On (expected polynomial) runtime in cryptography

Michael Klooß*

A common definition of black-box zero-knowledge considers *strict polynomial time* (PPT) adversaries but *expected polynomial time* (EPT) simulation. This is necessary for constant round black-box zero-knowledge in the plain model, and the asymmetry between simulator and adversary an accepted consequence. Consideration of EPT adversaries naturally leads to *designated* adversaries, i.e. adversaries which are only required to be efficient in the protocol they are designed to attack. They were first examined in Feige's thesis [Fei90], where obstructions to proving security are shown. Prior work on (designated) EPT adversaries by Katz and Lindell (TCC'05) requires superpolynomial hardness assumptions, whereas the work of Goldreich (TCC'07) postulates "nice" behaviour under rewinding.

In this work, we start from scratch and revisit the definition of *efficient* algorithms. We argue that the standard runtime classes, PPT and EPT, behave "unnatural" from a cryptographic perspective. Namely, algorithms can have indistinguishable runtime distributions, yet one is considered efficient while the other is not. Hence, classical runtime classes are not "closed under indistinguishability", which causes problems. Relaxations of PPT which are "closed" are (well-)known and used.

We propose *computationally* expected polynomial time (CEPT), the class of runtimes which are (computationally) indistinguishable from EPT, which is "closed". We analyze CEPT in the setting of *uniform complexity* (following Goldreich (JC'93)) with *designated adversaries*, and provide easy-to-check criteria for zero-knowledge protocols with black-box simulation in the plain model which show that many (all known?) such protocols handle designated CEPT adversaries in CEPT.

1. Introduction

Interactive proof systems allow a prover $\mathcal{P}$ to convince a verifier $\mathcal{V}$ of the "truth" of a statement x, i.e. that $x \in \mathcal{L}$ for some language $\mathcal{L}$. Soundness of the protocol ensures that if the verifier accepts, then $x \in \mathcal{L}$ with high probability. *Zero-knowledge* proof systems allow $\mathcal{P}$ to convince $\mathcal{V}$ of $x \in \mathcal{L}$ *without revealing anything else.* The definition of zero-knowledge relies on the (more general) simulation paradigm: It stipulates that, for every (malicious) verifier $\mathcal{V}^*$, there is a simulator Sim which, given only the inputs x, *aux* of $\mathcal{V}^*$, can produce a *simulated* output (or *view*[1]) *out* $= \mathsf{Sim}(x, \mathit{aux})$, which is indistinguishable from the output $\mathsf{out}_{\mathcal{V}^*}\langle \mathcal{P}(x, w), \mathcal{V}^*(x, \mathit{aux})\rangle$ of a real interaction. Thus, anything $\mathcal{V}^*$ learns in the interaction, it could simulate itself — *if* Sim *and* $\mathcal{V}^*$ *lie in the same complexity class.*

Let us write X/Y (zero-knowledge) for adversary complexity X and simulator complexity Y. The two widespread notions of zero-knowledge are PPT/PPT and PPT/EPT. The former satisfies the "promise

* Karlsruhe Institute of Technology, michael.klooss@kit.edu

[1] We use view and output synonymously in the introduction.

of zero-knowledge", but comes at a price. Barak and Lindell [BL04] show that it is impossible to construct *constant* round proof systems with *black-box* simulation and negligible soundness error in the plain model. Since *constant round black-box* zero-knowledge is attractive for many reasons, the relaxation of PPT/EPT zero-knowledge is common. However, this asymmetry breaks the "promise of zero-knowledge". The adversary *cannot* execute Sim, hence it cannot simulate the interaction. More concretely, this setting does not compose well. If we incorporate an EPT simulator into a (previously PPT) adversary, the new adversary is EPT. This common approach — constructing simulators for more complex systems from simulators of building blocks — therefore fails due to the asymmetry.

To remedy the asymmetry, we need to handle EPT adversaries. There are several sensible definitions of EPT adversaries, but the arguably most natural choice are *designated* EPT adversaries. That is, adversaries which only need to be *EPT when interacting with the protocol they are designed to attack*. Feige [Fei90] first considered this setting, and demonstrates significant technical obstacles against achieving security in the presence of such attacks.

The problems of EPT (and designated adversaries) are not limited to zero-knowledge, and extend to the simulation paradigm, e.g. multi-party computation.

Preliminary conventions. Throughout, κ denotes the security parameter. We generally consider objects which are families (of objects) parameterized by κ, but often leave the dependency implicit. We abbreviate *systems of (interactive) machines (or algorithms)* by *system*. A system is *closed*, if it only expects κ as input, and produces some output. For example, a prover $\mathcal{P}$ does not constitute a closed system, nor does the interaction $\langle \mathcal{P}, \mathcal{V} \rangle$, since it still lacks the inputs to $\mathcal{P}$ and $\mathcal{V}$. Our primary setting is *uniform complexity* [Gol93], where inputs to an (otherwise closed) system are generated efficiently by so-called *input generators*. Interaction of algorithms A, B is denoted $\langle \mathsf{A}, \mathsf{B} \rangle$, the time spent in A is denoted $\mathsf{time}_{\mathsf{A}}(\langle \mathsf{A}, \mathsf{B} \rangle)$, and similarly for time spent in B or $\mathsf{A} + \mathsf{B}$. Oracle access to $\mathcal{O}$ is written $\mathsf{A}^{\mathcal{O}}$.

1.1. Obstacles

We first recall some obstacles regarding expected runtime and designated adversaries which we have to keep in mind. For more discussions and details, we refer to the excellent introductions of [KL08; Gol10] and to [Fei90, Section 3].

Runtime squaring. Consider (a family of) random variables T_κ over $\mathbb{N}$, where $\mathbb{P}(T_\kappa = 2^\kappa) = 2^{-\kappa}$ and T is 0 otherwise. Then T_κ has polynomially bounded expectation $\mathbb{E}(T_\kappa) = 1$, but $\mathbb{E}(T_\kappa^2) = 2^\kappa$. That is $S_\kappa = T_\kappa^2$ is *not* expected polynomial time anymore. This behaviour not only prevents machine model independence of EPT as an efficiency notion, but also the non-black-box simulation technique of Barak [Bar01] (which suffers from a quadratic growth in runtime).

Composition and rewinding. Consider an oracle algorithm $\mathsf{A}^{\mathcal{O}}$ with access to a PPT oracle $\mathcal{O}$. Then to check if the total time $\mathsf{time}_{\mathsf{A}+\mathcal{O}}(\mathsf{A}^{\mathcal{O}})$ is PPT, we can count an oracle call as a single step. Moreover, it makes no difference if A has "straightline" or "rewinding" access to $\mathcal{O}$. For EPT, even a standalone definition of "$\mathcal{O}$ is EPT" is non-trivial and possibly fragile. For example, there are oracles, where any PPT A with "straightline" access to $\mathcal{O}$ results in an EPT interaction, yet access "with rewinding" to $\mathcal{O}$ allows an explosion of expected runtime. See [KL08] for a concrete example.

Designated EPT adversaries. For a **designated adversary** $\mathcal{A}$ against zero-knowledge of a proof system $(\mathcal{P}, \mathcal{V})$, we require (only) that $\mathcal{A}$ is *efficient when interacting with that protocol*. Since a zero-knowledge simulator *deviates* from the real protocol, the runtime guarantees of $\mathcal{A}$ are void.

1.2. Motivation: Reproving zero-knowledge of graph 3-colouring

The constant-round black-box zero-knowledge proof of Goldreich and Kahan [GK96] is our running example for demonstrating problems and developing our approach.

Recall that (non-interactive) commitment schemes allow a committer to commit to a value in a way which is *hiding* and *binding*, i.e. the commitment does not reveal the value to the receiver, yet it can be unveiled to at most one value. A commitment scheme consists of algorithms $(\mathsf{Gen}, \mathsf{Com}, \mathsf{VfyOpen})$. The *commitment key* is generated via $\mathsf{ck} \leftarrow \mathsf{Gen}(\kappa)$. For details, see Appendix B.1.

1.2.1. The constant round protocol of Goldreich–Kahan

The protocol of [GK96] uses two different commitments, $\mathsf{Com}^{(\mathrm{H})}$ is perfectly hiding, $\mathsf{Com}^{(\mathrm{B})}$ is perfectly binding. The idea of protocol $\mathrm{G3C}_{\mathrm{GK}}$ is a parallel, N-fold, repetition of the standard zero-knowledge proof for G3C, with the twist that the verifier commits to all of its challenges beforehand. Let $G = (V, E)$ be the graph and let ψ be a 3-colouring of G. The prover is given (G, ψ) and the verifier G.

(P0) $\mathcal{P}$ sends $\mathsf{ck}_{\mathsf{hide}} \leftarrow \mathsf{Gen}^{(\mathrm{H})}(\kappa)$. ($\mathsf{ck}_{\mathsf{bind}} \leftarrow \mathsf{Gen}^{(\mathrm{B})}(\kappa)$ is deterministic.)
(V0) $\mathcal{V}$ picks $N = \kappa \cdot \mathsf{card}(E)$ challenge edges $e_i \leftarrow E$, and commits to them using $\mathsf{Com}^{(\mathrm{H})}$.
(P1) $\mathcal{P}$ picks randomized colourings for each of the N parallel repetitions of the standard graph 3-colouring proof system, and sends the $\mathsf{Com}^{(\mathrm{B})}$-committed randomized node colours to $\mathcal{V}$.
(V1) $\mathcal{V}$ opens all commitments (to e_i).
(P2) $\mathcal{P}$ aborts if any opening is invalid. Otherwise, $\mathcal{P}$ proceeds in the parallel repetition using these challenges, i.e. in the i-th repetition $\mathcal{P}$ opens the committed colours for e_i.
(V2) $\mathcal{V}$ aborts iff any opening is invalid, any edge not correctly coloured, or if $\mathsf{ck}_{\mathsf{hide}}$ is "bad".

The soundness of this protocol follows from $\mathsf{Com}^{(\mathrm{H})}$ being statistically hiding. Therefore, each of the N parallel repetitions is essentially an independent repetition of the usual graph 3-colouring proof. For $N = \kappa \cdot \mathsf{card}(E)$ parallel rounds, the probability to successfully cheat is negligible (in κ), see [GK96].

1.2.2. Proving zero-knowledge: A (failed?) attempt

Now, we prove black-box zero-knowledge for *designated adversaries*. That is, we describe a simulator which uses the adversary $\mathcal{V}^*$ only as a black-box, which can be queried and rewound to a (previous) state. We proceed in three game hops, gradually replacing the view of a real interaction with a simulated view. Successive games are constructed so that their change in output (which is a purported view) is indistinguishable.

- G_0 This is the real G3C protocol. The output is the real view.
- G_1 The prover rewinds a verifier which completes (V1) successfully (i.e. sends *valid* openings on the first try) to (V0) and repeats (P1) until a second run where $\mathcal{V}$ validly opens all commitments. The output is the view of this second succesful run. The prover uses fresh randomness in each reiteration of (P1) (whereas the black-box has fixed randomness).
- G_2 If the two openings in (V1) differ, return `ambig`, indicating ambiguity of the commitment. Otherwise, proceed unchanged.
- G_3 The initial commitments (in (P1)) to a 3-colouring are replaced with commitments to 0. These commitments are never opened. In successive iterations, the commitments to a 3-colouring are replaced by commitments to a pseudo-colouring. These commitments, when opened, simulate a valid 3-colouring at the challenge edges e_i.

Evidently, Game G_3 outputs a purported view independent of the witness. Thus, the simulator is defined as in G_3: In a first try, it commits to garbage instead of a 3-colouring in (P1), in order to obtain the verifier's challenge (in (V1)). If the verifier does not successfully open the commitments (in (V1)), Sim aborts (as an honest prover would) and outputs the respective view. Otherwise, Sim rewinds the

verifier to Step 2 and sends a pseudo-colouring (w.r.t. the previously revealed challenge) instead. Sim retries until the verifier succesfully unveils (in (V1)) again. (If the verifier opens to a different challenge, return *view* = `ambig`.)

Now, we sketch a security proof for Sim. We argue by game hopping.

G_0 **to** G_1. The expected number of rewinds is at most 1. Namely, if $\mathcal{V}^*$ opens in (V1) with probability ε, then an expected number of $\frac{1}{\varepsilon}$ rewinds are required. Consequently, the expected runtime is polynomial (and G_1 is EPT). The output distribution of the games is identical.

G_1 **to** G_2. It is easy to obtain an adversary against the binding property of $\mathsf{Com}^{(\mathrm{H})}$ which succeeds with the same probability that G_2 outputs `ambig`. Thus, this probability is negligible.

G_2 **to** G_3. Embedding a (multi-)hiding game for $\mathsf{Com}^{(\mathrm{B})}$ in this step is straightforward. Namely, using the left-or-right indistinguishability formulation, where the commitment oracle either commits the first or second challenge message. Thus, by security of the commitment scheme, G_2 and G_3 are indistinguishable.[2]

A closer look. The above proof is clear and simple. But the described simulator is not EPT! While G_2 and G_3 are (computationally) indistinguishable, the transition *does not necessarily preserve expected polynomial runtime* [Fei90; KL08]. Feige [Fei90] points out a simple attack, where $\mathcal{V}^*$ brute-forces the commitments with some tiny probability p, and runs for a very long time if the contents are not valid 3-colourings. This is EPT in the real protocol, but our simulator as well as the simulator in [GK96] do not handle $\mathcal{V}^*$ in EPT. The problem lies with *designated* adversaries as following example shows.

Example 1.1. Let $\mathcal{V}^*$ sample in step (V0) a garbage commitment c to zeroes, just like Sim. Now $\mathcal{V}^*$ unveils e in (V1) if and only if it receives c. (c is a "proof of simulation".) The honest prover always aborts in (P2) because $\mathcal{V}^*$ will never unveil. But if Sim queried c as its garbage commitment, the simulation runs forever, because $\mathcal{V}^*$ unveils only for this c, which is not a pseudo-colouring.

As described, $\mathcal{V}^*$ is a priori PPT, and indeed, the simulator in [GK96] uses a "normalization technique" which prevents this attack. However, exploiting *designated* PPT, $\mathcal{V}^*$ may instead run for a very long time, when it receives c.

Obstructions to simple fixes. Let us recall a few simple, but insufficient fixes. A first idea is to *truncate* the execution of $\mathcal{A}$ at some point. For PPT adversaries, this may seem viable.[3] However, there are EPT adversaries, or more concretely runtime distributions, where *any strict polynomial truncation* affects the output in the real protocol *noticeably*. So we cannot expect that such a truncation works well for Sim. See [Fei90, Section 3] for a more convincing argument against truncation.

Being unable to truncate, we could enforce better behaviour on the adversary. Intuitively, it seems enough to require that $\mathcal{V}^*$ runs in expected polynomial time *in any interaction* [KL08; Gol10]. However, even this is not enough. Katz and Lindell [KL08] exploit the soundness error of the proof system to construct an adversary which runs in expected polynomial time in any interaction, but still makes the expected runtime of the simulator superpolynomial. The problem is that these runtime guarantees are void in the presence of *rewinding*.

Modifications of these fixes work, but at a price: Katz and Lindell [KL08] use *superpolynomial* truncation and need to assume *superpolynomial* hardness. Goldreich [Gol10] *restricts* to algorithms (hence adversaries) which behave well under rewinding. We discuss these in Section 1.5. Our price are proof techniques, which become more technical and, perhaps, more limited.

Our fix: There is no problem. Our starting point is *the conviction* that the given "proof" should *evidently* establish the security of the scheme for any *cryptographically sensible* notion of runtime. If

[2] We rely on security of binding and hiding against *expected time* adversaries, which easily follows from PPT-security.

[3] Even there, the situation is far from easy. In a UC setting with an *a posteriori* efficiency notion (and designated adversaries), Hofheinz, Unruh, and Müller-Quade show in [HUM13, Section 9] that (pathological) functionalities can make simulation in PPT is impossible (if one wants security under composition for just a single instance).

one could *distinguish* the runtime of G_2 and G_3, then this would break the hiding property of the commitment scheme. Thus, the *runtimes are indistinguishable*. Following, in computational spirit, Leibniz' "identity of indiscernibles", we declare runtimes which are *indistinguishable from efficient* by efficient distinguishers as *efficient per definition*. With this, the proof works and the simulator, while not expected polynomial time, is *computationally expected polynomial time* (CEPT), which means its runtime distribution is indistinguishable from EPT.

We glossed over a crucial detail: We solved the problem with the very strategy we claim to fix — different runtime classes for Sim and $\mathcal{V}^*$! Fortunately, Sim also handles CEPT adversaries in CEPT.

1.3. Contribution

Our main contribution is the reexamination of the notion of runtime in cryptography. We offer a novel, and arguably natural, alternative solution for a problem that was never fully resolved. Our contribution is therefore primarily of explorational and definitional nature. More concretely:

- We define CEPT, a small relaxation of EPT with a convenient characterization.
- To the best of our knowledge, this is the first work which embraces *uniform*[4] complexity, *expected* time, and *designated* adversaries.
- We develop general tools for this setting, most importantly, a hybrid lemma.
- Easy-to-check criteria show that many (all known?) black-box zero-knowledge arguments from standard assumptions in the plain model[5] have CEPT simulators which handle designated CEPT adversaries. Consequently, security against designated adversaries is natural. For example, the proof systems [GMW86; GK96; Lin13; Ros04; KP01; PTV14] satisfy our criteria.
- We impose no (non-essential) restrictions on the adversary, nor do we need additional (hardness) assumptions.
- We sketch the application of our techniques to secure function evaluation (SFE), and demonstrate that auxiliary input security implies modular sequential composition.

All of this comes at a price. Our notions and proofs are not complicated, yet somewhat technical. This is, in part, because of a posteriori runtime and uniform complexity. Still, we argue that we have demonstrated the viability of our new notion of efficiency, at least for zero-knowledge.

A complexity theoretic perspective. This work is only concerned with the complexity class of feasible *attacks*, and does not assume or impose complexity requirements on protocols. Due to designated adversaries, the complexity class of adversaries is (implicitly) defined per protocol, similar to [KL08]. We bootstrap feasibility from complexity classes for (standalone) sampling algorithms, i.e. algorithms with no inputs except κ. Hence a (designated) adversary is feasible if the *completed system* of protocol and adversary (including input generation) is CEPT (or more generally, in some complexity class of feasible sampling algorithms).

The complexity class of simulators is relative to the adversary, and thus depends both on the protocol and the ideal functionality. Namely, feasibility of a simulator Sim means that if an adversary $\mathcal{A}$ is feasible (w.r.t. the protocol), then "$\mathsf{Sim}(\mathcal{A})$" is feasible (w.r.t. the ideal functionality).

Comments on our approach. The uniform complexity setting drives complexity, yet is necessary, since a notion of time that depends on non-uniformity is rather pathological. Losing the power of non-uniformity (and strictness of PPT) requires many small adjustments to definitions.[6] Moreover,

[4] Our results are applicable to a minor generalization of the non-uniform setting as well, namely non-uniformly generated input *distributions*, see Appendix E.1.2.

[5] Unfortunately, problems might arise with superpolynomial hardness assumptions, see Section 8.

[6] For example, we need a stateful distinguisher for modular sequential security, whereas non-uniformly, state and even randomness can be trivially removed by coin-fixing, demonstrating the equivalence of many variations, whose equivalence in the uniform setting not clear. Thus, our definition of auxiliary input zero-knowledge deviates slightly from [Gol93].

annoying technical problems with efficiency arise inadvertently, depending on formalizations of games and models. As in prior work, we mostly ignore them, but do point them out and propose solutions. They are easily fixed by adding "laziness", "indirection", or "caching".

An important point raised by a reviewer of TCC'20 is the "danger of zero-knowledge being trivialized" by "expanding the class of attacks", and a case for "moving towards knowledge tightness" (with which we fully agree). Many variations of zero-knowledge, from weak distributional [Dwo+03; CLP15] to precise [MP06; DG12], exist. We argue that our notion is very close to the "standard" notion with EPT simulation, but allows designated (C)EPT adversaries. Indeed, it seems to gravitate towards "knowledge tightness" [Gol10], as seen by runtime explosion examples due to expectation.

1.4. Technical overview and results

We give an overview of our techniques, definitions, and results. Recall that we only consider runtimes for closed systems (which receive only κ as input and produce some output). W.r.t. *uniform complexity* and *designated adversaries*, i.e. adversaries which only need to be efficient in the real protocol [Fei90], closed systems are the default situation anyway. A **runtime class** $\mathcal{T}$ is a set of runtime distributions. A **runtime (distribution)** is a family $(T_\kappa)_\kappa$ of distributions T_κ over $\mathbb{N}_0$. We use *runtime* and *runtime distribution* synonymously. Computational $\mathcal{T}$-time indistinguishability of oracles and distributions is defined in the obvious way (c.f. Section 2.6). For statistical $\mathcal{T}$-query indistinguishability, we count *only* queries as steps, and require $\mathcal{T}$-time w.r.t. this. (In our setting, unbounded queries often imply perfect indistinguishability, which is too strong.)

1.4.1. The basic tools

Statistical vs. computational indistinguishability. The (folklore) *equivalence* of statistical and computational indistinguishability for distributions with *"small" support* is a simple, but central, tool. For polynomial time, "small" support means polynomial support, say $\{0, \dots, \mathsf{poly}_1(\kappa)\}$ since we consider runtime distributions. Assuming non-uniform advice, the advice is large enough to encode the optimal decisions, achieving statistical distance as distinguishing advantage. This extends to "polynomially-tailed" runtime distributions T. There, by assumption, for any poly_0 there is a poly_1 such that $\mathbb{P}(T_\kappa > \mathsf{poly}_1(\kappa)) \leq \frac{1}{\mathsf{poly}_0(\kappa)}$, Hence, we can reduce to strict polynomial support by truncating at poly_1, sacrificing $1/\mathsf{poly}_0$ in statistical distance. The Markov bound shows that expected polynomial time is polynomially tailed. Removing non-uniformity is possible with repeated sampling, e.g. by approximating the distribution.

Standard reduction. Another simple, yet central, tool is the *standard cutoff argument* (Section 4.1). It is the core tool to obtain *efficiency from indistinguishability*.

