol:

- Upon receiving $(\texttt{commit}, sid, b)$ as input, the committer samples $b' \xleftarrow{\$} \{0,1\}$ and sends $c = b \oplus b'$ to the receiver, which stores c and outputs $(\texttt{committed}, sid)$.
- Upon receiving $(\texttt{unveil}, sid)$ as input, the committer sends (b, b') to the receiver. If $c = b \oplus b'$, the receiver outputs $(\texttt{unveil}, sid, b)$.

Clearly π_1 does not realize $\mathcal{F}_{\text{COM}}$, as the commitment is not extractable (and not even binding for a malicious sender). However, one can show that $\pi_1 \geq_{\text{TLUC}} \pi_2$: Roughly, the simulator $\mathcal{S}$ for π_1 acts as follows: If the sender is corrupted, the simulator must be able to equivocate the commitment in π_2. This is possible as the simulator is able to determine the output of the coin-toss for the equivocation CRS. (This possible is as the corrupted committer is played by the simulator and not the environment in this situation.) Conversely, if the receiver is corrupted, the simulator sends a random value $c \xleftarrow{\$} \{0,1\}$ in the simulation of π_1. Later on, when it learns the committed bit b in π_2, the simulator sends $(b, c \oplus b)$ as unveil message. □

Note that the above simulator is not a legal adversary. While this is not necessary to argue that $\pi_1 \geq_{\text{TLUC}} \pi_2$, the prerequisite $\pi_2 \geq_{\text{TLUC}} \pi_3$ only requires the existence of a simulator for a *legal* adversary for π_1, which $\mathcal{S}$ is not.

C.6 Counterexample for Composition

TLUC security is not closed under general composition.

Proposition 14. *Let π, ϕ be subroutine-respecting protocols such that $\pi \geq_{\text{TLUC}} \phi$. There exists a protocol ρ that makes multiple subroutine calls to ϕ such that $\rho^{\pi} \not\geq_{\text{TLUC}} \rho^{\phi}$.*

As an example, take the commitment protocol π_{MCOM} (cf. Section 6) and replace the pCCA-secure commitment scheme $\mathsf{COM}_{\mathsf{pCCA}}$ used in SSCOM with a malleable extractable commitment scheme. One can easily prove that this protocol still TLUC-realizes a single instance of $\mathcal{F}_{\text{COM}}$, but is not even concurrently self-composable.