Lemma 1.2 (Standard reduction to PPT). *Let $\mathcal{D}$ be a distinguisher for two oracles $\mathcal{O}_0$, $\mathcal{O}_1$ (which may sample distributions, or model an IND-CPA game, or ...). Suppose $\mathcal{D}$ has advantage at least $\varepsilon \geq \frac{1}{\mathsf{poly}_{\mathrm{adv}}}$ (infinitely often). Suppose furthermore that $\mathcal{D}^{\mathcal{O}_0}$ is CEPT with expected time poly_0. Then there is an a priori PPT distinguisher $\mathcal{A}$ with advantage at least $\frac{\varepsilon}{4}$ (infinitely often).*

We stress that we require *no runtime guarantees* for $\mathcal{D}^{\mathcal{O}_1}$ — it may never halt for all we know. For a proof sketch, define $N = 4\mathsf{poly}_0 \cdot \mathsf{poly}_{\mathrm{adv}}$ and let $\mathcal{A}$ be the runtime cutoff of $\mathcal{D}$ at N. The outputs of $\mathcal{A}^{\mathcal{O}_0}$ and $\mathcal{D}^{\mathcal{O}_0}$ are $\frac{\varepsilon}{4}$ close. For $\mathcal{A}^{\mathcal{O}_1}$ and $\mathcal{D}^{\mathcal{O}_1}$ this may be false. However, if $\mathcal{D}^{\mathcal{O}_1}$ exceeds N steps with probability higher than $\frac{2\varepsilon}{4}$, then the runtime is a distinguishing statistic with advantage $\frac{\varepsilon}{4}$. Thus, we can assume the outputs of $\mathcal{A}^{\mathcal{O}_1}$ and $\mathcal{D}^{\mathcal{O}_1}$ are $\frac{2\varepsilon}{4}$ close. Now, a short calculation shows that $\mathcal{A}$ has advantage at least $\frac{\varepsilon}{4}$. Namely, $\Delta(\mathcal{A}^{\mathcal{O}_1}, \mathcal{A}^{\mathcal{O}_0}) \geq \Delta(\mathcal{D}^{\mathcal{O}_1}, \mathcal{D}^{\mathcal{O}_0}) - \Delta(\mathcal{A}^{\mathcal{O}_1}, \mathcal{D}^{\mathcal{O}_1}) - \Delta(\mathcal{D}^{\mathcal{O}_0}, \mathcal{A}^{\mathcal{O}_0})$.

1.4.2. Computationally expected polynomial time

We define the runtime classes $\mathcal{PPT}$ (resp. $\mathcal{EPT}$), as usual, i.e. $(T_\kappa)_\kappa \in \mathcal{PPT} \iff \exists \mathsf{poly}\colon \mathbb{P}(T_\kappa \leq \mathsf{poly}(\kappa)) = 1$ (resp. $(T_\kappa)_\kappa \in \mathcal{EPT} \iff \exists \mathsf{poly}\colon \mathbb{E}(T_\kappa) \leq \mathsf{poly}(\kappa)$).

Definition 1.3 (Simplified[7] Definition 3.5). A runtime S, i.e. a family of random variables S_κ with values in $\mathbb{N}_0$, is **computationally expected polynomial time (CEPT)**, if there exists a runtime T which is (perfectly) expected polynomial time (i.e. EPT), such that any a priori PPT distinguisher has negligible distinguishing advantage for the distributions T and S. The class of CEPT runtime distributions is denoted $\mathcal{CEPT}$. **Computationally strict polynomial time (CPPT)** is defined analogously.

Characterizing CEPT. At a first glimpse, CEPT looks hard to handle. Fortunately, this is a mirage. We have following characterization of CEPT.

Proposition 1.4 (Simplified[7] Corollary 3.9). *Let T be a runtime. The following are equivalent:*

(0) T is in $\mathcal{CEPT}$.
(1) $\exists S \in \mathcal{EPT}$ which is computationally *PPT-indistinguishable from T.*
(2) $\exists S \in \mathcal{EPT}$ s.t. T and S are statistically indistinguishable (given polynomially many samples).
(3) There is a set of good events $\mathcal{G}_\kappa$ with $\mathbb{P}(\mathcal{G}_\kappa) \geq 1-\varepsilon(\kappa)$ such that $\mathbb{E}(T_\kappa|\mathcal{G}_\kappa) = t_\kappa$ (for the conditional expectation), where ε is negligible and t is polynomial.

Let T be a runtime. Item (3) defines **virtually expected time** (t, ε) with *virtual expectation* t and *virtuality* ε. Thus, the characterization says that computational, statistical and virtual EPT coincide.

Proposition 1.4 follows essentially from the statistical-to-computational reduction and a variant of Lemma 1.2. Thanks to this characterization, working with CEPT is feasible. One uses item (1) to justify that indistinguishability transitions preserve CEPT. And one relies on item (3) to simplify to the case of EPT, usually in unconditional transitions, such as efficiency of rewinding.

An intrinsic characterization. The full Corollary 3.9 not only reveals that CEPT is "well-behaved". It also shows that the runtime class $\mathcal{CEPT}$ is "closed under indistinguishability": Any runtime S which is CEPT-indistinguishable from some $T \in \mathcal{CEPT}$ lies in $\mathcal{CEPT}$. This intrinsic property sets it apart from EPT. (Indeed, $\mathcal{CEPT}$ is the closure of $\mathcal{EPT}$.) PPT and CPPT behave analogously.

Example 1.5. Let A be an algorithm which outputs 42 in exactly 10^{10} steps, and let A' act identical to A, except with probability $2^{-\kappa}$, in which case it runs $2^{2\kappa}$ steps. Then A' is neither PPT nor EPT. Yet, A and A' are indistinguishable even given *timed* black-box access. That is, observing both output and runtime of the black-box, it is not possible to tell A and A' apart. Thus, it is rather unexpected that A' is considered inefficient. For many properties, e.g. correctness or soundness, statistical relaxations from "perfect" exist. CPPT and CEPT should be viewed as such relaxations for efficiency.

Working with CEPT. Applying the characterization of CEPT to a whole system $\langle \mathcal{P}, \mathcal{V}^* \rangle$, the good event $\mathcal{G}$ may induce arbitrary stochastic dependencies on (internal) random coins of the parties. This is inconvenient. We are interested only in one party, namely $\mathcal{V}^*$. Moreover, in a simulation, there is no $\mathcal{P}$ anymore and the probability space changed, hence there is no event $\mathcal{G}$. To account for this, we observe that only the messages $\mathcal{V}^*$ receives from $\mathcal{P}$ are relevant for $\mathcal{V}^*$'s behaviour, not $\mathcal{P}$'s internal randomness. We formulate a convenience lemma (Lemma 3.12) for handling this. Roughly Lemma 3.12 states that for interacting algorithms $\langle \mathsf{A}, \mathsf{B} \rangle$, there is a modification B' (which need not be efficiently computable), which immediately aborts "bad executions" by sending `timeout`. If the closed system $\langle \mathsf{A}, \mathsf{B} \rangle$ is CEPT, i.e. $\mathsf{time}_{\mathsf{A}+\mathsf{B}}(\langle \mathsf{A}, \mathsf{B} \rangle)$ is CEPT, the probability for `timeout` is negligible. Then, by construction, $\mathsf{time}_{\mathsf{B}'}(\langle \mathsf{A}, \mathsf{B}' \rangle)$ will be EPT. This makes B' into a convenient tool to track the evolution

[7] Formally, "triple-oracle" instead of "standard" indistinguishability is used. Assuming non-uniform advice, or runtimes T, S which are induced by algorithms, the simplified definition is equivalent to the actual one.

of runtime and virtuality under actions such as rewinding. By using B' only via oracle-access, its possible inefficiency poses no problems. After the (runtime) analysis, oracle-access to B' is replaced with B again. Importantly, B' is just a means to reason about changes in runtime when applying rewinding to B. One can also reason without introducing B', by using the analysis in Lemma 3.12 directly.

1.4.3. Definitions and tools for zero-knowledge

For uniform auxiliary input zero-knowledge, the input $(x, w, \textit{aux}, \textit{state}) \leftarrow \mathcal{I}(\kappa)$ is efficiently generated by an *input generator* $\mathcal{I}$. A designated adversary $(\mathcal{I}, \mathcal{V}^*)$ consists of input generation, malicious verifier, and distinguisher, but we leave $\mathcal{I}$ often implicit. The distinguisher receives *out* and *state*, the latter is needed for modular sequential composition.[8] Here, $\textit{out} = \mathsf{out}_{\mathcal{V}^*}\langle \mathcal{P}(x,w), \mathcal{V}^*(x, \textit{aux})\rangle$ or $\textit{out} = \mathsf{out}_{\mathsf{Sim}}\mathsf{Sim}(\mathsf{code}(\mathcal{V}^*), x, \textit{aux})$, where $(x, w, \textit{aux}, \textit{state})$ is sampled by $\mathcal{I}(\kappa)$. As a shorthand, for the system which lets $\mathcal{I}$ sample inputs and passes them as above, we write $\langle \mathcal{P}, \mathcal{V}\rangle_{\mathcal{I}}$. From designated CEPT adversaries, we require that $\mathsf{time}_{\mathcal{I}+\mathcal{P}+\mathcal{V}^*+\mathcal{D}}((\textit{state}, \mathsf{out}_{\mathcal{V}^*}\mathcal{P}(x,w), \mathcal{V}^*(x, \textit{aux})))$ is CEPT.

Concrete example. Recall that in Section 1.2, we showed zero-knowledge of the graph 3-colouring protocol $\mathrm{G3C}_{\mathrm{GK}}$ of Goldreich and Kahan [GK96] as follows:

Step 1: Introduce all rewinding steps as in G_1. Here, virtually expected runtime and virtuality at most doubles. To see this, one can use Lemma 3.12 to "replace" $\mathcal{V}^*$ with an modified $\mathcal{V}'$ which yields an EPT execution and outputs `timeout` for "bad" queries. Since Game G_1 at most doubles the probability that some query *query* is asked, bad queries are only twice as likely, i.e. virtuality at most doubles. It is easy to see that the virtually expected runtime also (at most) doubles.

Step 2: Apply indistinguishability transitions, which reduce to hiding resp. binding properties of the commitment. From this, we obtain both good output quality and efficiency of Sim. Concretely, indistinguishability and efficiency follow by an application of the standard reduction (to PPT).

We abstract this strategy to cover a large class of zero-knowledge proofs.[9] Intuitively, we apply the ideas of [Gol10] ("normality") and [KL08] ("query indistinguishability"), but separate the unconditional part (namely, that rewinding preserves efficiency), and the computational part (namely, that simulated queries preserve efficiency).[10]

Abstracting Step 1 (Rewinding strategies). A **rewinding strategy** RWS has black-box rewinding (bb-rw) access to a malicious verifier $\mathcal{V}^*$, and abstracts a simulator's rewinding behaviour. Unlike the simulator, RWS has access to the witness. For RWS to be **normal**, we impose three requirements.

Firstly, a normal rewinding strategy outputs an adversarial view which is *distributed (almost) as in the real execution.* Secondly, there is some poly so that

$$\mathbb{E}(\mathsf{time}_{\mathsf{RWS}+\mathcal{V}^*}(\mathsf{RWS}^{\mathcal{V}^*})) \leq \mathsf{poly}(\kappa) \cdot \mathbb{E}(\mathsf{time}_{\mathcal{P}+\mathcal{V}^*}(\langle \mathcal{P}, \mathcal{V}^*\rangle))$$

for any adversary $\mathcal{V}^*$. We call this (polynomial) **runtime tightness** of RWS.[11] Thirdly, RWS has (polynomial) **probability tightness**, which is defined as follows: Let $\mathsf{pr}_{\mathrm{rws}}(\textit{query})$ be the probability that RWS asks $\mathcal{V}^*$ a query *query*. Let $\mathsf{pr}_{\mathrm{real}}(\textit{query})$ be the probability that the prover $\mathcal{P}$ asks *query*. Then RWS has probability tightness poly if for all queries *query*

$$\mathsf{pr}_{\mathrm{rws}}(\textit{query}) \leq \mathsf{poly}(\kappa) \cdot \mathsf{pr}_{\mathrm{real}}(\textit{query}).$$

Intuitively, runtime tightness ensures that RWS preserves EPT, whereas probability tightness bounds the growth of virtuality. Indeed, the virtuality δ in $\langle \mathcal{P}, \mathcal{V}^*\rangle$ increases to at most $\mathsf{poly} \cdot \delta$ in $\mathsf{RWS}^{\mathcal{V}^*}$. This

[8] While [Gol93] passes no extra *state*, only sequential *repetition* is proven there.

[9] Strictly speaking, we concentrate on zero-knowledge *arguments*, since we need efficient provers.

[10] We significantly deviate from [KL08] to obtain simpler reductions. See Appendix F for an approach similar to [KL08].

[11] Up to minor technical details, polynomial runtime tightness of RWS coincides with "normality" of Sim in [Gol10, Def. 6].

follows because the probability for a "bad" query (a `timeout` of the modified $\mathcal{V}'$ from Lemma 3.12) in $\mathsf{RWS}^{\mathcal{V}^*}$ is at most poly-fold higher than in $\langle \mathcal{P}, \mathcal{V}^* \rangle$.

Lemma 1.6 (Informal Lemma 6.5). *Let* RWS *be a normal rewinding strategy for* $(\mathcal{P}, \mathcal{V})$ *with runtime and probability tightness* poly. *Let* $(\mathcal{I}, \mathcal{V}^*)$ *be an adversary. If* $\langle \mathcal{P}, \mathcal{V}^* \rangle_{\mathcal{I}}$ *is CEPT with virtually expected time* (t, ε)*, then* $\mathsf{RWS}(\mathcal{V}^*)$ *composed with* $\mathcal{I}$ *is CEPT with virtually expected time* $(\mathsf{poly} \cdot t, \mathsf{poly} \cdot \varepsilon)$.

(Weak) relative efficiency. We generalize the guarantees of rewinding strategies to *relative efficiency* of (oracle) algorithms. An oracle algorithm B is **efficient relative to** A with **runtime tightness** $(\mathsf{poly}_{\mathsf{time}}, \mathsf{poly}_{\mathsf{virt}})$ if for all oracles $\mathcal{O}$: If $\mathsf{time}_{\mathsf{A}+\mathcal{O}}(\mathsf{A}^{\mathcal{O}})$ is virtually expected (t, ε)-time, then $\mathsf{time}_{\mathsf{B}+\mathcal{O}}(\mathsf{B}^{\mathcal{O}})$ is virtually expected $(\mathsf{poly}_{\mathsf{time}} \cdot t, \mathsf{poly}_{\mathsf{virt}} \cdot \varepsilon)$-time.

We call B **weakly efficient relative** to A, if whenever $\mathsf{time}_{\mathsf{A}+\mathcal{O}}(\mathsf{A}^{\mathcal{O}})$ is efficient (e.g. CEPT), then $\mathsf{time}_{\mathsf{B}+\mathcal{O}}(\mathsf{B}^{\mathcal{O}})$ is efficient (e.g. CEPT).

Abstracting Step 2 (Simple assumptions). A "**simple" assumption** is a pair of efficiently computable oracles $\mathcal{C}_0$ and $\mathcal{C}_1$, and the assumption that $\mathcal{C}_0 \overset{c}{\approx} \mathcal{C}_1$, i.e. $\mathcal{C}_0$ and $\mathcal{C}_1$ cannot be distinguished in PPT.[12] For example, hiding resp. binding for commitment schemes are simple assumptions.

In Step 2, we reduce the indistinguishability of $\mathsf{RWS}^{\mathcal{V}^*}$ and $\mathsf{Sim}^{\mathcal{V}^*}$ to a simple assumption. That is, there is some algorithm R such that $\mathsf{RWS}^{\mathcal{V}^*} \equiv \mathrm{R}^{\mathcal{C}_0}(\mathcal{V}^*)$, and $\mathrm{R}^{\mathcal{C}_1}(\mathcal{V}^*) \equiv \mathsf{Sim}^{\mathcal{V}^*}$. Moreover, we assume that $\mathrm{R}^{\mathcal{C}_0}(\mathcal{V}^*)$ is efficient relative to $\mathsf{RWS}^{\mathcal{V}^*}$, and $\mathsf{Sim}^{\mathcal{V}^*}$ is efficient relative to $\mathrm{R}^{\mathcal{C}_1}(\mathcal{V}^*)$.

Putting it together (Benign simulators). Black-box simulators whose security proof follows the above outline are called **benign**. See Fig. 1 for an overview of properties and their relation.

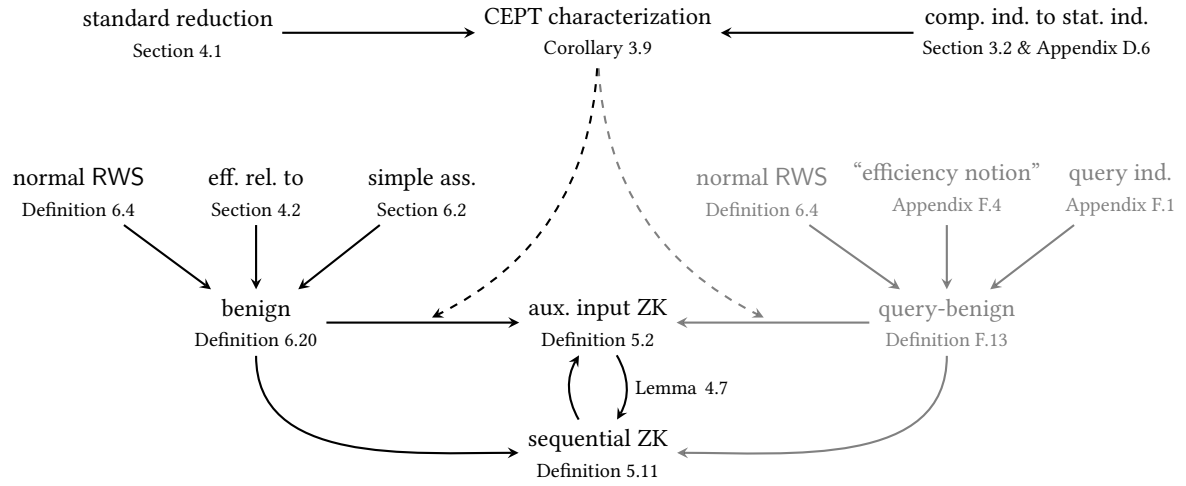

Figure 1: A rough overview of dependencies of core results and definitions. The greyed out approach follows [KL08] more closely. The top line is used everywhere implicitly.

Lemma 1.7 (Informal Lemma 6.23). *Argument systems with* benign *simulators are* auxiliary-input zero-knowledge *against CEPT adversaries.*

Proof summary. The proof strategy above can be summarized symbolically:

$$\mathsf{out}_{\mathcal{V}^*}\langle \mathcal{P}, \mathcal{V}^* \rangle \equiv \mathsf{RWS}(\mathcal{V}^*) \equiv \mathrm{R}^{\mathcal{C}_0}(\mathcal{V}^*) \overset{c}{\approx} \mathrm{R}^{\mathcal{C}_1}(\mathcal{V}^*) \equiv \mathsf{Sim}(\mathcal{V}^*).$$

[12] Technically, our definition of "simple assumption" corresponds to falsifiable assumptions [Nao03] in the sense of [GW10]. We deliberately do not call them falsifiable, since our proof techniques should extend to a larger class of assumptions, which includes non-falsifiable assumptions.

More precisely, consider a CEPT adversary $(\mathcal{I}, \mathcal{V}^*)$. By normality of RWS, $\mathsf{out}_{\mathcal{V}^*}\langle \mathcal{P}, \mathcal{V}^* \rangle$ and $\mathsf{RWS}(\mathcal{V}^*)$ have (almost) identical output distributions, and $\mathsf{RWS}(\mathcal{V}^*)$ is CEPT. By relative efficiency, $\mathrm{R}^{\mathcal{C}_0}(\mathcal{V}^*)$ is CEPT if $\mathsf{RWS}^{\mathcal{V}^*}$ is CEPT. Since $\mathcal{C}_0 \overset{c}{\approx} \mathcal{C}_1$, by a standard reduction, if $\mathrm{R}^{\mathcal{C}_0}(\mathcal{V}^*)$ is CEPT, so is $\mathrm{R}^{\mathcal{C}_1}(\mathcal{V}^*)$, and their outputs are indistinguishable. Finally, since $\mathsf{Sim}^{\mathcal{V}^*}$ is efficient relative to $\mathrm{R}^{\mathcal{C}_1}(\mathcal{V}^*)$, also $\mathsf{Sim}^{\mathcal{V}^*}$ is CEPT. All in all, $\mathsf{Sim}^{\mathcal{V}^*}$ is efficient and produces indistinguishable outputs. □

Benign simulators are common, e.g. the classic, constant round, and concurrent zero-knowledge protocols in [GMW86; GK96; Lin13; Ros04; KP01; PTV14] satisfy this property.

1.4.4. Sequential composition and hybrid arguments

It turns out, that the sequential composition theorem and the hybrid argument in general are non-trivial in the setting of a posteriori efficiency.

Intermezzo: Tightness bounds. The use of relative efficiency with polynomial tightnesss bounds is not strictly necessary. Nevertheless, it offers "more quantifiable" security and is easier to handle. For example, benign simulators are easily seen to "compose sequentially" because, (1) normal RWS and relative efficiency compose sequentially, and (2) "simple" assumptions satisfy indistinguishability under "repeated trials". Together, this translates to sequential composition of benign simulation. Hence, argument systems with *benign* simulators are *sequential zero-knowledge* against CEPT adversaries. Unfortunately, the general case is much more involved.

The hybrid lemma. To keep things tidy, we consider an abstract hybrid argument, which applies to zero-knowledge simulation and much more. Due to a posteriori efficiency, the lemma is both non-trivial to prove and non-trivial to state.

Lemma 1.8 (Lemma 4.7). *Let $\mathcal{O}_0$ and $\mathcal{O}_1$ be two oracles and suppose that $\mathcal{O}_1$ is weakly efficient relative to $\mathcal{O}_0$ and $\mathcal{O}_0 \overset{c}{\approx} \mathcal{O}_1$. Denote by $\mathsf{rep}(\mathcal{O}_0)$ and $\mathsf{rep}(\mathcal{O}_1)$ oracles which give repeated access to independent instances of $\mathcal{O}_b$. Then $\mathsf{rep}(\mathcal{O}_1)$ is weakly efficient relative to $\mathsf{rep}(\mathcal{O}_0)$ and $\mathsf{rep}(\mathcal{O}_0) \overset{c}{\approx} \mathsf{rep}(\mathcal{O}_1)$.*

Lemma 1.8 hides much of the complexity caused by a posteriori efficiency, and is often a suitable black-box drop-in for the hybrid argument. We sketch how to adapt the usual hybrid reduction. In our setting, $\mathsf{rep}(\mathcal{O}_b)$ gives access to arbitrarily many independent instances of $\mathcal{O}_b$. The usual hybrids H_i use $\mathcal{O}_1$ for the first i instances, and switch to $\mathcal{O}_1$ for all other instances. W.l.o.g., only $q = \mathsf{poly}(\kappa)$ many $\mathcal{O}$-instances are accessed by $\mathcal{D}$. The hybrid distinguisher $\mathcal{D}'$ guesses an index $i^* \leftarrow \{0, \dots, q-1\}$, and simulates a hybrid H_{i+b} embedding its challenge oracle $\mathcal{O}_b^*$.

If $\mathcal{D}$ has advantage ε, then the hybrid distinguisher $\mathcal{D}'$ has advantage ε/q. In the classic PPT setting, we assume that $\mathcal{O}_0$ and $\mathcal{O}_1$ are classical PPT, and hence find that $\mathcal{D}'$ is PPT and therefore efficient. In an a posteriori setting, the efficiency of $\mathcal{D}'$ is a bigger hurdle. We make the minimal assumptions, that $\mathsf{time}_{\mathcal{D}+\mathsf{rep}(\mathcal{O}_0)}(\mathcal{D}^{\mathsf{rep}(\mathcal{O}_0)})$ is efficient and that $\mathcal{O}_1$ is efficient relative to $\mathcal{O}_0$.[13] Hence, we do not trivially know whether $\mathsf{time}_{\mathcal{D}+\mathsf{rep}(\mathcal{O}_1)}(\mathcal{D}^{\mathsf{rep}(\mathcal{O}_1)})$ or the hybrid distinguisher $\mathcal{D}'$, which has to emulate many oracle instances, is efficient. Indeed, a naive argument would invoke weak relative efficiency q times. In the case of PPT, this would mean q-many polynomial bounds. But, for all we know, these could have the form $2^i\mathsf{poly}(\kappa)$ in the i-th invocation, leading to an inefficient simulation.

The core problem is therefore to avoid a superconstant application of relative efficiency.[14] Essentially this problem was encountered by Hofheinz, Unruh, and Müller-Quade [HUM13] in the setting of universal composability and a posteriori PPT. They provide a nifty solution, namely to *randomize*

[13] The hybrid proof technique requires the hybrid distinguisher to emulate all but one oracle instance, and for this we need weak relative efficiency.

[14] For reference, even for a priori PPT sequential composition for zero-knowledge, one must avoid a superconstant invocation of the existence of simulators. There, the solution is to consider a "universal" adversary and its "universal" simulator.

the oracle indexing. This ensures that, in each hybrid, every emulation of $\mathcal{O}_0$ (resp. $\mathcal{O}_1$) has identical runtime distribution T_0 (resp. T_1). This is gives a uniform bound on runtime changes. Now, we show how to extend the proof of [HUM13], which is limited to CPPT.

We prove the hybrid argument in game hops, starting from the real protocol G_1. In G_2, we replace *one* oracle instance of $\mathcal{O}_0$ by $\mathcal{O}_1$ (at a random point). In G_3, every instance of $\mathcal{O}_0$ but one is replaced by $\mathcal{O}_1$. In G_4, only $\mathcal{O}_1$ is used. Since $\mathcal{O}_1$ is weakly efficient relative to $\mathcal{O}_0$ and $\mathcal{O}_0 \overset{c}{\approx} \mathcal{O}_1$, the transitions from G_1 to G_2 (resp. G_3 to G_4) preserve efficiency and are indistinguishable. The step from G_2 to G_3 is the crux. Note that we have at least one $\mathcal{O}_0$ (resp. $\mathcal{O}_1$) instance in either game. Take any one and denote the time spent in that instance by T_0 (resp. T_1). Since we randomized the instances, the distribution of T_0 (resp. T_1) does not depend on the concrete instance. Importantly, even in the hybrid *reduction*, there is an instance which can be used to compute T_0 (resp. T_1). Moreover, the total time spent in computing instances of $\mathcal{O}_0$ and $\mathcal{O}_1$ is "dominated"[15] by $q \cdot T_0 + q \cdot T_1$. Thus, it suffices to prove that $S = T' + T_0 + T_1$ is CEPT, where T' is the time spent outside emulation of instances of $\mathcal{O}_0$ and $\mathcal{O}_1$. (Note that S, T', T_0, T_1 depend on the hybrid H_ℓ, where $\ell \in \{1, \ldots, q-1\}$, we suppressed this dependency.) Now, we have two properties:

- S_ℓ is CEPT if and only if $\mathsf{time}(\mathsf{H}_\ell)$ is CEPT for the ℓ-th hybrid H_ℓ.
- The reduction can compute and output S_ℓ.

Thus, it suffices that S_1 and S_{q-1} are indistinguishable, since we know that S_1 is CEPT. Curiously, we now reduced efficiency to indistinguishability.[16] To prove indistinguishability, we can truncate the reduction (or rather, the hybrids) to strict PPT as in the standard reduction. Thus, we obtain $S_1 \overset{c}{\approx} S_q$. The hybrid lemma follows. The actual reasoning of this last step is a bit lengthier, but follows [HUM13] quite closely: We truncate each oracle separately to maintain symmetry of `timeout` probabilities. Unfortunately, the reduction does not give the usual telescoping sum, since the challenge oracle cannot be truncated. Due to symmetry, the error is "dominated" by observed `timeouts`. Hence, it suffices to find a (uniform) bound for the `timeout` probabilities over all H_ℓ. Our reasoning for this is mildly more complex than [HUM13], since we do not have negligible bounds for `timeouts`, but only polynomial tail bounds, and we make a weaker assumption on efficiency of $\mathcal{O}_0$ and $\mathcal{O}_1$.

Modular sequential composition. With Lemma 1.8 at hand, it is straightforward to prove that auxiliary input zero-knowledge composes sequentially. In fact, it is possible to prove a modular sequential composition theorem similar to [KL08]. (In [KL08], the subprotocols must have simulators which are EPT *in any interaction*. In our setting, there is no such restriction and the straightforward proof works. The bulk of the complexity is absorbed by the hybrid lemma.)

1.5. Related work

We are aware of three (lines of) related works w.r.t. EPT: The results by Katz and Lindell [KL08] and those of Goldreich [Gol10], both focused on cryptography. And the relaxation of EPT for average-case complexity by Levin [Lev86]. A general difference of our approach is, that we treat the security parameter separate from input sizes, whereas [KL08; Gol10] assume $\kappa = |x|$.[17] With respect to a posteriori runtime, [HUM13] is a close analogue, although for PPT and in the UC setting.

Comparison with [KL08]. Katz and Lindell [KL08] tackle the problem of expected polynomial time by using a *superpolynomial runtime cutoff*. They show that this cutoff guarantees a (strict) EPT adversary. However, for the superpolynomial cutoff, they need to *fix* one superpolynomial function α and have to assume security of primitives w.r.t. (strict) α-time adversaries. Squinting hard enough, their

[15] To be exact, dominated with slack q: $\mathbb{P}(\mathsf{time}_{\mathcal{O}_0+\mathcal{O}_1}(\mathsf{H}_\ell) > t) \leq q \cdot \mathbb{P}(q(T_{\ell,0} + T_{\ell,1}) > t)$.

[16] The CEPT characterization (Corollary 3.10) does not strictly apply here, but a simple variation does.

[17] For completeness, we show how to mirror this weakened security in Appendix E.4.3.

approach is dual to ours. Instead of assuming superpolynomial security and doing a cutoff, we "ignore negligible events" in runtime statistics, thus doing a "cutoff in the probability space". Moreover, we require no *fixed* bound.

Interestingly, their first result [KL08, Theorem 5] holds for "adversaries which are EPT w.r.t. the real protocol". Their notion is minimally weaker than ours, as it requires efficiency of the adversary *for all inputs* instead of a sequence of input distributions.[18] [KL08, Section 3.5] claims that other scenarios, e.g. sequential composition, fall within [KL08, Theorem 5]. Their *modular* sequential composition theorem, however, requires that subprotocol simulators are "expected polynomial time *in any interaction*", which is *not* implied by [KL08, Theorem 5].

Comparison with [Gol10]. Goldreich [Gol10] strengthens the notion of expected polynomial time to obtain a complexity class which is stand-alone and suitable for rewinding based proofs. He requires *expected polynomial time w.r.t. any reset attack*, hence restricts to "nice" adversaries. With this, normal (in the sense of [Gol10]) black-box simulators run in expected polynomial time, essentially by assumption. This way of dealing with designated adversaries is far from the spirit of our work.

Comparison with [Lev86]. The relaxation of expected polynomial time adopted by Levin [Lev86] and variations [Gol11b; Gol10; BT06] are very strong. Let T be a runtime distribution. One definition requires that for some poly and $\gamma > 0$, $\mathbb{P}(T_\kappa > C) \leq \frac{\mathsf{poly}(\kappa)}{C^\gamma}$ for all κ and all $C \geq 0$. Equivalently, $\mathbb{E}(T_\kappa^\gamma)$ is polynomially bounded (in κ) for some $\gamma > 0$. Allowing negligible "errors" relaxes the notion further. This definition fixes the composition problems of expected polynomial time. But arguably, it stretches what is considered efficient far beyond what one may be willing to accept. Indeed, runtimes whose expectation is "very infinite" are considered efficient.[19] The goals of average case complexity theory and cryptography do not align here. We stress that our approach, while relaxing expected polynomial time, is far from being so generous, see Section 1.6.1. (For completeness, we note that we are not aware of work on designated adversaries in this setting.)

Related work on CPPT. The notion of CPPT is (in different forms) used and well-known. For example, Boneh and Shoup [BS20] rely on such a notion. This sidesteps technical problems, such as sampling uniformly from $\{0, 1, 2\}$ with binary coins. With a focus on complexity theory, Goldreich [Gol11a] defines *typical efficiency* similar to CPPT. As the relaxations for strict bounds is very straightforward, we suspect more works using CPPT variations for a variety of reasons.

Comparison with [HUM13]. Hofheinz, Unruh, and Müller-Quade [HUM13] define *PPT with overwhelming probability (w.o.p.)*, i.e. CPPT, and consider a posteriori efficiency. They work in the setting of universal composability (UC), and their main focus is an overall sensible notion of runtime, which does not artificially restrict evidently efficient *functionalities*, such as databases or bulletin boards. Their notion of efficiency is similar to our setting with CPPT. In fact, we use their techniques for the hybrid argument. Since [HUM13] defines and assumes *protocol efficiency*, which we deliberately neglect, there are some differences. Reinterpreting [HUM13], their approach is based on: "If *for all* (stand-alone) efficient $\mathcal{D}$ the machine $\mathcal{D}^{\mathcal{O}_0}$ is efficient, then *for all* (stand-alone) efficient $\mathcal{D}$ the machine $\mathcal{D}^{\mathcal{O}_1}$ is efficient."[20] Our approach is based on: "*For all* $\mathcal{D}$, if the machine $\mathcal{D}^{\mathcal{O}_0}$ efficient, then the machine $\mathcal{D}^{\mathcal{O}_1}$ is efficient." The stronger (protocol) efficiency requirements are harder to justify in our setting. (Even classical PPT $\mathcal{O}_0$ can be "inefficient" for *expected* poly-size inputs. E.g., disallowing quadratic time protocols seems harsh.)

[18] Their definitions are a consequence of their non-uniform security definition and complexity setting. The proof of [KL08, Theorem 5] never changes adversarial inputs, so there is no obstruction to handling designated adversaries in our sense.

[19] Setting $c = 2$ and $\gamma = 3$ in Remark 1.9 yields a runtime T with $\mathbb{E}(T) = \sum_{n=1}^{\infty} n$, which is still considered efficient. (The limit $-\frac{1}{12}$ is not applicable here.)

[20] Think of $\mathcal{D}$ as the environment, $\mathcal{O}_0$ as the protocol, and $\mathcal{O}_1$ as the simulator.

More related work. Halevi and Micali [HM98] define a notion of efficiency for extractors in proofs of knowledge, which closely resembles our notion of normal rewinding strategies. Precise zero-knowledge [MP06; Pas06] requires that simulation and real execution time are closely related. Due to Feige's "attack" (or Example 1.1), this does not seem to help with designated EPT adversaries.

1.6. Separations

We briefly provide separations between some runtime notions. Here, we focus only on efficiency of adversaries, and *ignore* requirements imposed on protocol efficiency, since we deliberately neglected those. We consider *basic runtime classes* (i.e. runtimes of sampling algorithms) and how they are *lifted to interactive algorithms*.

Both [KL08, Definition 1] and [HUM13, Definitions 1 and 2] use an "a posteriori" lifting. The former lifts EPT, the latter lifts CPPT; both allow designated adversaries and are similar to our setting. "A priori" liftings, such as [Gol10, Definitions 1–4] are far more restrictive (on adversaries), effectively disallowing designated adversaries.

Regarding the underlying runtime classes, the works [KL08; Gol10] deal with (perfect) EPT, negligible deviations are not allowed. The notion of PPT w.o.p. from [HUM13] and CPPT coincide. To separate PPT, EPT, CPPT, CEPT, and Levin's relaxations, we first recall fat-tailed distributions.

Remark 1.9 (Fat-tailed distributions). The sum $\sum_n n^{-c}$ is finite if and only if $c > 1$. Thus, we obtain a random variable X with $\mathbb{P}(X = n) \propto n^{-c}$. For $\gamma > 0$ we have $\mathbb{E}(X^\gamma) \propto \sum_n n^{-c+\gamma}$. If $c-\gamma \leq 1$, then $\mathbb{E}(X^\gamma) = \infty$. Moreover, $\mathbb{P}(X \geq k) \geq k^{-c}$, i.e. X has **fat tails**. In particular, for $c = 3$, $\mathbb{E}(X) < \infty$ but $\mathbb{E}(X^2) = \sum_n n^{-1} = \infty$, and $\mathbb{P}(X \geq \mathsf{poly}) \geq \frac{1}{\mathsf{poly}^3}$ for any poly.

Allowing a negligible deviation clearly separates perfect runtime distributions from their computational counterparts. Clearly, PPT is strictly contained in EPT. The separation of CPPT and CEPT follows from fat-tailed distributions. In Section 1.6.1 below, we separate CEPT from Levin's relaxations of EPT, denoted $\mathcal{LT}$, and Vadhan's relaxation of $\mathcal{LT}$, denoted $\mathcal{VT}$, which allows negligible deviation. In the following diagram, *strict* inclusions are denoted by arrows.

$$\begin{array}{ccccc} \mathcal{PPT} & \longrightarrow & \mathcal{EPT} & \longrightarrow & \mathcal{LT} \\ \downarrow & & \downarrow & & \downarrow \\ \mathcal{CPPT} & \longrightarrow & \mathcal{CEPT} & \longrightarrow & \mathcal{VT} \end{array}$$

1.6.1. Levin's relaxation and CEPT

We noted in Remark 1.9, that $\sum_{n=1}^{\infty} n^{-c} < \infty$ for $c > 1$ gives rise to a distribution Z_c over $\mathbb{N}$ via normalizing the sum. Let $X = Z_2^3$. Then $\mathbb{E}(X) = \sum_{n=1}^{\infty} n = \infty$. Since Z_2 is fat-tailed, so is X. Let $Y_k = X|_{(\cdot \geq k) \mapsto 0}$. It follows immediately that $\mathbb{E}(Y_k) = \mathbb{E}(X|_{(\cdot \geq k) \mapsto 0}) \geq \frac{1}{2}k^{2/3}$ for any $k \in \mathbb{N}$. Thus, for any superpolynomial cutoff K, we find $\mathbb{E}(Y_K) \geq \frac{1}{2}K^{2/3}$ is superpolynomial, and as a consequence, there is no superpolynomial cutoff which makes X EPT. (We interpret X (and Y_K) as a constant family of runtimes, i.e. $X_\kappa = X$ for all κ.)

Formally, CEPT uses ν-quantile cutoffs (i.e. we may condition on an event $\mathcal{G}$ of overwhelming probability $1 - \nu$ that minimizes $\mathbb{E}(T \mid \mathcal{G})$). For X, any ν-quantile cutoff for negligible ν induces some bound k which maximizes $\mathbb{P}(T \leq k) \geq \nu$. If k were polynomial, then (due to "fat tails") ν must also be polynomial. Hence, k must be superpolynomial, and consequently there is no negligible quantile cutoff which makes X EPT. All in all, the runtime distribution X is allowed by Levin's relaxation, but is not CEPT.

1.7. Structure of the paper

In Section 2, we clarify preliminaries, such as (non-)standard (notational) conventions, shorthands and terminology, and some basic concepts and results. In Section 3, we define CEPT and prove the characterization as well as generalizations and convenience lemmas. In Section 4, introduce the standard reduction, relative efficiency and the hybrid lemma. In Section 5, we apply CEPT to zero-knowledge. We define (uniform complexity auxiliary input) zero-knowledge, and consider the example of $\mathrm{G3C}_{\mathrm{GK}}$ in detail. Then, we define sequential zero-knowledge and prove that it is implied by auxiliary input zero-knowledge. In Section 6, we define rewinding strategies, simple assumptions, and benign simulation, Moreover, we give a simple proof that benign simulators are (sequential) zero-knowledge. In Section 7, we sketch the application of CEPT to (uniform complexity) multiparty computation. In Section 8, we conclude and highlight some open questions.

In Appendix A, we give a detailed discussion on the effect of machine models and their (in)compatibility with expected time. Appendix B contains supplementary definitions for commitment schemes. The remaining appendices contain further material and discussion. Appendix C contains some simple but useful results and reminders for our general discussion of runtime classes. Appendix D treats runtime more abstractly. It justifies the notion of "closed runtime classes" formally and demonstrates how most of our results extend to algebra-tailed runtime class. In Appendix E, we discuss asides for each chapter, and more. These provide clarifications, justify decisions, technical details, effects of variations in definitions, give (simple) examples, and so on. Finally, for completeness, we show in Appendix F that our approach is applicable even if we follow the work of Katz and Lindell [KL08] much more closely, although at the expense of more convoluted proofs.

See page 94 for the table of contents.

2. Preliminaries

In this section, we state some basic definitions and (non-)standard conventions.

2.1. Notation and basic definitions

We denote the security parameter by κ; it is often suppressed. Similarly, we often speak of an object X, instead of a family of objects $(X_\kappa)_\kappa$ parameterized by κ. We always assume binary encoding of data, unless explicitly specified otherwise.[21] We write $X \sim Y$ if a random variable X is distributed as Y. For random variables X, Y over a set A We write $X|_{a\mapsto b}$ (resp. $X|_{S\mapsto b}$, resp. $X|_{\mathsf{pred}\mapsto b}$) for the random variable where a (resp. any a satisfying $a \in S$ resp. $\mathsf{pred}(a) = 1$) is mapped to b, and everything else unchanged, e.g. $X|_{\perp\mapsto 0}$ or $X|_{S\mapsto 0}$ or $X|_{.\geq N\mapsto N}$.

For a countable set $\mathcal{S}$ and a function $\phi\colon \mathcal{S} \to \mathbb{R}$, let $\|\phi\|_p := (\sum_{x\in\mathcal{S}}|\phi(x)^p|)^{1/p}$ be the p-norms for $p \in [1, \dots, \infty]$. (Recall that $\|\phi\|_\infty := \sup_{x\in\mathcal{S}}|\phi(x)|$.) We define statistical distances $\Delta_p(\rho, \sigma) := \frac{1}{2}\|\rho - \sigma\|_p$ of distributions $\rho, \sigma\colon \mathcal{S} \to [0, 1]$. Recall that $\Delta_1(\rho, \sigma) = \sup_{X\subseteq\Omega}|\rho(X) - \sigma(X)|$. We refer to the variational distance $\Delta(\cdot, \cdot) := \Delta_1(\cdot, \cdot)$ as the **statistical distance**.

We call $\mathsf{D}_{\mathsf{rat}}(\rho/\sigma) := \sup_x \frac{\rho(x)}{\sigma(x)}$ (where $\frac{0}{0} := 0$) the **sup-ratio** of ρ over σ; ρ and σ may be arbitrary non-negative functions.

With poly, polylog, and negl we denote polynomial, polylogarithmic and negligible functions (in κ) respectively. Usually, we (implicitly) assume that poly, polylog, and negl are *monontone*. A function negl is (polynomially) negligible if $\lim_{\kappa\to\infty} \mathsf{poly}(\kappa)\mathsf{negl}(\kappa) = 0$ for every polynomial poly. In many definitions, we assume the existence of a negligible bound negl on some advantage $\varepsilon = \varepsilon(\kappa)$. We generally use "strict pointwise $\leq$" for bounds, e.g. $\varepsilon \leq \mathsf{negl}$ denotes $\forall\kappa\colon \varepsilon(\kappa) \leq \mathsf{negl}(\kappa)$. We avoid

[21] In classical efficiency settings, unary encoded data is primarily used to model efficiency restrictions implicitly. We model these explicitly, and, due to a posteriori, efficiency depends only on κ anyway. It is irrelevant if κ is passed as binary or unary to the machines, hence we use binary encodings unless otherwise specified.

"eventually $\leq$", denoted $\varepsilon \leq_{\text{ev}} \mathsf{negl}$ (defined via $\exists C \forall \kappa > C \colon \varepsilon(\kappa) \leq \mathsf{negl}(\kappa)$). If $\varepsilon \leq_{\text{ev}} \mathsf{negl}$, then $\max\{\varepsilon(\kappa), \mathsf{negl}(\kappa)\} =: \nu(\kappa)$ is negligible and $\varepsilon \leq \nu$, hence this makes no difference in most situations. However, "$\leq$" behaves "more intuitively" than "$\leq_{\text{ev}}$" in some sense.[22]

2.2. Systems, algorithms, interaction and machine models

More detailed discussion of (unexplained) terms in this section are in Appendix A.

Machine models. We fix some **admissible** machine model, which in particular implies that emulating a system of interacting machines has small overhead. The reader may assume a RAM model without much loss. In particular, polylogarithmic (emulation) overhead is acceptable in our setting, see. Appendix A.4.[23] Another irksome technicality are non-halting computations. One may follow [Gol10], and assume all algorithms halt after a finite number $n(\kappa)$ of steps. Instead, we deal with non-halting executions explicitly. For this, we define the symbol `nohalt` as the "output" of such a computation, and assume that any system which receives `nohalt` also outputs `nohalt`, if not specified otherwise.

Systems, algorithms and oracles. We always consider (induced) systems, which offer **interfaces** for (message-based) communication.[24] Input and output are modelled as interfaces as well. The security parameter κ is an implicit input interface of (almost) every system; a system is **closed** if its only interfaces are for κ and output, i.e. it is a "sampling algorithm" (which takes κ and samples some output). A **system** is a "mathematical" object, which defines (probabilistic) behaviour of the offered interfaces. An **algorithm** is given by *code*, a *finite*[25] string describing the behaviour and interfaces, and has a notion of runtime and randomness interface (e.g. random tape) which are imparted on it by the machine model. **Oracles** or **parties** are, unless stated otherwise, algorithms, which are only used via their interface. To emphasize availability of a certain oracle to some algorithm, we speak of **oracle algorithms**. A **timed** oracle offers an extended interface to its caller, which allows to bound the maximum time spent in an invocation (and return `timeout` if the allotted time is exceeded), and also returns the elapsed time of any invocation. Oracles also serve as a means to make **subroutine calls** explicit. A **timeful** oracle (or system) comes with some notion of *purported elapsed runtime*. For consistency, the purported elapsed runtime is always at least the answer length of an invocation, and this is usually also the runtime notion of interest. Timeful oracles (or systems) are used as convenience abstractions to specify and analyze unconditional properties. Timed timeful are defined in the obvious way.

Interaction. It will always be clear from the context how interfaces are used or connected. Interactivity is implicit, and implied by open interfaces. Let $\mathsf{A}_1, \mathsf{A}_2$ be a algorithms (or more generally, systems). For connecting A_1 and A_2, i.e. interaction, with (fixed) inputs x, y, z, we write $\langle \mathsf{A}_1(x,z), \mathsf{A}_2(y,z) \rangle$. The result is another algorithm (or system), where we write $\mathsf{out}_{\mathsf{A}_i}\langle \mathsf{A}_1, \mathsf{A}_2 \rangle$ for the output (interface) of A_i for $i = 1, 2$. We write $\mathsf{A}^{\mathcal{O}}$ for an algorithm (or system) A, with access to an oracle $\mathcal{O}$ (where $\mathcal{O}$ may be a subroutine, e.g. a commitment scheme). This notation emphasizes, that the output of the system is that of A. Otherwise, the system is equivalent to $\langle \mathsf{A}, \mathcal{O} \rangle$, or even $\mathcal{O}^{\mathsf{A}}$. We view interaction, oracle, and subroutine calls as essentially identical and use the notation interchangeably if no confusion arises.

[22] When infinitely many functions are considered, $\leq$ and $\leq_{\text{ev}}$ behave differently. For $\leq_{\text{ev}}$, any countable set of negligible functions is $\leq_{\text{ev}}$-dominated by some negl, c.f. [Bel02]. This is false for $\leq$. Indeed, $\leq_{\text{ev}}$ behaves unintuitive. Consider a sum of a growing number (in κ) of negligible functions ν_i. It is well-known that $\mu(\kappa) := \sum_{i=1}^{\kappa} \nu_i(\kappa)$ need not be negligible, even if all ν_i are negligible. But if all ν_i are "strictly dominated" by some ν, i.e. $\nu_i \leq \nu$, then $\mu(\kappa) \leq \kappa\nu(\kappa)$ hence μ is negligible. However, if all ν_i are only "*eventually* dominated", i.e. $\nu_i \leq_{\text{ev}} \nu$, then the standard counterexample ($\nu_i(j) = 1$ if $i = j$ and 0 else) shows that μ need not be negligible. Concretely, $\nu = 0$ *eventually* dominates all ν_i, yet $\mu(n) = 1 > 0 = n\nu(n)$. Due to this behaviour, we avoid "$\leq_{\text{ev}}$".

[23] More precisely, CEPT is robust w.r.t. polylogarithmic overhead, due to virtuality. For robustness of EPT, an additional strict a priori runtime bound is needed, e.g. $2^{\mathsf{poly}(\kappa)}$ works.

[24] We use an ad-hoc definition of system. A compatible, precise notion was recently (concurrently) introduced in [LM20].

[25] Non-uniform notions deviate here and allow infinite descriptions.

Black-box rewinding (bb-rw) access to an algorithm A (or timeful system) means access to an oracle $\mathsf{bbrw}(\mathsf{A})$ emulating A with fresh but fixed randomness, which allows to feed A messages and rewind it to any visited state. For notational simplicity, we treat $\mathsf{bbrw}(\mathsf{A})$ like a $\mathsf{NextMsg}_{\mathsf{A}}$ function, which upon a query $\mathit{query} = (m_1, \ldots, m_n)$ returns the result of A when given m_i as its i-th message. The query $(m_1, \ldots, m_n)$ is viewed as a *logical handle* $(m_1, \ldots, m_{n-1})$ to a previously visited state, and a message m_n to A when in that state. Implementations of $\mathsf{bbrw}(\mathsf{A})$ use *short* handles, say a counter. A **timed** bb-rw oracle truncates and returns the elapsed runtime of its emulated program. By abuse of notation, we often write B^{A} instead of $\mathsf{B}^{\mathsf{bbrw}(\mathsf{A})}$ if it is clear that B has bb-rw access to A.

Remark 2.1 (Efficient implementations). Access to $\mathsf{NextMsg}_{\mathsf{A}}$ and $\mathsf{bbrw}(\mathsf{A})$ is "*logically equivalent*", yet, the efficiency characteristics differ vastly. For expected time, this is a critical point. We encounter such issues also in other situations, and will offer a brief warning but proceed with the usual notation. Using more efficient "logically equivalent" implementations solves such problems. See Appendix A.3.

2.3. Input generation: Conventions and shorthands

In *non*-uniform complexity settings, it is possible to quantify over all inputs to a protocol universally. In uniform complexity settings [Gol93], these inputs must be *efficiently samplable*. For this, we use efficient algorithm, usually denoted $\mathcal{I}$, called the **input generator**. For non-uniform security, $\mathcal{I}$ is non-uninform, i.e. has tape-like access to an (unbounded) non-uniform advice string advc_κ. This deviates from standard definitions [Gol01] slightly by allowing input *distributions*.

Notation 2.2 (Shorthand expressions for composing systems). Let $\mathcal{P}, \mathcal{V}^*$ be two (interacting) parties and let $\mathcal{I}$ be an input generator. We use the shorthand notation $\langle \mathcal{P}, \mathcal{V}^* \rangle_{\mathcal{I}}$ for the system resp. interaction of $\langle \mathcal{P}, \mathcal{V} \rangle$ completed with $\mathcal{I}$, where it is either clear how to connect the interfaces or it is explicitly described. We also say: "Let $\mathsf{out}_{\mathcal{V}^*}\langle \mathcal{P}(x, w), \mathcal{V}^*(x, \mathit{aux}) \rangle$, where $(x, w, \mathit{aux}) \leftarrow \mathcal{I}(\kappa)$."

What we mean by this is: Consider the system obtained by composing $\mathcal{I}$, $\mathcal{P}$ and $\mathcal{V}^*$ as indicated, that is, the system which first runs $\mathcal{I}$ to obtain (x, w, aux), then passes (x, w) to $\mathcal{P}$ as input, and passes (x, aux) to $\mathcal{V}^*$, and then runs $\mathcal{P}$ and $\mathcal{V}^*$ (i.e. letting them interact). Of this composed system, take and return the output of $\mathcal{V}^*$.

Note that we do *not* mean to quantify over all inputs (x, w, aux) which $\mathcal{I}$ may produce, except if made explicit, e.g. by stating "*for all* $(x, w, \mathit{aux}) \leftarrow \mathcal{I}$" or more precisely "*for all* $(x, w, \mathit{aux}) \in \mathsf{supp}(\mathcal{I})$". Since we almost exclusively consider closed systems, and fixed inputs make little sense in a uniform asymptotic setting, no confusion should arise.

2.4. Preliminary remarks on runtime

An abstract treatment of runtime is in Appendix D, and meant for the inclined reader only. This section contains all essential definitions for Section 3 and later sections, which only deal with polynomial times, namely PPT, EPT, CPPT and CEPT.

For an oracle algorithm A, we write $\mathsf{time}_{\mathsf{A}}(\mathsf{A}^{\mathcal{O}})$ for the time spent in A (called **oracle-excluded time**), $\mathsf{time}_{\mathcal{O}}(\mathsf{A}^{\mathcal{O}})$ for the time spent in $\mathcal{O}$, and $\mathsf{time}_{\mathsf{A}+\mathcal{O}}(\mathsf{A}^{\mathcal{O}})$ for the time spent in both (called **oracle-included time**). This notation extends naturally to interaction and composite systems built from interacting machines. Note that $T = \mathsf{time}_{\mathsf{A}}(\mathsf{A}^{\mathcal{O}})$ is a *random variable*, or more precisely, a sequence of random variables T_κ parameterized by κ. We assume that that runtimes sum up, i.e. $\mathsf{time}_{\mathsf{A}}(\mathsf{A}^{\mathcal{O}}) + \mathsf{time}_{\mathcal{O}}(\mathsf{A}^{\mathcal{O}}) = \mathsf{time}_{\mathsf{A}+\mathcal{O}}(\mathsf{A}^{\mathcal{O}})$, *as dependent random variables*.

Definition 2.3. A **runtime (distribution)** T is a family of random variables (resp. distributions) over $\mathbb{N}_0$ parameterized by the security parameter κ. We (only) view a runtime as a random variable $T_\kappa \colon \Omega_\kappa \to \mathbb{N}_0$, when stochastic dependency is relevant.

Definition 2.4. A **runtime class** $\mathcal{T}$ is a set of runtime distributions.[26] An algorithm A is $\mathcal{T}$**-time** if $\mathsf{time}_{\mathsf{A}}(\mathsf{A}) \in \mathcal{T}$.

[26] For our general treatment of runtimes, we use a more restrictive definition, c.f. Appendix D.3.

Example 2.5. The runtime classes $\mathcal{PPT}$ and $\mathcal{EPT}$ of strict polynomial time (PPT) and expected polynomial time (EPT) are defined in the obvious way, i.e.: $T \in \mathcal{PPT}$ (resp. $T \in \mathcal{EPT}$) if there exists a polynomial poly such that $\mathbb{P}(T_\kappa > \mathsf{poly}(\kappa)) = 0$ (resp. $\mathbb{E}(T_\kappa) \leq \mathsf{poly}(\kappa)$).

Our central tool for dealing with expected time is truncation. Also recall that timed oracles abstract the ability to truncate executions.

Definition 2.6 (Runtime truncation). Let A be an algorithm. We define $\mathsf{A}^{\leq N}$ as the algorithm which executes A up to N steps, and then returns A's output. If A did not finish in time, $\mathsf{A}^{\leq N}$ returns `timeout`.

A priori time, a posteriori time, and designated adversaries. In any closed system, every component has an associated random variable, describing the time spent in it. We only consider such runtimes (most often, the total runtime). Hence, efficiency depends only on κ, since closed systems have no (other) input. In particular, we do not assign a stand-alone notions of efficiency or runtime to a non-closed system, e.g. an algorithm A which still needs inputs (or oracle access, or communication partners). The exception to the rule are *a priori PPT* (resp. *a priori EPT*) algorithms A, for which there is a bound poly such that $\mathsf{time}_{\mathsf{A}}(\dots) \leq \mathsf{poly}$ (resp. $\mathbb{E}(\mathsf{time}_{\mathsf{A}}(\dots)) \leq \mathsf{poly}$) for any choice of inputs, oracles, and communication partners.[27]

A posteriori efficiency of algorithms (or systems) considers them in a complete context, i.e. as part of a closed system. Let A be an algorithm and $\mathcal{E}$ be an environment such that $\langle \mathcal{E}, \mathsf{A} \rangle$ is a closed system. For **a posteriori** time, there are two sensible definitions: We can call A a posteriori PPT (resp. EPT, ...) w.r.t. $\mathcal{E}$, if $\mathsf{time}_{\mathsf{A}}(\langle \mathcal{E}, \mathsf{A} \rangle)$ is PPT (resp. EPT, ...), or if $\mathsf{time}_{\mathcal{E}+\mathsf{A}}(\langle \mathcal{E}, \mathsf{A} \rangle)$ is PPT (resp. EPT, ...). We generally use the latter, but are always explicit about it. Applied to security notions, we get **designated adversaries**, which need only be *efficient for the protocol they are designed to attack*, see [Fei90] or [KL08; Gol10].

2.5. Probability theory

By $\mathsf{Dists}(X)$ we denote the space of probability distributions on X. The underlying probability space for random variables is usually denoted by Ω, the associated σ-algebra is always left implicit. We neglect measurability questions because they do not pose any problems and are merely trivial technical overhead, see Appendix E.8 for a brief discussion.

We allow product extension of Ω to suit our needs, say extending to $\Omega' = \Omega \times \Sigma$ with Bernoulli distribution $\mathsf{Ber}(\frac{1}{3})$ on $\Sigma = \{0, 1\}$. Random variables over Ω are lifted implicitly and we again write Ω instead of Ω'. Let $\mathbb{N}_0 \cup \{\infty, \mathtt{timeout}\}$ be totally ordered via $n < \infty < \mathtt{timeout}$ for all $n \in \mathbb{N}_0$. For $X \colon \Omega \to \mathbb{R}$, if $\mathbb{P}(X > c) < \mathsf{tail}(c)$, we call tail a **tail bound**. For families $X_\kappa \colon \Omega \to \mathbb{R}$, we sometimes sloppily call t_κ a *tail bound* for c_κ if $\mathbb{P}(X_\kappa > c_\kappa) < t_\kappa$. We denote the cumulative density function (CDF) of X by $\mathsf{CDF}_X(c) = \mathbb{P}(X \leq c)$ and let $\overline{\mathsf{CDF}}_X(\,\cdot\,) \coloneqq 1 - \mathsf{CDF}_X(\,\cdot\,) = \mathbb{P}(X > \cdot)$.

For convenience, we use a relaxation of stochastic domination.

Definition 2.7 (Domination with slack). Let $X, Y \colon \Omega \to \mathcal{S}$ be random variables and $\mathcal{S}$ be a totally ordered set (usually $\mathcal{S} = \mathbb{R} \cup \{\mathtt{timeout}\}$). Let $L \geq 1$. We say Y **dominates** X **with slack** L (in distribution), if $\overline{\mathsf{CDF}}_X \leq L \cdot \overline{\mathsf{CDF}}_Y$, that is, if

$$\forall c \in \mathcal{S} \colon \quad \mathbb{P}(X > c) \leq L \cdot \mathbb{P}(Y > c).$$

We denote this by $X \overset{d}{\leq}_L Y$. If $L = 1$, we write $X \overset{d}{\leq} Y$. We use the same notation for families of random variables, i.e. we write $X \overset{d}{\leq} Y$ and mean $X_\kappa \overset{d}{\leq} Y_\kappa$ for all κ.

Instead of truncating runtimes in the domain, we often "truncate" in the probability space.

[27] By definition, a priori PPT is the essentially same as a priori PPT in any interaction of [KL08; Gol10], but in our setting where only the security parameter grants runtime. Note that "classical" PPT algorithms are not a priori PPT in our sense, since their runtime bound depends on the input size, while ours are fixed by κ alone. We can mitigate this discrepancy by size-guarding (see Appendix E.4.3).

Definition 2.8 (ν-quantile cutoff). Let T be a distribution on $\mathbb{N}_0 \cup \{\infty\}$ and $\nu > 0$. Suppose that $\mathbb{P}(T = \infty) \leq \nu$.[28] The (exact) **$\nu$-quantile (cutoff)** T^ν is following distribution on $\mathbb{N}_0 \cup \texttt{timeout}$. Let $\mathsf{CDF}_T(\,\cdot\,)\colon \mathbb{N}_0 \cup \{\infty\} \to [0,1]$ be the CDF of T. Then $\mathsf{CDF}_{T^\nu}(\,\cdot\,)\colon \mathbb{N}_0 \cup \texttt{timeout} \to [0,1]$ is defined by $\mathsf{CDF}_{T^\nu}(n) = \min\{1-\nu, \mathsf{CDF}_T(n)\}$ for $n \in \mathbb{N}$, and $\mathsf{CDF}_{T^\nu}(\infty) = \lim_{n\to\infty} \min\{1-\nu, \mathsf{CDF}_T(n)\}$, hence $\mathbb{P}(T^\nu = \infty) = 0$, and $\mathsf{CDF}_{T^\nu}(\texttt{timeout}) = 1$,

An exact ν-quantile cutoff for a random variable $T\colon \Omega \to \mathbb{N}_0 \cup \{\infty\}$ can be constructed by: First pick $N = \inf\{n \mid \mathbb{P}(T > n) \leq \nu\}$. If $\mathbb{P}(T > N) =: \nu'$ equals ν, let $T^\nu := T|_{>N \mapsto \texttt{timeout}}$. Else, pick a (measurable) subset of $A = \{\omega \in \Omega \mid T(\omega) = N\}$ of probability $\nu - \nu'$, and let $T^\nu := T|_{A \mapsto \texttt{timeout}}$. If necessary, modify Ω. So we assume w.l.o.g. that there is such a set of events. An *approximate ν-quantile cutoff* with *error* δ is an exact ν'-quantile cutoff, where $\nu \leq \nu' \leq \nu + \delta$.

In case of discrete distributions, one can find a unique maximal (measurable) subset A (e.g. minimal by lexicographic order), and a unique atomic event which may have to be split between N and $\texttt{timeout}$. By modifying Ω to $\Omega \times \{0,1\}^n$, an approximate cutoff with error at most to 2^{-n} is possible. Using $\Omega \times \mathsf{Ber}(\nu - \nu')$, exact cutoffs are possible.

Remark 2.9 (Equal-unless). If $X, Y\colon \Omega \to \mathcal{S}$ are random variables and coincide (as functions), except for an event $\mathcal{E} \subseteq \Omega$, then X and Y are **(pointwise) equal unless** $\mathcal{E}$. Typically, $\mathcal{E} = \{\omega \mid Y(\omega) = \texttt{bad}\}$ (for some symbol $\texttt{bad}$), and we say X equals Y unless $\texttt{bad}$ happens. We also say X and Y *coincide unless* (or *agree except*) if $\texttt{bad}$ happens. The definition naturally extends to oracles and systems.

Remark 2.10 (Truncation of values vs. quantiles). Consider random variables X, Y over $\mathbb{R}$ with $X \overset{d}{\leq}_L Y$ (for some $L \geq 1$). As seen in Lemma C.7, quantile-truncation preserves domination even if we additionally condition on $\neg\texttt{timeout}$. Truncating in the domain does *not* preserve domination if we additionally condition on $\neg\texttt{timeout}$. For example, over $\{1,2,3,4\}$ consider the probability vectors $p_X \mathrel{\widehat{=}} (\beta, 0, 1-\beta, 0)$ and $p_Y \mathrel{\widehat{=}} (0, \alpha, 0, 1-\alpha)$. Truncating X, Y at 3 and conditioning on $\neg\texttt{timeout}$ yields X', Y' with $p_{X'} = p_X$ and $p_{Y'} = (0,1,0)$, and thus $X' \overset{d}{\not\leq}_L Y'$, even for $L = 1$.

2.6. Indistinguishability and oracle-related notions

We define (oracle-)indistinguishability, repeated trials, and query sequences.

2.6.1. Oracle-indistinguishability

The (in)distinguishability of oracles (or systems) is a folklore abstraction. "Bit-guessing" experiments, such as indistinguishability of distributions, and more generally game-based security notions can be straightforwardly rephrased as an oracle pair, see Appendix E.1.3. Depending on the oracles (or systems) and their interfaces, distinguishing can encompass (adversarial) input generation, protocol runs, and more. For example, an oracle may present an IND-CPA game for public key encryption, or it may present the distinguisher with a concurrent zero-knowledge setting.

Definition 2.11 (Oracle-indistinguishability). Let $\mathcal{O}_0$ and $\mathcal{O}_1$ be (not necessarily computable) oracles with identical interfaces. A distinguisher $\mathcal{D}$ is a system which connects to all interfaces or $\mathcal{O}_0$, $\mathcal{O}_1$, resulting in a closed system $\mathcal{D}^{\mathcal{O}_b}$. The **(standard) distinguishing advantage** of $\mathcal{D}$ is defined by

$$\mathrm{Adv}^{\mathrm{dist}}_{\mathcal{D},\mathcal{O}_0,\mathcal{O}_1}(\kappa) = |\mathbb{P}(\mathcal{D}^{\mathcal{O}_1}(\kappa) = 1) - \mathbb{P}(\mathcal{D}^{\mathcal{O}_0}(\kappa) = 1)|.$$

By abuse of notation, we sometimes abbreviate $\mathrm{Adv}^{\mathrm{dist}}_{\mathcal{D},\mathcal{O}_0,\mathcal{O}_1}$ by $\mathrm{Adv}^{\mathrm{dist}}_{\mathcal{D},\mathcal{O}}$.

Let $\mathcal{T}$ be a runtime class. Then $\mathcal{O}_0$ and $\mathcal{O}_1$ are **computationally (standard) indistinguishable in $\mathcal{T}$-time**, written $\mathcal{O}_0 \overset{c}{\approx}_{\mathcal{T}} \mathcal{O}_1$ if for any $\mathcal{T}$-time distinguisher $\mathcal{D}$ with $\mathsf{time}_{\mathcal{D}}(\mathcal{D}^{\mathcal{O}_b(\kappa)}(\kappa)) \in \mathcal{T}$ (for $b = 0, 1$)[29]

[28] It is straightforward to deal with general $\nu \geq 0$. But distributions S over $\mathbb{N}_0 \cup \{\infty\} \cup \texttt{timeout}$ with $\mathbb{P}(S = \infty) > 0$ are not particularly useful for us.

[29] This is equivalent to being efficient in the respective distinguishing experiment.

there is some negligible negl such that $\mathrm{Adv}^{\mathrm{dist}}_{\mathcal{D},\mathcal{O}}(\kappa) \leq \mathsf{negl}$. We define **statistical** $\mathcal{T}$-query indistinguishability by counting only oracle-queries as runtime.

Perfect indistinguishability is special, and we reserve the notation "$\equiv$" for it.

Definition 2.12. Oracles $\mathcal{O}_0$, $\mathcal{O}_1$ (or systems, or algorithms), for which all (unbounded) distinguishers have advantage 0 are called **perfectly indistinguishable**. We also write $\mathcal{O}_0 \equiv \mathcal{O}_1$ to emphasize this.

Remark 2.13 (Indistinguishability of distributions). Indistinguishability of distributions X and Y (under repeated samples) is defined in the natural compatible way, namely via oracles $\mathcal{O}_X$ and $\mathcal{O}_Y$ which output a single (a fresh) sample of X resp. Y (for each query).

2.6.2. Repeated trials

It is useful to make repeated oracle access explicit.

Definition 2.14 (Repeated oracle access). Let $\mathcal{O}$ be an oracle. We denote by $\mathsf{rep}(\mathcal{O})$ an oracle which offers repeated access to *independent instances* of $\mathcal{O}$. For example, $\mathsf{rep}(\mathcal{O})$ may implement this by expecting message tuples (i, m) of oracle index i and query m, and a special message which starts a new independent copy of $\mathcal{O}$, increasing the maximal admissible index i by 1. We denote by $\mathsf{rep}_q(\mathcal{O})$ an oracle which limits access to a total of at most q instances of $\mathcal{O}$. (Effectively, the admissible indices are $1, \ldots, q$. Also observe that $\mathsf{rep}(\mathcal{O}) = \mathsf{rep}_\infty(\mathcal{O})$.)

Definition 2.15 (Indistinguishability under repeated trials). Let $\mathcal{O}_0$ and $\mathcal{O}_1$ be two oracles. We say $\mathcal{O}_0$ and $\mathcal{O}_1$ are **$\mathcal{T}$-time computationally indistinguishable under q (repeated) trials**, if $\mathsf{rep}_q(\mathcal{O}_0)$ and $\mathsf{rep}_q(\mathcal{O}_1)$ are $\mathcal{T}$-time computationally indistinguishable. We say $\mathcal{O}_0$ and $\mathcal{O}_1$ are **$\mathcal{T}$-time indistinguishable under (unbounded many) repeated trials**, if they are $\mathcal{T}$-time indistinguishable for $q = \infty$ repeated trials. The definition for **$\mathcal{T}$-query statistically indistinguishable** is analogous.

2.6.3. Query-sequences

We use following definition and notation for the sequence of queries made by an algorithm to its oracle.

Definition 2.16 (Query-sequence). Let $\mathsf{A}^{\mathcal{O}}$ be an oracle algorithm. The **query-sequence** $\mathsf{qseq}_{\mathcal{O}}(\mathsf{A}^{\mathcal{O}}(x))$ is the (distribution of the) sequence of queries made by A to $\mathcal{O}$. We view $\mathsf{qseq}_{\mathcal{O}}(\mathsf{A}^{\mathcal{O}}(x))$ as an oracle, which grants lazy (tape-like) access to the queries.

3. Computationally expected polynomial time

In this section, we define computationally expected polynomial time (CEPT), briefly recap the general results of Appendix D for polynomial runtime classes, and have a first glimpse of the behaviour of CEPT. The inclined reader may wish to continue with Appendix D instead; it deals with runtime classes in more generality.

3.1. A brief recap

3.1.1. Virtually expected time

We are interested in properties, which need only hold with overwhelming probability. We formalize this for the expectation of non-negative random variables as follows.

Definition 3.1 (Virtual expectation). Let $X \colon \Omega \to \mathbb{R}_{\geq 0} \cup \{\infty\}$ Let $\varepsilon > 0$. We say X has **ε-virtual expectation (bounded by)** t if

$$\exists \mathcal{G} \subseteq \Omega \colon \mathbb{P}(\mathcal{G}) \geq 1 - \varepsilon \ \wedge \ \mathbb{E}(X \mid \mathcal{G}) \leq t$$

We extend this to families by requiring it to hold component-wise. Moreover, we say a runtime T is ε-**virtually** t-**time** if T has ε-virtual expectation bounded by t. We abbreviate this as **virtually expected** (t, ε)-**time** and call ε the **virtuality** of time (t, ε).

Virtual properties have a "probably approximately" flavour. They are closely related to "ε-smooth properties", such as ε-smooth min-entropy, which smudge over statistically close random variables (instead of conditioning).[30] Virtual properties must behave well under restriction (up to a certain extent).

Lemma 3.2. *Let $X \colon \Omega \to \mathbb{R}_{\geq 0}$ be a random variable and $\mathbb{E}(X) = t$. Then any restriction of X to an event $\mathcal{G}$ of measure $1 - \varepsilon$ implies $\mathbb{E}(X \mid \mathcal{G}) \leq (1 - \varepsilon)^{-1} t$.*

The upshot of Lemma 3.2 is that, if we condition on an *overwhelming* (in fact, noticeable) event $\mathcal{G}$, polynomially bounded expectation is preserved. Also, consecutive restrictions of Ω are unproblematic.

3.1.2. Triple-oracle indistinguishability

Using *triple-oracle indistinguishability*, instead of standard indistinguishability, for (runtime) distributions abstracts technical details and prevents technical problems. Recall that we always use *binary* encodings, and this includes runtime oracles (even though unary encodings work there without change).

Definition 3.3. A **triple-oracle distinguisher** $\mathcal{D}$ for distributions X_0, X_1, receives access to three oracles $\mathcal{O}_0$, $\mathcal{O}_1$ resp. $\mathcal{O}_b^*$, which sample according to some distributions X_0, X_1, resp. X_b. The distinguishing advantage is $\mathrm{Adv}^{\text{3-dist}}_{\mathcal{D},\mathcal{O}_0,\mathcal{O}_1} = |\mathbb{P}(\mathcal{D}^{\mathcal{O}_0,\mathcal{O}_1,\mathcal{O}_1^*}(\kappa)) - \mathbb{P}(\mathcal{D}^{\mathcal{O}_0,\mathcal{O}_1,\mathcal{O}_0^*}(\kappa))|$.

Two runtime distributions T, S are **computationally $\mathcal{T}$-time triple-oracle indistinguishable**, denoted by $T \overset{c}{\approx}_{\mathcal{T}} S$, if any $\mathcal{T}$-time distinguisher has advantage $o(1)$. If $\mathcal{T}$ contains $\mathcal{PPT}$, then (by amplification) any distinguisher has negligible advantage. For statistical triple-oracle indistinguishability, we only count oracle queries as steps (and often explicitly speak of statistical $\mathcal{T}$-query distinguishers)[31] and write $T \overset{s}{\approx}_{\mathcal{T}} S$.

A runtime class $\mathcal{T}$ is **computationally closed** if for all runtimes S, if there exists some $T \in \mathcal{T}$ such that $T \overset{c}{\approx}_{\mathcal{T}} S$, then $S \in \mathcal{T}$. **Statistically closed** is defined analogously.

In the definition, we sketched our approach for general runtime classes (namely requiring $o(1)$ advantage bound, see Appendix D). This definition applies to runtime classes from other algebras, such as polylog or quasi-polynomial time, and implicitly uses the notion of negligible function for these algebras. The use of tail bounds as our proof technique seems limited to the setting, where "advantage" and "time" algebras coincide. From now on, we specialize to the polynomial setting, where amplification enforces negligible advantage.

Triple-oracle distinguishing should be interpreted as distinguishing with repeated samples, plus sampling access to the distributions X_0, X_1. It allows for quite modular reductions, as we see now.

Remark 3.4 (Standard and triple-oracle indistinguishability). To clearly distinguish triple-oracle and "normal" indistinguishability, we call the latter *standard* when in doubt. We use (and defined) triple-oracle indistinguishability only for (runtime) distributions, not for general oracles.

3.2. Characterizing CEPT

We begin with the fundamental definition of this section.

[30] We borrowed the terminology of virtual properties from group theory.

[31] We *never* consider unbounded queries for statistical triple-oracle distinguishing, as this trivially coincides with perfect indistinguishability.

Definition 3.5 (CEPT and CPPT). The runtime class $\mathcal{CEPT}$ of **computationally expected polynomial time** contains all runtimes which are (triple-oracle) $\mathcal{PPT}$-time indistinguishable from expected polynomial time. In other words: A runtime T is CEPT if there is an EPT $\widetilde{T}$, such that T and $\widetilde{T}$ are triple-oracle $\mathcal{PPT}$-time indistinguishable, i.e. $T \overset{c}{\approx} \widetilde{T}$.

Computationally (strict) probabilistic polynomial time is defined analogously and denoted $\mathcal{CPPT}$.

Now, we turn towards the characterization of CEPT. We start with a few simple lemmata. Their central technique is to approximate probability distributions with suitable precision, and then use this information for distinguishing.

Lemma 3.6. *Suppose S and T are runtimes and $T \in \mathcal{CEPT}$. Then statistical $\mathcal{CEPT}$-query and computational $\mathcal{CEPT}$-time triple-oracle indistinguishability coincide, i.e. $S \overset{c}{\approx}_{\mathcal{PPT}} T \iff S \overset{s}{\approx}_{\mathcal{PPT}} T$. Moreover, a priori PPT distinguishers are sufficient.*

Proof sketch. It is clear that statistical indistinguishability implies computational indistinguishability. Thus, we concentrate on the converse. For $T \in \mathcal{CEPT}$ there exists, by definition, some $\widetilde{T} \in \mathcal{EPT}$ such that $T \overset{c}{\approx} \widetilde{T}$ (triple-oracle *computational* indistinguishability). Hence, for any (efficiently computable) $N = N(\kappa)$, we have $|\mathbb{P}(T > N) - \mathbb{P}(\widetilde{T} > N)| \leq \mathsf{negl}$.

We show that T and $\widetilde{T}$ are *statistically* triple-oracle indistinguishable as well. Assume the statistical distance $\Delta(T, \widetilde{T})$ is at least $\delta = \frac{1}{\mathsf{poly}_0}$ infinitely often. Note that $\mathbb{P}(\widetilde{T} > N) \leq \frac{\mathsf{poly}_1}{N}$, where $\mathbb{E}(\widetilde{T}) \leq \mathsf{poly}_1$. Thus, by truncating $T, \widetilde{T}$ after, say $N = 4\mathsf{poly}_0\mathsf{poly}_1$, we know that $T^{\leq N}$ and $\widetilde{T}^{\leq N}$ are distributions with *polynomial support in* $\{0, \ldots, N\}$ and *non-negligible statistical distance* $\frac{\delta}{4}$ infinitely often. Since we have (repeated) sample access to $T, \widetilde{T}$ and the challenge runtime, we can approximate the probability distributions (by the empirical probabilities) up to any $\frac{1}{\mathsf{poly}}$ precision in polynomial time, see Appendix C.4. Consequently, we can construct a (computational) PPT distinguisher if T and $\widetilde{T}$ are not statistically $\mathcal{PPT}$-query indistinguishable.

The described statistical-to-computational distinguisher works for T and S as well. Let $\delta = \Delta(T, S)$. Since $T \in \mathcal{CEPT}$, there is a suitable tail bound N with $\Delta(T, T^{\leq N}) \leq \frac{\delta}{4}$. It is easy to see that $\Delta(T^{\leq N}, S^{\leq N}) \geq \frac{\delta}{4}$.[32] If $\delta \geq \frac{1}{\mathsf{poly}}$ infinitely often, then there is a suitable polynomial N, such $\Delta(T^{\leq N}, S^{\leq N}) \geq \frac{\delta}{4}$ infinitely often. Thus, we are in the same setting as before, and can distinguish by approximation. Lastly, note that the distinguisher we constructed is a priori PPT. □

The proof of Lemma 3.6 also shows closedness of $\mathcal{CEPT}$.

Corollary 3.7. *Let T, S be two runtimes and $T \in \mathcal{CEPT}$. Then $S \overset{c/s}{\approx}_{\mathcal{PPT}} T \iff S \overset{c/s}{\approx}_{\mathcal{CEPT}} T$, i.e. statistical $\mathcal{PPT}$-query and computational $\mathcal{PPT}$-time triple-oracle indistinguishability coincide.*

Thus, we have shown that all relations $\overset{m}{\approx}_{\mathcal{T}}$ for $m \in \{c, s\}$, $\mathcal{T} \in \{\mathcal{PPT}, \mathcal{CEPT}\}$ coincide. For concrete applications, we want to use standard indistinguishability instead of triple-oracle indistinguishability whenever possible.

Lemma 3.8. *Let T and S be runtimes induced by algorithms* A, B*, and suppose $T \in \mathcal{CEPT}$. Then triple-oracle and standard $\mathcal{PPT}$-time indistinguishability coincide, i.e. $S \overset{c/s}{\approx}_{\mathcal{PPT}} T \iff S \overset{c/s}{\approx}_{\mathcal{PPT}} T$.*

Proof sketch. Suppose T and S are triple-oracle distinguishable with advantage at least $\delta = \frac{1}{\mathsf{poly}_0}$ infinitely often. The distinguisher $\mathcal{D}'$ from the proof of Lemma 3.6 is a priori PPT with advantage $\frac{\delta}{4}$ infinitely often. Moreover, $\mathcal{D}'$ *truncates all samples* at *polynomial* N, i.e. $\mathcal{D}$ actually distinguishes $T^{\leq N}$ and $S^{\leq N}$. These truncated runtime distributions can be *sampled via emulation* in strict polynomial

[32] Intuitively, either `timeout` accumulates a difference in probability of $\frac{\delta}{4}$, or a difference of $\frac{\delta}{4}$ in probability is present on $\{0, \ldots, N\}$, see Corollary C.5.

time. By sampling via emulation and a hybrid argument, we find an a priori PPT distinguisher $\mathcal{D}$ with advantage at least $\frac{\delta}{4N}$ infinitely often. □

We stress that to efficiently distinguish two induced runtimes, it is sufficient that *one* of the two algorithms is efficient.[33]

Putting things together yields following convenient characterization of CEPT and CPPT:

Corollary 3.9 (Characterization of CEPT). *Let T be a runtime. The following conditions are equivalent:*

(0) T is in $\mathcal{CEPT}$.
(1) T is $\mathcal{PPT}$-time triple-oracle computationally indistinguishable from some $\widetilde{T} \in \mathcal{EPT}$.
(2) T is $\mathcal{PPT}$-query triple-oracle statistically indistinguishable from some $\widetilde{T} \in \mathcal{EPT}$.
(3) T is virtually expected polynomial time. Explicitly: There is a negligible function negl*, an event $\mathcal{G}$ with $\mathbb{P}(\mathcal{G}) \geq 1 - \mathsf{negl}$, and a polynomial* poly*, such that $\mathbb{E}(T_\kappa \mid \mathcal{G}) \leq \mathsf{poly}(\kappa)$.*

Furthermore, $T \in \mathcal{CEPT}$ satisfies the following tail bound

$$\mathbb{P}(T_\kappa > N) \leq \frac{\mathsf{poly}(\kappa)}{N} + \mathsf{negl}(\kappa)$$

for poly *and* negl *as in (3). Consequently, $\mathcal{CEPT}$ distinguishers are not more powerful than $\mathcal{PPT}$ distinguishers. In particular, $\mathcal{CEPT}$ is a closed runtime class. (In fact, it is the closure of $\mathcal{EPT}$.)*

For induced runtimes $T = \mathsf{time}_{\mathsf{A}}(\mathsf{A})$, $S = \mathsf{time}_{\mathsf{B}}(\mathsf{B})$, where $T \in \mathcal{CEPT}$, and S is arbitrary, computational $\mathcal{CEPT}$-time (resp. statistical $\mathcal{CEPT}$-query) triple-oracle indistinguishability and standard computational (resp. statistical) indistinguishability coincide.

The analogous characterization and properties hold for CPPT.

The essence of Corollary 3.9 is the equivalence of items (1) and (3). The former is easy to prove, as it is follows by reductions to indistinguishability assumptions. The latter is easy to use, as it guarantees that, after ignoring a negligible set of bad events, one can work with perfect EPT.

Proof sketch of Corollary 3.9. Equivalence of items (1) and (2) follows from Lemma 3.6. Now, we show that (2) implies (3). For our triple-oracle notion, being statistically indistinguishable implies being statistically close, as one can see by approximating the probability distribution, as in Lemma 3.6. Say the statistical distance is δ. Let T^ν be the respective ν-quantile of T. Clearly, $T^\varepsilon|_{\texttt{timeout}\mapsto 0}$ minimizes the value of $\mathbb{E}(S)$ under the constraint that S is a non-negative random variable with $\Delta(T, S) \leq \varepsilon$. By assumption, there is some δ-close EPT S. Hence, we have $\mathbb{E}(T^\delta\big|_{\texttt{timeout}\mapsto 0}) \leq \mathsf{poly}$. Consequently $\mathbb{E}(T^\delta \mid \neg\texttt{timeout}) \leq \frac{1}{(1-\delta)}\mathsf{poly}$, and the claim follows.

The converse is trivial: If $\mathbb{E}(T \,|\, \mathcal{G}) \leq \mathsf{poly}$ for an event $\mathcal{G}$ of overwhelming probability $1 - \mathsf{negl}$, then $\widetilde{T} = T|_{\mathcal{G}\mapsto 0}$ is evidently EPT and has statistical distance at most negl. This finishes the equivalence of items (1), (2) and (3).

To see the tail bound, note that for $T \in \mathcal{CEPT}$ there is a "good" runtime $\widetilde{T} \in \mathcal{EPT}$ with $\Delta(T, \widetilde{T}) \leq \mathsf{negl}$. Thus, the tail bound follows immediately from Markov's bound (Lemma C.2) applied to $\widetilde{T}$ and statistical distance of negl. That PPT distinguisher suffice and $\mathcal{CEPT}$ is closed was already shown in 3.7, but follows easily from the tail bound.

Finally, for induced runtimes, Lemma 3.8 demonstrates the equivalence of triple-oracle and standard distinguishing. □

As noted before, non-uniform advice can replace sampling access. For non-uniform distinguishers, triple-oracle and standard indistinguishability coincide. In fact, all the above results follow almost trivially by using the optimal decision table of a distinguisher for $T^{\leq N}$ and $S^{\leq N}$ as advice.

Applications require a further corollary which, though unmotivated, best fits here.

[33] If neither runtime is efficient, we are in a setting where the truncation argument does not work. Indeed, strings can be encoded as numbers, hence runtimes. Thus, this is indistinguishability of general distributions.

Corollary 3.10. *Let* A, B *be two algorithms which output a number in* $\mathbb{N}_0$. *Let* A, B *denote the output distribution and let* $T = \mathsf{time}_{\mathsf{A}}(\mathsf{A})$, $S = \mathsf{time}_{\mathsf{B}}(\mathsf{B})$. *Suppose* $T \overset{d}{\leq_L} q \cdot A$ *and* $S \overset{d}{\leq_L} q \cdot B$ *and let* $L = L(\kappa)$ *and* $q = q(\kappa)$ *polynomial in* κ. *Suppose furthermore* $A \in \mathcal{CEPT}$ *(and hence* $T \in \mathcal{CEPT}$*).*

If $A \overset{c}{\approx} B$ *then* $S \in \mathcal{CEPT}$. *In particular,* $A \in \mathcal{CEPT} \iff B \in \mathcal{CEPT}$ *and statistical and computational (standard and triple-oracle) indistinguishability coincide. The claims generalize to oracle algorithms w.r.t.* $T = \mathsf{time}_{\mathsf{A}}(\mathsf{A}^{\mathcal{O}_A})$, $S = \mathsf{time}_{\mathsf{B}}(\mathsf{B}^{\mathcal{O}_B})$.

The corollary says that, if we measure (and output) a statistic which bounds the runtime (up to polynomial slack), then indistinguishability of that statistic implies preservation of efficiency. This is a core step for the hybrid argument. We stress that the claim is non-trivial, as equivalence of standard and triple-oracle indistinguishability was only proven for *induced* runtimes. Nevertheless, after "rescaling" the tail bound from N to $N \cdot q \cdot L$, the argument is quite analogous to Lemma 3.8.

Proof sketch. By assumption, $A \in \mathcal{CEPT}$, and therefore $T \in \mathcal{CEPT}$ (by Lemma C.7). By the tail bound for CEPT (see Corollary 3.9), for any polynomial $\delta = 1/\mathsf{poly}$, there exists a polynomial N with $\mathbb{P}(A > N) < \delta$. Usually, we would choose such an N for suitable small δ and truncate A and B at N, and argue with a standard cutoff argument. However, we cannot truncate A (resp. B) w.r.t. A (resp. B), since these are not the runtimes, and the standard cutoff argument does not apply. Nevertheless, we have a relation between T (resp. S) and A (resp. B) which we can use. So instead, we truncate at K, where $\mathbb{P}(A > K) < \frac{\delta}{qL}$. Then

$$\overline{\mathsf{CDF}}_T(K) \leq Lq \cdot \overline{\mathsf{CDF}}_A(K) \leq Lq\frac{\delta}{qL} = \delta.$$

Note that $\frac{\delta}{qL}$ is again polynomial. Thus, we can approximate the distribution of A up to precision δ by using $\mathsf{A}^{\leq K}$.

The first inequality also works with S and B, but as in the standard cutoff argument, we do not know if $\overline{\mathsf{CDF}}_A(K) \leq \frac{\delta}{qL}$. Suppose $\Delta(A, B) > \varepsilon = 1/\mathsf{poly}$ infinitely often. Take $\delta = \varepsilon/4$ and fix the respective cutoff K from above. Then $|\mathbb{P}(B > K) - \mathbb{P}(A > K)| \leq \varepsilon/4$ as this yields a distinguisher (and we assume $A \overset{c}{\approx} B$). Hence, $\Delta(A, A') \leq \varepsilon/4$, and $\Delta(B, B') \leq \varepsilon/4$. Thus, we can sample approximations A' (resp. B') of A (resp. B) via $\mathsf{A}^{\leq K}$ (resp. $\mathsf{B}^{\leq K}$) which are $\varepsilon/4$ statistically close. With this, we can retrace the steps of the CEPT characterization, in particular Lemma 3.8, to prove equivalence of triple-oracle and standard indistinguishability. (That is, we approximate the distribution of A, B, and compute from this a decision table for the challenge sample. With non-uniform advice, we can again skip this just the advice contains the optimal decisions.)

□

3.3. From CEPT to EPT

The characterization of CEPT ensures that, conditioning on "good" events yields a strict EPT algorithm. For interacting parties, this is not yet very useful, because it "entangles" their probability spaces.

Example 3.11. Let $\langle \mathcal{P}, \mathcal{V} \rangle$ be an interactive protocol. Suppose $\mathcal{P}$ sends a random message $r \in R$. Suppose $\mathcal{V}$ picks a random number $s \in R$, and if $r = s$, it loops forever. Otherwise the protocol finishes. Now, the bad event is $\{(r, r) \mid r \in R\}$ (or some superset).

This "entanglement" of probability spaces prevents one core separation, namely the random coins of honest and adversarial parties. Fortunately, they can be "disentangled" as far as possible. Namely, only the (distribution of) *messages* of (honest) parties are of relevance, but no internal coin tosses. This essentially follows from the fact, that the interacting systems have "independent" randomness spaces, and the interaction is mediated solely by messages between the systems.

Lemma 3.12 (Timeout oracles). *Let* A *be an interactive algorithm and* $\mathcal{O}$ *be a (probablistic) timeful oracle. Suppose* $\mathsf{time}_{\mathcal{O}}(\langle \mathsf{A}, \mathcal{O} \rangle)$ *is CEPT with virtual runtime* (t, ε)*. Then there exists an oracle* $\mathcal{O}'$*, modelled as a* timeful *oracle, such that:* $\mathcal{O}$ *and* $\mathcal{O}'$ *behave identically except when* $\mathcal{O}'$ *sends* `timeout` *(and halts) to signal bad executions. If* A *aborts upon receiving* `timeout`*, then*[34] $\mathsf{time}_{\mathcal{O}}(\langle \mathsf{A}, \mathcal{O}' \rangle)$ *is EPT with expected runtime* $t + O(1)$ *(with small hidden constant).*[35] *The probability for a* `timeout` *message in* $\langle \mathsf{A}, \mathcal{O}' \rangle$ *is* ε*.*[36]

We stress that $\mathcal{O}'$ is a *timeful* oracle. While the construction shows that $\mathcal{O}'$ is computable from timed bb-rw access to $\mathcal{O}$, it is generally far from efficiently computable. The usage of Lemma 3.12 is roughly as follows: Replace $\mathcal{O}$ with the timeful $\mathcal{O}'$. Now, the runtime problems are is easier to analyse, since we have guaranteed EPT runtime. In the analysis, track the effects on runtime and `timeout` messages of $\mathcal{O}'$. Finally, replace $\mathcal{O}'$ with $\mathcal{O}$ again, noting that only if `timeout` occurs, there is a difference. Of course, such arguments can be made directly, without introducing $\mathcal{O}'$ at all. However, the explicit modification simplifies the presentation.

The construction of $\mathcal{O}'$ is straightforward, one defines $\mathcal{O}'$ by a runtime truncation at N, i.e. $\mathcal{O}'$ acts exactly as $\mathcal{O}$ until the total elapsed time exceeds N. Then, $\mathcal{O}'$ aborts with `timeout`. Exact ν-quantile cutoffs are achieved by extension of $\Omega_{\mathcal{O}}$, as usual.

Proof of Lemma 3.12. A runtime truncation of $\mathcal{O}$ at N is defined in the obvious way, i.e. $\mathcal{O}^{\leq N}$ returns `timeout` if, after an invocation, the purported elapsed runtime exceeds N. An exact ν-quantile cutoff is constructed as usual, i.e. let N be the minimal such that

$$\nu' \coloneqq \mathbb{P}(\langle \mathsf{A}, \mathcal{O}^{\leq N} \rangle \text{ has } \texttt{timeout}) \leq \nu.$$

If this is an equality, let $\mathcal{O}^{\nu}$ be defined as $\mathcal{O}^{\leq N}$. Else, extend $\Omega_{\mathcal{O}}$ via $b \sim \mathsf{Ber}(\nu - \nu')$, so that there is an exact cutoff if one truncates at time t for $t > N$ and for $t = N$ if additionally $b = 1$.

Let $T_{\mathcal{O}} = \mathsf{time}_{\mathcal{O}}(\langle \mathsf{A}, \mathcal{O} \rangle)$. Then

$$T^{\nu}_{\mathcal{O}} = \mathsf{time}_{\mathcal{O}}(\langle \mathsf{A}, \mathcal{O}^{\nu} \rangle),$$

assuming the execution of $\langle \mathsf{A}, \mathcal{O}^{\nu} \rangle$ stops with `timeout` (and the purported runtime is $N = N(\nu)$.) In other words, truncating the runtime distributions and truncating the oracle have the "same" effect.

Our timeout oracle $\mathcal{O}'$ is defined as the ν-quantile truncated oracle, except that $\mathcal{O}'$ additionally pays a small constant time overhead for sending `timeout`. (Recall that due to consistency reasons, sending messages sets lower bounds for purported runtime for timeful oracles.) Note that $\mathbb{P}(\langle \mathsf{A}, \mathcal{O}^{\leq N} \rangle \text{ has } \texttt{timeout}) = \nu$ by construction. Moreover

$$\mathsf{time}_{\mathcal{O}}(\langle \mathsf{A}, \mathcal{O}' \rangle) \leq \mathsf{time}_{\mathcal{O}}(\langle \mathsf{A}, \mathcal{O}^{\nu} \rangle) + O(1),$$

hence the claims follow (as in Corollary 3.9). □

In our setting, we usually deal with "multi-oracle" adversaries. For example, zero-knowledge needs input generation $\mathcal{I}$ and a malicious verifier $\mathcal{V}^*$ (and a distinguisher $\mathcal{D}$ which of lesser concern). Clearly, we can view $\mathcal{I}$ and $\mathcal{V}^*$ as a single oracle (or party), by merging everything except the prover $\mathcal{P}$ into one entity. The new entity first runs $\mathcal{I}$ to produce inputs, and then continues as $\mathcal{V}^*$. For completeness, this is discussed more explicitly in Appendix E.2.2

4. Towards applications

In this section, we gather the basic tools to deal with a posteriori efficiency. While we focus on CEPT, it will again be evident that our techniques work for "algebra-tailed" runtime classes, and the results generalize to any such class (where the algebra for negligible functions coincides with the algebra for runtime), see Appendix D for the definitions.

[34] More formally, one should *lift* A to an algorithm which aborts upon receiving `timeout`, since `timeout` is a special symbol which A cannot receive/interpret.

[35] The constant $O(1)$ merely accounts for $\mathcal{O}'$ and outputting `timeout`.

[36] The probability space may be enlarged to achieve an exact cutoff, see Section 2.5.

4.1. Standard reductions and truncation techniques

In this section, we give some semi-abstract reduction and truncation techniques, which are the workhorse for dealing with designated CEPT adversaries.

Lemma 4.1 (Reduction to a priori runtime). *Let $\mathcal{O}_0$ and $\mathcal{O}_1$ be two oracles. Suppose $\mathcal{D}$ is a distinguisher with advantage $\varepsilon \coloneqq \mathrm{Adv}^{dist}_{\mathcal{D},\mathcal{O}_0,\mathcal{O}_1}$. Let $T_0 = \mathsf{time}_{\mathcal{D}}(\mathcal{D}^{\mathcal{O}_0})$ and let $\nu_0 \in [0,1]$ be some negligible function. Suppose there is a tail bound t_0 for T_0 with $\mathbb{P}(T_0 > t_0) \leq \frac{1}{4}\varepsilon + \nu$. Then there is a (standard) distinguisher $\mathcal{A}$ with runtime strictly bounded by $t = t_0$ (up to emulation overhead), and advantage at least $\frac{\varepsilon}{4} - \nu_0$ infinitely often. More concretely, $\mathcal{A}$ is a runtime truncation of $\mathcal{D}$ after t steps with tiny overhead.*

The only reason for stating Lemma 4.1 in the asymptotic setting is convenience. We also note that instead of runtime, any "runtime-like" statistic, e.g. the number of oracle queries, can be used.

Proof sketch. Let $\mathcal{A}$ run $b \leftarrow \mathcal{D}^{\leq t_0}$ and output b, except if $b = \texttt{timeout}$, where $\mathcal{A}$ returns a random bit instead. The outputs of $\mathcal{D}^{\mathcal{O}_0}$ and $\mathcal{A}^{\mathcal{O}_0}$ have statistical distance at most $\frac{1}{4}\varepsilon + \nu_0$ by assumption on the tail bound t_0.

Suppose the output of $\mathcal{A}^{\mathcal{O}_1}$ has statistical distance δ of $\mathcal{D}^{\mathcal{O}_1}$. If $\delta > \frac{2\varepsilon}{4}$, then necessarily, the probability that $\mathcal{A}^{\mathcal{O}_1}$ exceeds t_0 steps is greater than $\frac{2\varepsilon}{4}$. Thus, this runtime statistic can be used as a distinguishing property, with advantage at least $\frac{\varepsilon}{4} - \nu_0$ (infinitely often). (The distinguisher $\mathcal{A}'$ obtained from this returns 1 on `timeout` and 0 otherwise.)

Now suppose $\delta \leq \frac{2\varepsilon}{4}$. Then the advantage of $\mathcal{A}$ is at least $\frac{\varepsilon}{4} - \nu_0$ (by statistical distance of the outputs). The promised runtime bounds for $\mathcal{A}$ and $\mathcal{A}'$ follow immediately. □

Plugging in the tail bounds for CEPT, we get the following.

Corollary 4.2 (Standard reduction to PPT). *Let $\mathcal{O}_0$ and $\mathcal{O}_1$ be two oracles. Suppose $\mathcal{D}$ a distinguisher with advantage $\mathrm{Adv}^{dist}_{\mathcal{D},\mathcal{O}_0,\mathcal{O}_1}$ at least $\varepsilon \coloneqq \frac{1}{\mathsf{poly}}$ infinitely often, and $\mathsf{time}_{\mathcal{D}}(\mathcal{D}^{\mathcal{O}_0}) \in \mathcal{CEPT}$. Then there is an* a priori PPT *(standard) distinguisher $\mathcal{A}$ with advantage at least $\frac{\varepsilon}{4} - \mathsf{negl}$ infinitely often.*

Note that $\mathcal{D}$ need only be efficient for $\mathcal{O}_0$, and that the constructed $\mathcal{A}$ has roughly the same runtime distribution as $\mathcal{D}$.

Remark 4.3 (Standard cutoff argument). The strategy in the proof of Lemma 4.1 and Corollary 4.2 is the *standard cutoff argument.* It works with minor variations in many situations.

Notation 4.4. We often sloppily write $\overset{c}{\approx}$ instead of $\overset{c}{\approx}_{\mathcal{T}}$ when specifying indistinguishability. Corollary 4.2 justifies this (for the runtime classes of interest).

4.2. Relative efficiency

By considering a posteriori runtime and designated adversaries, we lack a notion of "absolute" efficiency of an algorithm (or timeful system). Instead, we rely on a relative notion of efficiency, which is a definitional cornerstone in our setting.

Definition 4.5 (Weak relative efficiency). Let A and B be two (interactive) algorithms (or timeful systems) with identical interfaces. We say that B is **weakly $(\mathcal{T},\mathcal{S})$-efficient relative to** A *w.r.t. (implicit) runtime classes $\mathcal{T}$, $\mathcal{S}$*, if for all distinguishing environments $\mathcal{E}$ (which yield closed systems $\langle \mathcal{E}, \mathsf{A}\rangle$, $\langle \mathcal{E}, \mathsf{B}\rangle$)

$$\mathsf{time}_{\mathcal{E}+\mathsf{A}}(\langle \mathcal{E}, \mathsf{A}\rangle) \in \mathcal{T} \implies \mathsf{time}_{\mathcal{E}+\mathsf{A}}(\langle \mathcal{E}, \mathsf{B}\rangle) \in \mathcal{S}$$

We say B is **weakly efficient relative to** A *w.r.t. an (implicit) runtime class $\mathcal{T}$*, if it is weakly $(\mathcal{T},\mathcal{T})$-efficient relative to A.

Efficiency relative to a "base" algorithm is the notion of efficiency we need in security definitions and reductions. Indeed, if an adversary is not efficient in the real protocol, the simulator (or reduction) need not be efficient either. However, whenever the adversary is efficient, so should the simulation (or reduction) be.[37] A stronger, unconditional form of relative efficiency is the following. In the following, the runtime classes $\mathcal{T}$ and $\mathcal{S}$ are from $\{\mathcal{PPT}, \mathcal{EPT}, \mathcal{CPPT}, \mathcal{CEPT}\}$, and they decide whether strict or expected time is measured. This allows us to specify cases $\mathcal{T} = \mathcal{S} = \mathcal{PPT}$, $(\mathcal{T}, \mathcal{S}) = (\mathcal{PPT}, \mathcal{EPT})$, and $\mathcal{T} = \mathcal{S} = \mathcal{EPT}$, as well as variations with virtuality succinctly.

Definition 4.6 (Tight relative efficiency). Let A, B be as in Definition 4.5. We say that B is **$(\mathcal{T}, \mathcal{S})$-efficient relative** to A with **runtime tightness** $(\mathsf{poly}_{\mathsf{time}}, \mathsf{poly}_{\mathsf{virt}})$, if: For all *timeful* environments $\mathcal{E}$, if $\mathsf{time}_{\mathsf{A}}(\langle \mathcal{E}, \mathsf{A} \rangle)$ is virtually strict/expected (t_0, ε_0)-time, then $\mathsf{time}_{\mathsf{B}}(\langle \mathcal{E}, \mathsf{B} \rangle)$ is virtually strict/expected (t_1, ε_1)-time, with $t_1(\kappa) \leq \mathsf{poly}_{\mathsf{time}}(\kappa) t_0(\kappa)$ with $\varepsilon_1(\kappa) \leq \mathsf{poly}_{\mathsf{virt}}(\kappa)\varepsilon_0(\kappa)$ (for all κ).

4.3. Hybrid lemma

The formulation and proof of the hybrid lemma is more involved than for a priori definitions of runtime. To state it, we require relative efficiency. To prove it, we use the random embedding trick from [HUM13] to allow us to get a measure of runtime, which is closely related to the runtime of the full hybrid, even though the time spent in the challenge oracle is inaccessible to a reduction.

Lemma 4.7 (Hybrid-Lemma for CEPT). *Suppose that $\mathcal{O}_1$ is weakly efficient relative to $\mathcal{O}_0$ and that $\mathcal{O}_0 \overset{c}{\approx} \mathcal{O}_1$. Suppose that $\mathcal{D}$ is an algorithm with oracle-access to $\mathsf{rep}(\mathcal{O}_b)$, and $\mathsf{time}_{\mathcal{D}+\mathsf{rep}(\mathcal{O}_0)}(\mathcal{D}^{\mathsf{rep}(\mathcal{O}_0)}) \in \mathcal{CEPT}$. Then $\mathsf{time}_{\mathcal{D}+\mathsf{rep}(\mathcal{O}_1)}(\mathcal{D}^{\mathsf{rep}(\mathcal{O}_1)}) \in \mathcal{CEPT}$ and the distinguishing advantage is*

$$\mathrm{Adv}^{dist}_{\mathcal{D},\mathcal{O}_0,\mathcal{O}_1} = |\mathbb{P}(\mathcal{D}^{\mathsf{rep}(\mathcal{O}_0)} = 1) - \mathbb{P}(\mathcal{D}^{\mathsf{rep}(\mathcal{O}_1)} = 1)| \leq \mathsf{negl}.$$

In other words, $\mathsf{rep}(\mathcal{O}_1)$ is weakly efficient relative to $\mathsf{rep}(\mathcal{O}_0)$, and $\mathsf{rep}(\mathcal{O}_0) \overset{c}{\approx} \mathsf{rep}(\mathcal{O}_1)$.

For a detailed sketch of the proof and the intuition, we refer back to Section 1.4.4.

Proof. We split the proof into several steps. We first show, that proving the claim for q-fold repeated access, for arbitrary but fixed polynomial $q(\kappa)$, is enough.

Claim 4.8. *Suppose the hybrid lemma holds for $\mathsf{rep}_q(\mathcal{O}_0)$ and $\mathsf{rep}_q(\mathcal{O}_1)$ for any polynomial q. That is, for any distinguisher $\mathcal{D}$ with $\mathsf{time}_{\mathcal{D}+\mathsf{rep}_q(\mathcal{O}_0)}(\mathcal{D}^{\mathsf{rep}_q(\mathcal{O}_0)}) \in \mathcal{CEPT}$, we have that $\mathsf{time}_{\mathcal{D}+\mathsf{rep}_q(\mathcal{O}_1)}(\mathcal{D}^{\mathsf{rep}_q(\mathcal{O}_1)}) \in \mathcal{CEPT}$ and advantage $\mathrm{Adv}^{dist}_{\mathcal{D},\mathcal{O}_0,\mathcal{O}_1} = |\mathbb{P}(\mathcal{D}^{\mathsf{rep}_q(\mathcal{O}_0)} = 1) - \mathbb{P}(\mathcal{D}^{\mathsf{rep}_q(\mathcal{O}_1)} = 1)| \leq \mathsf{negl}$. Then the hybrid lemma holds.*

Proof of Claim 4.8. Suppose $\mathcal{D}$ is a distinguisher with $\mathsf{time}_{\mathcal{D}+\mathsf{rep}(\mathcal{O}_0)}(\mathcal{D}^{\mathsf{rep}(\mathcal{O}_0)}) \in \mathcal{CEPT}$ and non-negligible advantage. Let the advantage exceed $\varepsilon = 1/\mathsf{poly}$ infinitely often. The number of Q_0 of $\mathcal{O}_0$-instances generated in $\mathcal{D}^{\mathsf{rep}(\mathcal{O}_0)}$ is certainly CEPT. Let Q_1 denote the number of $\mathcal{O}_1$-instances generated in $\mathcal{D}^{\mathsf{rep}(\mathcal{O}_1)}$. Treating these statistics as "runtime", the standard truncation argument (Corollary 4.2), ensures that there is a PPT distinguisher $\mathcal{A}$ which makes a strictly polynomial number q of queries and has advantage at least $\varepsilon/4 - \mathsf{negl}$ infinitely often. Clearly, $\mathcal{A}^{\mathsf{rep}(\mathcal{O}_0)}$ has runtime bounded by $\mathcal{D}^{\mathsf{rep}(\mathcal{O}_0)}$ (up to emulation overhead), hence remains efficient. Consequently, we have reduced to the setting where at most q queries are made for some polynomial q which depends on $\mathcal{D}$. □

From now on, we assume that $\mathcal{D}$ generates at most q oracle instances, for some polynomial q. We proceed in game hops, starting with $\mathcal{D}^{\mathsf{rep}(\mathcal{O}_0)}$ and finishing with $\mathcal{D}^{\mathsf{rep}(\mathcal{O}_1)}$ For each hop, we have to ensure indistinguishable output bits and preservation of efficiency.

[37] Strictly speaking, a simulator depends on the adversary, and Definition 4.5 should be applied "pointwise", i.e. for every adversary, the simulator should be efficient relative to the simulator. For completeness, we give a generalized definition in Appendix E.3.3.

- **Game** G_0 is simply the execution of $\mathscr{D}^{\mathsf{rep}(\mathcal{O}_0)}$.
- In **Game** G_1 we pick a random permutation $\pi\colon \{1,\dots,q\} \to \{1,\dots,q\}$ and reroute the access to the oracles: If $\mathscr{D}$ queries for the i-th oracle, it is routed to the $\pi(i)$-th oracle. More precisely, in G_0 the adversary $\mathscr{D}$ has access to $\vec{\mathcal{O}} = (\mathcal{O}_0^1,\dots,\mathcal{O}_0^q)$, whereas in G_1 it has access to a random permutation $\vec{\mathcal{O}}^\pi = (\mathcal{O}_0^{\pi(1)},\dots,\mathcal{O}_0^{\pi(q)})$. Clearly, G_0 and G_1 are perfectly indistinguishable and have almost identical runtime (up to bookkeeping). The key change is, that all $\mathcal{O}^i$ now have identical runtime distributions.
- In **Game** G_2, we replace the $\mathcal{O}_0^1$ with $\mathcal{O}_1^1$, that is we consider $\vec{\mathcal{O}} = (\mathcal{O}_1^1, \mathcal{O}_0^2,\dots,\mathcal{O}_0^q)$. Indistinguishability (of outputs) of G_2 follows directly from the standard reduction, whereas efficiency of G_2 follows from $\mathcal{O}_1$ being weakly efficient relative to $\mathcal{O}_0$. Note that the runtime of G_2 may differ significantly from that of G_1.
- In **Game** G_3, we have $\vec{\mathcal{O}} = (\mathcal{O}_1^1,\dots,\mathcal{O}_1^{q-1},\mathcal{O}_0^q)$. That is, all but one oracle instance is of $\mathcal{O}_1$-type. Proving that G_3 is CEPT is the key point in this argument. Indistinguishability of G_2 and G_3 follows easily. We postpone the proof to Claim 4.9, and finish up first.
- In **Game** G_4, we use $\vec{\mathcal{O}} = (\mathcal{O}_1^1,\dots,\mathcal{O}_1^{q-1},\mathcal{O}_1^q)$, that is, we switched completely to $\mathcal{O}_1$ for every instance. Efficiency and output indistinguishability follow as from G_1 to G_2.
- In **Game** G_5, we remove the random permutation π. Thus, G_5 is $\mathscr{D}^{\mathsf{rep}(\mathcal{O}_1)}$, as claimed. □

Claim 4.9. *If* G_2 *is CEPT, so is* G_3. *Moreover, their outputs are indistinguishable.*

We will prove Claim 4.9 by establishing a relatively precise grasp on the runtime.

Proof. Recall that we have to switch the oracle setup from $(\mathcal{O}_1^1,\mathcal{O}_0^2,\dots,\mathcal{O}_0^q)$ to $(\mathcal{O}_1^1,\dots,\mathcal{O}_1^{q-1},\mathcal{O}_0^q)$. The core difficulty is the efficiency in the latter case. Following the trick of [HUM13], we randomized the oracle order for $\mathscr{D}$ using a random permutation π. This spreads the runtime of $\mathcal{O}_1^1$ and $\mathcal{O}_0^q$ evenly over all possible positions, and this property is at the heart of the reduction.

Let H_ℓ denote the game with oracle setup $\vec{\mathcal{O}} = (\mathcal{O}_1^1,\dots,\mathcal{O}_1^\ell,\mathcal{O}_0^{\ell+1},\dots,\mathcal{O}_0^q)$.[38] By construction, H_1 equals G_2 and H_{q-1} equals G_3. Thus, it suffices to prove indistinguishability of H_1 and H_{q-1}.

Clearly, hybrids H_ℓ and $\mathsf{H}_{\ell+1}$ (for $\ell = 1,\dots,q-2$) are related by a direct reduction to $\mathcal{O}_0 \overset{c}{\approx} \mathcal{O}_1$. The hybrid reduction R embeds the challenge oracle $\mathcal{O}_b^*$ in position $\ell+1$, picking $\ell \leftarrow \{1,\dots,q-2\}$ uniformly. Denote by R_ℓ the reduction with fixed choice ℓ. By construction, $R_\ell^{\mathcal{O}_b^*} = \mathsf{H}_{\ell+b}$. Note that $\mathscr{D}$ has randomly permuted access, so the challenge oracle is embedded uniformly from $\mathscr{D}$'s view.

One main complication is, that $R_\ell^{\mathcal{O}^*}$ cannot keep runtime statistics of $\mathcal{O}^*$. Yet, we need enough control over the runtime to guarantee efficiency of R and H_{q-1}. By randomizing the order of the oracle instances (from the view of $\mathscr{D}$), we can exploit the strong symmetry of the local runtimes of instances. Let us now take a close look at these runtimes. To keep notation in check, we fix a hybrid H_ℓ, and notationally suppress the dependency of most variables on ℓ.

- Let $T_\ell = \mathsf{time}(\mathsf{H}_\ell)$ be the total runtime of H_ℓ (as a random variable). Recall that it is understood that H_ℓ emulates all oracles $\mathcal{O}^1,\dots,\mathcal{O}^q$.
- Let $T^{\mathcal{O},j}$ denote the time H_ℓ spends in $\mathcal{O}^j$.
- Let $T^{\mathscr{D}}$ denote the time H_ℓ spends outside $\mathcal{O}^j$ (mostly emulating $\mathscr{D}$).
- We have $T_\ell = T^{\mathscr{D}} + \sum_{j=1}^q T^{\mathcal{O},j}$ as random variables by definition.

By the symmetry introduced by the random permutation π, the distributions of the $T^{\mathcal{O},j}$ for the same type of oracle coincide. That is, $T^{\mathcal{O},i} \overset{d}{\equiv} T^{\mathcal{O},j}$ for all $(i,j) \in \{1,\dots,\ell\}$ for $\mathcal{O}_1$-type instances, and likewise with $(i,j) \in \{\ell+1,\dots,q\}$ for all $\mathcal{O}_0$-type instances.

Claim 4.10. *Let* $S_\ell = T^{\mathscr{D}} + T^{\mathcal{O},1} + T^{\mathcal{O},q}$. *Then we have*

$$\overline{\mathsf{CDF}}_{T_\ell}(\,\cdot\,) \le (q+1)\overline{\mathsf{CDF}}_{(q+1)S_\ell}(\,\cdot\,) \quad \textit{that is} \quad T_\ell \overset{d}{\le}_{(q+1)} (q+1)\cdot S_\ell. \tag{4.1}$$

[38] Note that all hybrids have $\mathcal{O}_1^1$ and $\mathcal{O}_0^q$ fixed. Thus there $q-1$ hybrids and $q-2$ hybrid transitions.

Proof. Using the definition and symmetries, we argue that

$$\begin{aligned}
\overline{\mathsf{CDF}}_{T_\ell}(t) &= \mathbb{P}(T^{\mathscr{D}} + \sum_{i=1}^{\ell} T^{\mathcal{O},i} + \sum_{j=\ell+1}^{q} T^{\mathcal{O},j} > t) \\
&\leq \mathbb{P}(T^{\mathscr{D}} > \frac{1}{q+1}t) + \mathbb{P}(\sum_{i=1}^{\ell} T^{\mathcal{O},i} > \frac{\ell}{q+1}t) + \mathbb{P}(\sum_{j=\ell+1}^{q} T^{\mathcal{O},j} > \frac{q-\ell}{q+1}t) \\
&\leq \mathbb{P}(T^{\mathscr{D}} > \frac{1}{q+1}t) + \sum_{i=1}^{\ell} \mathbb{P}(T^{\mathcal{O},i} > \frac{1}{q+1}t) + \sum_{j=\ell+1}^{q} \mathbb{P}(T^{\mathcal{O},j} > \frac{1}{q+1}t) \\
&= \mathbb{P}((q+1)\cdot T^{\mathscr{D}} > t) + \ell \cdot \mathbb{P}((q+1)\cdot T^{\mathcal{O},1} > t) + (q-\ell)\cdot \mathbb{P}((q+1)\cdot T^{\mathcal{O},q} > t) \\
&\leq (q+1)\cdot \mathbb{P}((q+1)\cdot S_\ell > t).
\end{aligned}$$

The first two inequalities use that for any sum $\sum_{i=1}^{n} \lambda_i X_i > x$ with $\lambda_i \geq 0$ and $\sum_{i=1}^{n} \lambda_i = 1$, there exists some i such that $X_i > \lambda_i x$. The next (in)equalities follow from symmetries and simplifications. The final inequality holds, because by construction, S_ℓ dominates $T^{\mathscr{D}}$, $T^{\mathcal{O},1}$ and $T^{\mathcal{O},q}$. Also, we bound $1, \ell$ and $q-\ell$ by $q+1$. Thus the claim follows. □

Note that S_ℓ can be computed, even in a reduction between hybrids, since $\mathcal{O}^1$ and $\mathcal{O}^q$ are fixed and there is never a challenge-embedding there. This is crucial. Using Eq. (4.1), the generalized CEPT characterization, Corollary 3.10, is applicable. Therefore, it suffices to prove that $S_1 \overset{c}{\approx} S_{q-1}$, and Corollary 3.10 ensures that T_{q-1} is CEPT if T_1 is CEPT This is our next step. (In a sense, we have now reduced efficiency to indistinguishability.)

Claim 4.11. *For S_ℓ defined as above, we have $S_1 \overset{c}{\approx} S_{q-1}$.*

To prove $S_1 \overset{c}{\approx} S_{q-1}$, we modify the hybrids H_ℓ to also output this quantity. That is, the hybrid H_ℓ outputs the runtime S_ℓ and output of $\mathscr{D}$, obtained by emulating $\mathscr{D}$ given access to $\vec{\mathcal{O}}^\pi$ with $\vec{\mathcal{O}} = (\mathcal{O}_1^1, \ldots, \mathcal{O}_1^\ell, \mathcal{O}_0^{\ell+1}, \ldots, \mathcal{O}_0^q)$ for $\ell = 1, \ldots, q-1$. For now, we focus solely on the output S_ℓ.

Recall that R denotes the hybrid reduction, which embeds its challenge-oracle $\mathcal{O}^*$ into $\mathcal{O}^{i^*}$ for $i^* = i+1$, where $i \leftarrow \{1, \ldots q-2\}$), and simulates the remaining oracles. Recall, that R_i denotes reduction R with fixed choice i. Hence, $R_\ell^{\mathcal{O}_b^*}$ simulates $\mathsf{H}_{\ell+b}$.

If $R^{\mathcal{O}_0}$ were efficient, this would almost finish the proof. However, we do not yet know whether $R^{\mathcal{O}_0}$ is efficient. Since we only need to prove S_1 and S_{q-1} indistinguishable, we truncate the hybrids and the reduction. We then need to prove that the truncations are close to the originals, i.e. the probability for `timeout` is (arbitrarily polynomially) small. We define:

- $[\mathsf{H}_\ell]$ is the hybrid which imposes a strict time bound of $t_{\max}$ (to be chosen later) on each oracle emulation (i.e. each $\mathcal{O}^i$) individually as well as the emulation of $\mathscr{D}$. If a bound is exceed, $[\mathsf{H}_\ell]$ aborts with `timeout`. (That is, $[\mathsf{H}_\ell]$ aborts if $T^{\mathscr{D}} > t_{\max}$ or $T^{\mathcal{O},i} > t_{\max}$ for any $i = 1, \ldots, q$.)
- $[R]_\ell$ is defined analogously to $[\mathsf{H}_\ell]$. Note that $[R]$ cannot truncate its challenge oracle $\mathcal{O}^*$.
- $[\mathcal{O}]$ (resp. $[\mathscr{D}]$) denotes same cutoff at $t_{\max}$ applied to an oracle (resp. $\mathscr{D}$).

By definition

$$[\mathsf{H}_\ell] = [R]_{\ell-1}^{[\mathcal{O}_1]} = [R]_\ell^{[\mathcal{O}_0]}$$

if the expressions are defined. The technical problem, is that we can only compute $[R]_\ell^{\mathcal{O}^*}$, but not $[R]_\ell^{[\mathcal{O}^*]}$. Yet, the latter is necessary in the usual telescoping sum. To quantify the introduced error, we define:

- $h_\ell \coloneqq \mathbb{P}([\mathsf{H}_\ell] = \texttt{timeout})$, the `timeout` probability of $[\mathsf{H}_\ell]$.

- $r_\ell^b := \mathbb{P}([R]_\ell^{\mathcal{O}_b} = \texttt{timeout})$, the `timeout` probability of $[R]_\ell^{\mathcal{O}_b}$. Note that the challenge oracle $\mathcal{O}_b$ cannot time out.

Claim 4.12. *For all $\delta = 1/\mathsf{poly}$, there exists a polynomial $t_{\max}$ such that for all $\ell = 1, \dots q-1$*

$$h_\ell \le \delta \quad \textit{and} \quad |h_\ell - r_{\ell-1}^1| \le \delta \quad \textit{and} \quad |h_\ell - r_\ell^0| \le \delta.$$

Similar to before, and as in [HUM13], it is easy to use the symmetry of `timeouts` to show[39]

$$r_{\ell-1}^1 \le h_\ell \le \frac{\ell}{\ell-1} \cdot r_{\ell-1}^1 \tag{4.2}$$

$$r_\ell^0 \le h_\ell \le \frac{q-\ell}{q-\ell-1} \cdot r_\ell^0 \tag{4.3}$$

for all $\ell = 1, \dots, q-2$. This implies

$$h_\ell \le \frac{\ell}{\ell-1} r_{\ell-1}^1 = \frac{\ell}{\ell-1} r_{\ell-1}^0 + \frac{\ell}{\ell-1}(r_{\ell-1}^1 - r_{\ell-1}^0) \le \frac{\ell}{\ell-1} h_{\ell-1} + \frac{\ell}{\ell-1}\rho_{\ell-1}$$

where we let $\rho_i = r_i^1 - r_i^0$. Inductively we find[40]

$$h_\ell \le \ell \cdot h_1 + \sum_{i=1}^{\ell-1} \frac{\ell}{i} \rho_i \tag{4.4}$$

Recall that we can make h_1 arbitrarily polynomially small by picking $t_{\max}$ large enough. Now, we want to prove that h_ℓ is also small for all ℓ. Hence we are looking for a (small) upper bound on $\sum_{i=1}^{\ell-1} \frac{\ell}{i}\rho_i$.

If all ρ_i were positive, we could just guess a good i and use a completely standard hybrid argument, but we do not know this. In [HUM13], non-uniform advice is used, namely the index i^* which maximizes $|\rho_{i^*}|$. It is easy to see that $[R]_{i^*}$ is a distinguisher with advantage $|\rho_{i^*}|$, hence $|\rho_{i^*}|$ is negligible and consequently $|\sum_{i=1}^{\ell-1} \frac{\ell}{i}\rho_i| \le q \cdot |\rho_{i^*}|$ is also negligible. It is also noted in [HUM13], that one can approximate ℓ^* using $[R]^{\mathcal{O}_b}$ to obtain a uniform distinguisher. In [HUM13], $[R]^{\mathcal{O}_0}$ is efficient by assumption, which makes approximation of i^* straightforward. In our setting, efficiency of $[R]^{\mathcal{O}_0}$ does not hold by assumption. Thus, we need to argue differently.

We approximate a good index i^*, where ρ_{i^*} is large by using the following inequality

$$\rho_{\ell-1}^- := \frac{\ell-1}{\ell} h_\ell - h_{\ell-1} \le \rho_{\ell-1} = r_{\ell-1}^1 - r_\ell^0 \tag{4.5}$$

where we used Eqs. (4.2) and (4.3). Observe that $\mathsf{Ber}(h_i)$ can be sampled in $\mathsf{time}([\mathsf{H}_i]) \le (q+1) \cdot t_{\max}$ (up to emulation overhead), since $h_i = \mathbb{P}([\mathsf{H}_i] = \texttt{timeout})$. Hence, we can approximate h_i (and hence ρ_i^-) via sampling to arbitrary polynomial precision. By induction, we find

$$h_\ell = \ell \cdot h_1 + \sum_{i=1}^{\ell-1} \frac{\ell}{i} \rho_i^- \tag{4.6}$$

which is an equality by definition. Note that if $\max_{i=1}^{q-2} \rho_i^- \le \nu$ for some negligible ν, we get $h_\ell \le \ell h_1 + \ell^2 \nu$. This is sufficient for our purposes. We stress that we do not consider absolute values here, as we only need an upper bound for the timeout probability.[41]

[39] This is the reason we applied `timeouts` to each oracle individually, instead of to the whole game H_ℓ. The latter may not exhibit this symmetry.

[40] In a standard hybrid argument, we would have $\sum_i \rho_i = r_1^0 - r_{\ell-1}^1$ instead of a weighted sum.

[41] One can similarly define $\rho_\ell^+ := h_{\ell+1} - \frac{q-\ell-1}{q-\ell} h_\ell \ge \rho_\ell$ and consider $\min_{i=2}^{q-1} \rho_i^-$. Hence there is a negl such that $|\rho_\ell| \le \mathsf{negl}$ for all ℓ. We do not need this.

We argue by contradiction. Suppose $\max_{i=1}^{q-2} \rho_i^- > 1/\mathsf{poly}$ infinitely often. Then, as noted, we can approximate ρ_i^- up to any polynomial precision, and in particular we can sample a "good" i^* which satisfies $\rho_{i^*} \geq \rho_{i^*}^- \geq \frac{1}{2\mathsf{poly}(\kappa)}$ with overwhelming probability.[42] Thus, this yields a distinguisher for $\mathcal{O}_0$ and $\mathcal{O}_1$, which is efficient for any choice of polynomial cutoff bound $t_{\max}$ and has advantage $\geq \frac{1}{2\mathsf{poly}} - \mathsf{negl}$ infinitely often. (Namely, first approximate i^* and then run $[R]_{i^*}^{\mathcal{O}^*}$. The choice of i^* is "good" with overwhelming probability, so we get an advantage of at least $\frac{1}{2\mathsf{poly}(\kappa)} - \mathsf{negl}$ infinitely often.) This finally proves that

$$\forall t_{\max} = \mathsf{poly}\ \exists \nu = \mathsf{negl}\ \forall \ell = 2, \ldots, q-1 \colon \quad h_\ell \leq \ell h_1 + \ell^2 \nu.$$

In particular, for almost all κ, $h_\ell \leq q h_1 + q^2 \nu$. Since H_1 is CEPT, for any poly there is a $t_{\max}$ such that $h_1 \leq 1/\mathsf{poly}$. Let $\delta = 1/\mathsf{poly}$ for some poly. Pick $t_{\max}$ such that $h_1 \leq \frac{\delta}{2q}$. Then $h_\ell \leq \delta/2 + q^2\nu \leq \delta$ for almost all κ. This proves the first part of Claim 4.12. From Eqs. (4.2) and (4.3) we also find

$$|h_\ell - r_\ell^0| \leq \frac{1}{q-\ell-1} h_\ell \quad \text{and} \quad |h_\ell - r_{\ell-1}^1| \leq \frac{1}{\ell-1} h_\ell.$$

This completes the proof of Claim 4.12. Claim 4.11 now follows from Lemma 4.13 below. More concretely, it follows that $\mathsf{H}_1 \overset{c}{\approx} \mathsf{H}_{q-1}$, which implies $S_1 \overset{c}{\approx} S_{q-1}$ (since we augmented the output of H_ℓ by S_ℓ), and, by Corollary 3.10, H_{q-1} is CEPT (since H_1 is CEPT). This finishes the proof of Claim 4.9. □

The following lemma uses notation similar to the above, but does not follow the indexing. This simplifies the presentation, as we can go from 0 to q instead of 1 to $q-1$.

Lemma 4.13 (Approximable hybrid lemma). *Let $\mathcal{O}_0, \mathcal{O}_1$ be oracles, let $\mathsf{H}_0, \ldots, \mathsf{H}_q$ be hybrid games (for polynomial q) and let R be an algorithm. We call R a* hybrid reduction *for H, if it is of the following form:*

- *R is an oracle algorithm which in the beginning chooses a random integer $i^* \in \{0, \ldots, q-1\}$.*
- *Denoting by R_i the algorithm with fixed choice $i^* = i$, we have $R_i^{\mathcal{O}_b} \equiv \mathsf{H}_{i+b}$, where we mean equivalence as systems.[43]*

We say R is a time-approximable hybrid reduction*, if for any choice $\delta = \frac{1}{\mathsf{poly}_\delta}$:*

(1) There exist "truncated" a priori PPT hybrids $[\mathsf{H}_\ell]$ for $i = 0, \ldots, q$, which may return `timeout`*.*
(2) H_ℓ and $[\mathsf{H}_\ell]$ are equal until `timeout` *and $\mathbb{P}([\mathsf{H}_\ell] = \texttt{timeout}) \leq \delta$.*
(3) There exists a "truncated" a priori PPT reduction algorithm $[R]$ with $\Delta([\mathsf{H}_\ell], [R]_\ell^{\mathcal{O}_0}) \leq \delta$ and $\Delta([\mathsf{H}_{\ell+1}], [R]_\ell^{\mathcal{O}_1}) \leq \delta$ (for almost all κ).

If R is a time-approximable hybrid reduction for H, then $R_0^{\mathcal{O}_0} \equiv \mathsf{H}_1 \overset{c}{\approx} \mathsf{H}_q \equiv R_{q-1}^{\mathcal{O}_1}$. More precisely, for every a priori PPT distinguisher $\mathcal{D}$ and for every $\delta = \frac{1}{\mathsf{poly}_\delta}$ there exists an adversary $\mathcal{A}$ such that $\mathrm{Adv}^{dist}_{\mathsf{H}_0,\mathsf{H}_q,\mathcal{D}} \leq q \cdot \mathrm{Adv}^{dist}_{\mathcal{O}_0,\mathcal{O}_1,\mathcal{A}} + (2q+2)\delta$ (for almost all κ).
One can replace item (3) with

(4) For any $\delta = \frac{1}{\mathsf{poly}_\delta}$ there exists a "truncated" a priori PPT reduction algorithm $[R]$, such that $\Delta(R_\ell^{\mathcal{O}_0}, [R]_\ell^{\mathcal{O}_0}) \leq \delta$ for all ℓ.

In this case, we get $\mathrm{Adv}^{dist}_{\mathsf{H}_0,\mathsf{H}_q,\mathcal{D}} \leq q \cdot \mathrm{Adv}^{dist}_{\mathcal{O}_0,\mathcal{O}_1,\mathcal{A}} + (4q+4)\delta$ (for almost all κ).

[42] We can use the same approximation of distributions as for CEPT, see also Appendix C.4.
[43] We assume that H_ℓ is a closed system. But this actually unnecessary, if we allow distinguishing environments.

Note that Claim 4.11 establishes items (1), (2) and (4) in the respective setting. Hence, Lemma 4.13 is applicable.

One can easily generalize Lemma 4.13 beyond runtime truncation. Truncation and `timeout` ("time-approximation") is just the special case of approximation we are most interested in.

Proof. We can assume w.l.o.g. that H_ℓ outputs a bit, since we can integrate a distinguisher (which is w.l.o.g. a priori PPT) into H. Thus, the distinguisher advantage is now simply $\Delta(\mathsf{H}_0, \mathsf{H}_q)$. Consider some $\delta = \frac{1}{\mathsf{poly}_\delta}$ and let $[\mathsf{H}_\ell]$ and $[R]_\ell$ as in the statement. By the triangle inequality and item (2), we get

$$\Delta(\mathsf{H}_0, \mathsf{H}_q) \leq \Delta([\mathsf{H}_0], [\mathsf{H}_q]) + 2\delta.$$

Moreover, we have for almost all κ

$$\begin{aligned}
\Delta([\mathsf{H}_0], [\mathsf{H}_q]) &= \Big|\sum_{\ell=0}^{q-1} \mathbb{P}([\mathsf{H}_\ell] = 1) - \mathbb{P}([\mathsf{H}_{\ell+1}] = 1)\Big| \\
&= \Big| \sum_{\ell=0}^{q-1} \mathbb{P}([\mathsf{H}_\ell] = 1) - \mathbb{P}([R]_\ell^{\mathcal{O}_0} = 1) + \mathbb{P}([R]_\ell^{\mathcal{O}_1} = 1) \\
&\quad - \mathbb{P}([\mathsf{H}_{\ell+1}] = 1) + \mathbb{P}([R]_\ell^{\mathcal{O}_0} = 1) - \mathbb{P}([R]_\ell^{\mathcal{O}_1} = 1)\Big| \\
&\leq \sum_{\ell=0}^{q-1} |\mathbb{P}([\mathsf{H}_\ell] = 1) - \mathbb{P}([R]_\ell^{\mathcal{O}_0} = 1)| \\
&\quad + \sum_{\ell=0}^{q-1} |\mathbb{P}([R]_\ell^{\mathcal{O}_1} = 1) - \mathbb{P}([\mathsf{H}_{\ell+1}] = 1)| \\
&\quad + |\sum_{\ell=0}^{q-1} \mathbb{P}([R]_\ell^{\mathcal{O}_0} = 1) - \mathbb{P}([R]_\ell^{\mathcal{O}_1} = 1)| \\
&\leq q\delta + q\delta + q \cdot \mathrm{Adv}^{\mathrm{dist}}_{\mathcal{O}_0, \mathcal{O}_1, \mathcal{A}}(\kappa)
\end{aligned}$$

where the inequality follows from item (3) and the "hybrid reduction" adversary $\mathcal{A}$ which runs $d \leftarrow [R]^{\mathcal{O}^*}$ and if $d \neq \texttt{timeout}$ outputs d, else 1. Taken together, we find for almost all κ

$$\Delta(\mathsf{H}_0, \mathsf{H}_q) \leq q \cdot \mathrm{Adv}^{\mathrm{dist}}_{\mathcal{O}_0, \mathcal{O}_1, \mathcal{A}}(\kappa) + (2q+2)\delta.$$

Since $\mathcal{O}_0 \overset{c}{\approx} \mathcal{O}_1$, $\mathrm{Adv}^{\mathrm{dist}}_{\mathcal{O}_0, \mathcal{O}_1, \mathcal{A}}(\kappa)$ is negligible. Now, let $\varepsilon = 1/\mathsf{poly}$ some prescribed polynomial bound. Since q is polynomial and δ is chosen after q, one can choose $\delta^{-1} \geq (4q+4)\varepsilon^{-1}$, which is still polynomial and ensures that $(2q+2)\delta \leq \frac{1}{2}\varepsilon$. Hence, $\Delta(\mathsf{H}_0, \mathsf{H}_q) = \frac{1}{2}\varepsilon + q \cdot \mathsf{negl} < \varepsilon$ (for almost all κ). Thus, $\Delta(\mathsf{H}_0, \mathsf{H}_q)$ is smaller than any polynomial ε. Consequently, $\Delta(\mathsf{H}_0, \mathsf{H}_q)$ is negligible, and the first part of the claim follows.

For the second part, note that from items (2) and (4) we find

$$\Delta([\mathsf{H}_{\ell+b}], [R]_\ell^{\mathcal{O}_b}) \leq \Delta([\mathsf{H}_{\ell+b}], \mathsf{H}_\ell) + \Delta(\mathsf{H}_{\ell+b}, R_\ell^{\mathcal{O}_b}) + \Delta(R_\ell^{\mathcal{O}_b}, [R]_\ell^{\mathcal{O}_b}).$$

By assumption $\Delta(\mathsf{H}_{\ell+b}, R_\ell^{\mathcal{O}_b}) = 0$. Thus, if $\Delta([\mathsf{H}_\ell], \mathsf{H}_\ell) \leq \delta/2$ and $\Delta(R_\ell^{\mathcal{O}_b}, [R]_\ell^{\mathcal{O}_b}) \leq \delta/2$ for all ℓ and $b \in \{0,1\}$, then $\Delta([\mathsf{H}_\ell], [R]_\ell^{\mathcal{O}_b}) \leq \delta$ and hence item (3) is satisfied. The claim now follows by using $\delta' = \delta/2$ as choice of δ for $[\mathsf{H}_\ell]$ and $[R]_\ell$ in items (2) and (4). □

5. Application to zero-knowledge arguments

Our flavour of zero-knowledge follows Goldreich's treatment of uniform complexity [Gol93], combined with Feige's designated adversaries [Fei90]. We only define efficient proof systems for NP-languages.

Definition 5.1 (Interactive arguments). Let $\mathcal{R}$ be an NP-relation with corresponding language $\mathcal{L}$. An **argument (system) for** $\mathcal{L}$ consists of two interactive algorithms $(\mathcal{P}, \mathcal{V})$ such that:

Efficiency: There is a polynomial poly so that for all (κ, x, w) the runtime $\mathsf{time}_{\mathcal{P}+\mathcal{V}}(\langle \mathcal{P}(x,w), \mathcal{V}(x)\rangle)$ is bounded by $\mathsf{poly}(\kappa, |x|)$.
Completeness: $\forall \kappa, (x,w) \in \mathcal{R} : \mathsf{out}_{\mathcal{V}}\langle \mathcal{P}(x,w), \mathcal{V}(x)\rangle = 1$.

Definition 5.1 essentially assumes "classic" PPT algorithms, but it will be evident that our techniques do not require this. We do not define soundness, but note that computational soundness is easily handled via truncation to a PPT adversary. The terms proof and argument systems are often used interchangeably (and we also do this). Strictly speaking, *proof* systems require unconditional soundness and allow unbounded provers. All our exemplary proof systems [GMW86; GK96; Lin13; Ros04; KP01; PTV14] have efficient provers, hence are also argument systems.

5.1. Zero-knowledge

Definition 5.2. Let $\mathcal{T}, \mathcal{S} \in \{\mathcal{PPT}, \mathcal{CPPT}, \mathcal{EPT}, \mathcal{CEPT}\}$. Let $(\mathcal{P}, \mathcal{V})$ be an argument system. A **universal simulator** Sim takes as input $(\mathsf{code}(\mathcal{V}^*), x, \mathit{aux})$ and simulates $\mathcal{V}^*$'s output. Let $(\mathcal{I}, \mathcal{V}^*, \mathcal{D})$ be an adversary. We define the real and ideal executions as

$$\mathsf{Real}_{\mathcal{I},\mathcal{V}^*}(\kappa) \coloneqq (\mathit{state}, \mathsf{out}_{\mathcal{V}^*}\langle \mathcal{P}(x,w), \mathcal{V}^*(x, \mathit{aux})\rangle)$$
$$\text{and} \quad \mathsf{Ideal}_{\mathcal{I},\mathsf{Sim}(\mathsf{code}(\mathcal{V}^*))}(\kappa) \coloneqq (\mathit{state}, \mathsf{Sim}(\mathsf{code}(\mathcal{V}^*), x, \mathit{aux}))$$

where $(x, w, \mathit{aux}, \mathit{state}) \leftarrow \mathcal{I}$ and $(x,w) \in \mathcal{R}$, else Real and Ideal return a failure symbol, say $\perp$. We omit the input $\mathsf{code}(\mathcal{V}^*)$ to Sim, if it is clear from the context. The advantage of $(\mathcal{I}, \mathcal{V}^*, \mathcal{D})$ is

$$\mathrm{Adv}^{\mathrm{zk}}_{\mathcal{I},\mathcal{V}^*,\mathcal{D}}(\kappa) \coloneqq |\mathbb{P}(\mathcal{D}(\mathsf{Real}_{\mathcal{I},\mathcal{V}^*}(\kappa)) = 1) - \mathbb{P}(\mathcal{D}(\mathsf{Ideal}_{\mathcal{I},\mathsf{Sim}}(\kappa)) = 1)|.$$

A (designated) adversary $(\mathcal{I}, \mathcal{V}^*, \mathcal{D})$ is $\mathcal{T}$-time if $\mathsf{time}_{\mathcal{I}+\mathcal{P}+\mathcal{V}^*+\mathcal{D}}(\mathcal{D}(\mathsf{Real}_{\mathcal{I},\mathcal{V}^*})) \in \mathcal{T}$.

The argument is **(uniform) (auxiliary input) zero-knowledge** against $\mathcal{T}$-time adversaries w.r.t. $\mathcal{S}$-time Sim, if for any $\mathcal{T}$-time adversary $(\mathcal{I}, \mathcal{V}^*, \mathcal{D})$:

- $\mathsf{time}_{\mathcal{I}+\mathsf{Sim}+\mathcal{D}}(\mathcal{D}(\mathsf{Ideal}_{\mathcal{I},\mathsf{Sim}})) \in \mathcal{S}$. The runtime of Sim includes whatever time is spent to emulate $\mathcal{V}^*$. In a (generalized) sense, Sim is weakly $(\mathcal{T}, \mathcal{S})$-efficient relative to $\mathcal{P}$, see Definition E.12.
- $\mathrm{Adv}^{\mathrm{zk}}_{\mathcal{I},\mathcal{V}^*,\mathcal{D}}(\kappa)$ is negligible

Some more remarks are in order.

Remark 5.3. In the uniform complexity setting, *existential* and *universal* simulation are equivalent. The adversary $\mathcal{V}_{\mathrm{univ}}$, which executes *aux* as its code, is universal, see Appendix E.4.4.

Remark 5.4 (Reductions to PPT). One may expect that, by a standard reduction to PPT, w.l.o.g. $\mathcal{I}$ and $\mathcal{D}$ are a priori PPT. This is true when verifying the output *quality* of Sim. However, it is false when verifying the *efficiency* of Sim. The cause are *expected* poly-size inputs. See also Example E.26.

Remark 5.5 (Efficiency of the simulation). Definition 5.2 only ensures that Sim is *weakly* efficient relative to $\mathcal{P}$, i.e. we have no tightness bounds. Relative efficiency with tightness bounds is an unconditional property, and hence not possible if zero-knowledge holds only computationally.

In the definition, using $\mathsf{time}_{\mathsf{Sim}}(\mathcal{D}(\mathsf{Ideal}_{\mathcal{I},\mathsf{Sim}})) \in \mathcal{S}$ instead of $\mathsf{time}_{\mathcal{I}+\mathsf{Sim}+\mathcal{D}}(\mathcal{D}(\mathsf{Ideal}_{\mathcal{I},\mathsf{Sim}})) \in \mathcal{S}$ is equivalent, since $\mathcal{I}$ is not influenced, and $\mathcal{D}$ is w.l.o.g. a priori PPT.

Definition 5.2 can be extended to proof systems with unbounded provers, but technical artefacts can arise, see Remark E.21.

Remark 5.6 ("Environmental" distinguishing: Why $\mathcal{I}$ outputs *state*). In Definition 5.2, we allow $\mathcal{I}$ to output *state*, effectively making $(\mathcal{I}, \mathcal{D})$ into a stateful distinguishing "environment". Viewing Sim and $\mathcal{P}$ as oracles, this corresponds to oracle indistinguishability. Without this, the security does not obviously help when used as a subprotocol, since a protocol is effectively a (stateful) distinguishing environment. Definition 5.2 is discussed in-depth in Appendix E.4.1. Here, we only note that in the non-uniform classical PPT setting, it coincides with the standard definition.

Remark 5.7. We seldom mention non-uniform zero-knowledge formulations in the rest of this work. Our definitions, constructions and proofs make timed bb-rw use of the adversary, and therefore apply in the non-uniform setting without change.

5.2. Application to graph 3-colouring

To exemplify the setting, the technical challenges, and our techniques, we use the constant-round zero-knowledge proof of Goldreich and Kahan [GK96] as a worked example. We only prove zero-knowledge, as completeness and soundness are unconditional. Formal definitions of commitment schemes are in Appendix B.1. We assume *left-or-right (LR) oracles* in the hiding experiment for commitment schemes. Intuitively, we assume a built-in hybrid argument. (Security against CEPT adversaries follows from security against PPT adversaries by a simple truncation argument.)

5.2.1. The protocol

We recall $\mathrm{G3C}_{\mathrm{GK}}$ from Section 1.2. It requires two different commitments schemes; $\mathsf{Com}^{(\mathrm{H})}$ is perfectly hiding, $\mathsf{Com}^{(\mathrm{B})}$ is perfectly binding. See [GK96] for the exact requirements. We assume non-interactive commitments for simplicity.

- **(P0)** $\mathcal{P}$ sends $\mathrm{ck}_{\mathrm{hide}} \leftarrow \mathsf{Gen}^{(\mathrm{H})}(\kappa)$. ($\mathrm{ck}_{\mathrm{bind}} \leftarrow \mathsf{Gen}^{(\mathrm{B})}(\kappa)$ is deterministic.)
- **(V0)** $\mathcal{V}$ randomly picks challenge edges $e_i \leftarrow E$ for $i = 1, \dots, N = \kappa \cdot \mathsf{card}(E)$, commits to them as $c^e_i = \mathsf{Com}^{(\mathrm{H})}(\mathrm{ck}_{\mathrm{hide}}, e_i)$, and sends all c^e_i.
- **(P1)** $\mathcal{P}$ picks randomized colourings ψ_i for all $i = 1, \dots, N$ and commits to all node colours for all graphs in (sets of) commitments $\{\{c^{\psi}_{i,j}\}_{j \in V}\}_{i=1,\dots,N}$ using $\mathsf{Com}^{(\mathrm{B})}$. $\mathcal{P}$ sends all $c^{\psi}_{i,j}$ to $\mathcal{V}$.
- **(V1)** $\mathcal{V}$ opens the commitments c^e_i to e_i for all i.
- **(P2)** $\mathcal{P}$ aborts if any opening is invalid ($e_i \notin E$). Otherwise, for all iterations $i = 1, \dots, n$, $\mathcal{P}$ opens the commitments $c^{\psi}_{i,a}$, $c^{\psi}_{i,b}$ for the colours of the nodes of edge $e_i = (a, b)$ in repetition i.
- **(V2)** $\mathcal{V}$ aborts iff any opening is invalid, any edge not correctly coloured, or if $\mathrm{ck}_{\mathrm{hide}}$ is bad.

Checking $\mathrm{ck}_{\mathrm{hide}}$ only at the end of the weakens the requirements on VfyCK. In [GK96], it receives the setup randomness as additional input. But this is irrelevant for zero-knowledge.

5.2.2. Proof of zero-knowledge

Our goal is to show the following lemma.

Lemma 5.8. *Suppose $\mathsf{Com}^{(\mathrm{H})}$ and $\mathsf{Com}^{(\mathrm{B})}$ are a priori PPT algorithms. Then protocol $\mathrm{G3C}_{\mathrm{GK}}$ in Section 5.2.1 is zero-knowledge against CEPT adversaries with a bb-rw CEPT simulator. Let $(\mathcal{I}, \mathcal{V}^*)$ be a CEPT adversary and suppose $T \coloneqq \mathsf{time}_{\mathcal{P}+\mathcal{V}^*}(\mathsf{Real}_{\mathcal{I},\mathcal{V}^*})$ is (t, ε)-time. Then Sim handles $(\mathcal{I}, \mathcal{V}^*)$ in virtually expected time $(t', 2\varepsilon + \varepsilon')$. Here ε' stems from an advantage against the hiding property of $\mathsf{Com}^{(\mathrm{B})}$, hence ε' negligible. If the time to compute a commitment depends only on the message length, then t' is roughly $2t$.*

Our proof differs from that in [GK96] on two accounts: First, we do not use the runtime normalization procedure in [GK96]. This is because a negligible deviation from EPT is absorbed into the CEPT virtuality, namely ε'. Second, we handle designated CEPT adversaries. In particular, the runtime classes of simulator and adversary coincide. We first prove the result for *perfect* EPT adversaries.

Lemma 5.9. *The claims in Lemma 5.8 hold if $T \in \mathcal{EPT}$, i.e. $\varepsilon = 0$.*

Proof sketch. We proceed in game hops. The initial game being $\mathsf{Real}_{\mathcal{I},\mathcal{V}^*}$ and the final game being $\mathsf{Ideal}_{\mathcal{I},\mathsf{Sim}}$. We consider (timed) bb-rw simulation.

Game G_0 is the real protocol. The output is the verifier's output and *state* (from $\mathcal{I}$). From now on, we ignore the *state* output, since no game hop affects it.

Game G_1: If the verifier opens the commitments in (V1) correctly, the game repeatedly rewinds it to (P1) using fresh prover randomness, until it obtains a second run where $\mathcal{V}^*$ unveils the commitments correctly (in (V1)). The output is $\mathcal{V}^*$'s output at the end of this second successful run. If the verifier failed in the first run, the protocol proceeds as usual. The outputs of G_1 and G_0 are identically distributed. It can be shown that this modification preserves (perfect) EPT of the overall game, i.e. G_1 is perfect EPT. More precisely, the (virtually) expected time is about $2t$ (plus emulation overhead). To see this, use that each iteration executes $\mathcal{P}$'s code with fresh randomness (whereas the bb-rw $\mathcal{V}^*$ has fixed randomness), and therefore, then number of rewinds is geometrically distributed. Since $1 + p\sum_{i=1}^{\infty} i \cdot p(1-p)^{i-1} = 2$ is the expected number of iterations, assuming probability p that $\mathcal{V}^*$ opens the commitment in step (V0), we see that the (average) number of iterations is 2, hence runtime doubles at most.

Game G_2: We assume that both (valid) openings of $\mathcal{V}^*$'s commitments in (P1) open to the same value. Otherwise, G_2 outputs `ambig`, indicating equivocation of the commitment. This modification hardly affects the runtime, so it is still bounded roughly by $2t$. The probability for `ambig` is negligible, since one can (trivially) reduce to an adversary against the binding property of $\mathsf{Com}^{(\mathrm{B})}$. That is, there is an adversary $\mathcal{B}$ such that $|\mathbb{P}(\mathcal{D}(\mathsf{out}(\mathsf{G}_2))) - \mathbb{P}(\mathcal{D}(\mathsf{out}(\mathsf{G}_1)))| = \mathrm{Adv}^{\mathrm{bind}}_{\mathsf{Com}^{(\mathrm{B})}}(\mathcal{B})$.

In **Game** G_3, the initial commitments (in (P1)) to 3-colourings are replaced with commitments to 0. These commitments are never opened. Thus, we can reduce distinguishing Games 2 and 3 to breaking the hiding property of $\mathsf{Com}^{(\mathrm{B})}$ modelled as left-or-right indistinguishability. More precisely, the reduction constructs real resp. all-zero colourings, and uses the *LR-challenge commitment oracle* $\mathcal{O}_b$ which receives two messages (m_0, m_1) and commits to m_b. Use m_0 to commit to the real colouring *(left)*, whereas m_1 is the all-zero colouring *(right)*. The modification of G_2 to "oracle committing" yields an EPT Game $\mathsf{G}_{2'}$ (instantiated with $\mathcal{O}_0$). The modification of G_3 to $\mathsf{G}_{3'}$ (with $\mathcal{O}_1$) is CEPT. This follows immediately from the standard reduction, because Games $\mathsf{G}_{2'}$ and $\mathsf{G}_{3'}$ differ only in their oracle, and the case of $\mathcal{O}_0$ is EPT. More precisely, the standard reduction applied to $\mathcal{O}_0$ and $\mathcal{O}_1$ yields an adversary $\mathcal{B}$ such that $|\mathbb{P}(\mathcal{D}(\mathsf{out}(\mathsf{G}_2))) - \mathbb{P}(\mathcal{D}(\mathsf{out}(\mathsf{G}_1)))| \geq \frac{1}{4}\,\mathrm{Adv}^{\mathrm{hide}}_{\mathsf{Com}^{(\mathrm{B})}}(\mathcal{B})$ infinitely often, assuming $\mathcal{B}$ has non-negligible advantage.

Consequently, Game $\mathsf{G}_{3'}$ is efficient with (oracle) runtime $T_{3'} \overset{c}{\approx} T_{2'}$, and the output distributions of Games $\mathsf{G}_{2'}$ and $\mathsf{G}_{3'}$ are indistinguishable. Finally, note that Game G_3 and $\mathsf{G}_{3'}$ only differ by (not) using oracle calls. Incorporating these oracles does not affect CEPT (as $\mathcal{O}_1$ is an *a priori* PPT oracle). Thus, G_3 is efficient (i.e. CEPT) as well. Assuming the time to compute a commitment depends only on the message length, a precise analysis shows, that the (virtually) expected time is affected negligibly (up to machine model artefacts).

In **Game** G_4, the commitments in the reiterations of (P1) are replaced by commitments to a pseudo-colouring, that is, at the challenge edge, two random different colours are picked, and all other colours are set to 0. If $\mathcal{V}^*$ equivocates, the game outputs `ambig`. The argument for efficiency and indistinguishability of outputs is analogous to the step from Game G_2 to Game G_3. It reduces all replacements to the hiding property in a single step. This is possible since our definition of hiding is left-or-right oracle indistinguishability with an arbitrary number of challenge commitments. As before, a precise analysis shows that the (virtually) expected time is affected negligibly.

All in all, if G_0 runs in (virtually) expected time t, then G_4 runs in expected time $\approx 2t$, ignoring the overhead introduced by bb-rw emulation, etc. Moreover, the output is indistinguishable, i.e. overall $|\mathbb{P}(\mathcal{D}(\mathsf{out}(\mathsf{G}_4))) - \mathbb{P}(\mathcal{D}(\mathsf{out}(\mathsf{G}_0)))| \leq \mathsf{negl}$.

The simulator is defined as in G_4: It makes a first test-run with an all-zeroes colouring. If the verifier does not open its challenge, Sim aborts (like the real prover). Otherwise, it rewinds $\mathcal{V}^*$ (and uses

pseudo-colourings) until $\mathcal{V}^*$ opens the challenge commitment again, and outputs the verifier's final output of this run (or `ambig`). (To prevent non-halting executions, we may abort after, e.g., $2^{2\kappa}$ steps. But this is not necessary for our results.) □

We point out some important parts: First, in Game G_1, rewinding and its preservation of EPT is unconditional. That is, rewinding is separated from the computational steps happening after it. Second, since the simulator's time per iteration is roughly that of the prover, the total simulation time is CEPT (and roughly virtually expected $2t$). Third, with size-guarded security (Appendix E.4.3), we could have argued efficiency much simpler and coarser. It would suffice if the runtime per rewind is polynomial in the input size (not counting $\mathcal{V}^*$).

There is only one obstacle to extend our result to CEPT adversaries. It is not obvious, whether the introduction of rewinding in G_1 preserves CEPT. Fortunately, this is quite simple to see: The probability that a certain commitment is sent in (P1) increases, since the verifier is rewound and many commitments may be tried. However, the probability only increases by a factor of 2. Thus, "bad" queries are only twice as likely as before.

More concretely, using Lemma 3.12, we obtain a $\mathcal{I}'$ and $\mathcal{O}'$ which output `timeout` in case of "bad" queries. By the above claim, the probability for `timeout` at most doubles. Thus, the virtuality of G_1 is at most twice that of G_0, (and the virtually expected runtime is roughly doubled as well). Hence, G_1 is CEPT. We show this in more detail in the following proof.

Proof sketch of Lemma 5.8. G_0 **to** G_1: Fix the first message $\mathsf{ck}_{\mathsf{hide}}$ of $\mathcal{P}$ to $\mathsf{bbrw}(\mathcal{V}^*)$ and the randomness of $\mathcal{V}^*$ (which is fixed since we consider a bb-rw oracle). Let $p_b(c)$ be the probability, that in protocol step (P1) G_b sends $c = \{\{c_{i,j}^{\psi}\}_{j\in V}\}_{i=1,\dots,N}$ to $\mathsf{bbrw}(\mathcal{V}^*)$ at least once. (For G_0, also at most once. But rewinding in G_1 increases the chances.) Let γ_i denote the i-th query sent in step (P1) (or $\perp$ if none was sent), let the random variable I denote the total number of queries. Then

$$\begin{aligned} p_1(c) \;&=\; \mathbb{P}(\exists j \le i\colon \gamma_j = c \wedge I \le j) \;\le\; \sum_{i=1}^{\infty} \mathbb{P}(I \ge i \wedge \gamma_i = c) \\ &=\; \sum_{i=1}^{\infty} \mathbb{P}(I \ge i)\,\mathbb{P}(\gamma_i = c \mid I \ge i) \;\le\; \sum_{i=1}^{\infty} \mathbb{P}(I \le i)\cdot p_0(c) \;=\; \mathbb{E}(I)\cdot p_0(c). \end{aligned}$$

In the penultimate equality, we use that, for any fixed i, γ_i is a fresh random commitment (or never sampled, if $I < i$). As argued before, $\mathbb{E}(I) = 2$, hence $p_1(c) \le 2p_0(c)$. Thus, the probability $p_1(c)$ for G_1 to issue query c is at most twice that of G_0. Evidently, the derivation still holds for a variable first message, i.e. the full (logical) query $\mathit{query} = (\mathsf{ck}_{\mathsf{hide}}, c)$. Next, we conclude from this, that the virtuality at most doubles.

By an application of Lemma 3.12, we can assume a perfect EPT $\mathcal{V}'$ derived from $\mathcal{V}^*$, i.e. $\mathsf{time}_{\mathcal{V}'}(\mathsf{G}_0)$ is EPT. We can then use $\mathcal{V}'$ the transition of from Game G_0 to G_1. Recall that $\mathcal{V}^*$ and $\mathcal{V}'$ are equal until `timeout` by construction.

Denote by G_0' the modification of G_0 which uses $\mathcal{V}'$ instead of $\mathcal{V}^*$, and let G_0' immediately output `timeout` if $\mathcal{V}'$ does. Then $\mathsf{time}_{\mathcal{V}'}(\mathsf{G}_0')$ is EPT by construction, and essentially equals the virtual expected time of $\mathsf{time}_{\mathcal{V}^*}(\mathsf{G}_0)$. The statistical difference $\Delta(\mathsf{G}_0, \mathsf{G}_0')$ is exactly the probability that $\mathcal{V}'$ outputs `timeout`. Let G_1' be defined analogously to G_0'.

Let $\texttt{timeout}(\mathit{query})$ be 1 if *query* causes a `timeout` and 0 otherwise. Then

$$\begin{aligned} \mathbb{P}_{\mathsf{G}_1'}(\texttt{timeout}) &= \sum_{\mathit{query}} \texttt{timeout}(\mathit{query})\cdot p_1(\mathit{query}) \\ &\le 2\sum_{\mathit{query}} \texttt{timeout}(\mathit{query})\cdot p_0(\mathit{query}) = 2\cdot \mathbb{P}_{\mathsf{G}_0'}(\texttt{timeout}). \end{aligned}$$

Since the probability for `timeout` bounds the virtuality if we use $\mathcal{V}^*$ instead of $\mathcal{V}'$, this shows that G_1 is CEPT, with virtuality 2ε. If G_0 always halts, the outputs of G_1 and G_0 are identically distributed. In

general, the statistical difference is (at most) $2 \cdot \mathbb{P}(\mathsf{G}_0 = \texttt{nohalt})$; this follows as for virtuality, which must encompass the probability of non-halting executions. Conditioned on halting executions, the distributions G_0 and G_1 are identical. The transition to G_2 now relies on the standard reduction, all other steps of Lemma 5.9 apply literally. □

We abstract the above proof strategy in Section 6, to cover a large class of proof systems.

Remark 5.10. The simulator in [GK96] is also a CEPT simulator. For a proof, proceed as in Lemma 5.9. The advantage of [GK96] is, that the simulator handles adversaries which are *a priori PPT*, as well as *EPT w.r.t. any reset attack* [Gol10], without introducing any "virtuality", i.e. the simulation is EPT. On the other hand, it increases virtuality by a larger factor.

5.3. Sequential composition of zero-knowledge

The formulation of sequential security is not merely sequential *repetition*, but considers adaptive choices of inputs. With this, our notion and proof is very close to modular sequential composition for SFE.

5.3.1. Security definition

To model adaptive inputs, we replace the input generator $\mathcal{I}$ by an "environment" $\mathcal{E}$. This "environment" $\mathcal{E}$ provides all inputs for the protocol, but does not participate in the protocol execution itself, it only learns the participants final outputs. This definition of sequential composition allows adaptive sequential executions.

Informally, our definition of sequential zero-knowledge can be summarized as follows: Instead of indistinguishability of $\langle \mathcal{P}, \mathcal{V}^* \rangle$ and $\mathsf{Sim}(\mathcal{V}^*)$, we assume indistinguishability of $\mathsf{rep}(\langle \mathcal{P}, \mathcal{V}^* \rangle)$ and $\mathsf{rep}(\mathsf{Sim}(\mathcal{V}^*))$, i.e. indistinguishability under repeated trials (Definition 2.15).

Definition 5.11 (Sequential zero-knowledge). Let $\mathcal{T}, \mathcal{S} \in \{\mathcal{PPT}, \mathcal{CPPT}, \mathcal{EPT}, \mathcal{CEPT}\}$. Let $(\mathcal{P}, \mathcal{V})$ be an argument system. A **universal simulator** Sim takes as input $(x, \mathsf{code}(\mathcal{V}^*), \mathit{aux})$ and simulates $\mathcal{V}^*$'s output. Let $(\mathcal{E}, \mathcal{V}^*, \mathcal{D})$ be an adversarial environment $\mathcal{E}$ and an adversarial verifier $\mathcal{V}^*$and a distinguisher $\mathcal{D}$. The environment is given access one of two oracles $\mathcal{O}_{\mathcal{P}}$, $\mathcal{O}_{\mathsf{Sim}}$, which take as input (x, w, aux) and

- $\mathcal{O}_{\mathcal{P}}(x, w, \mathit{aux})$ returns $\mathsf{out}_{\mathcal{V}^*}\langle \mathcal{P}(x, w), \mathcal{V}^*(\mathit{aux}) \rangle$. $(\mathcal{O}_{\mathcal{P}} \mathrel{\widehat{=}} \mathsf{rep}(\langle \mathcal{P}(\cdot), \cdot \rangle))$
- $\mathcal{O}_{\mathsf{Sim}}(x, w, \mathit{aux})$ returns $\mathsf{Sim}(x, \mathsf{code}(\mathcal{V}^*), \mathit{aux})$. $(\mathcal{O}_{\mathsf{Sim}} \mathrel{\widehat{=}} \mathsf{rep}(\mathsf{Sim}(\cdot)))$

We assume that both oracles reject (say with $\perp$) if $(x, w) \notin \mathcal{R}$. We consider two executions, a real and an ideal one, defined by:

$$\mathsf{Real}_{\mathcal{E},\mathcal{V}^*}(\kappa) \coloneqq \mathsf{out}_{\mathcal{E}}\langle \mathcal{E}, \mathsf{rep}(\mathcal{O}_{\mathcal{P}}) \rangle$$
$$\text{and} \quad \mathsf{Ideal}_{\mathcal{I},\mathsf{Sim}}(\kappa) \coloneqq \mathsf{out}_{\mathcal{E}}\langle \mathcal{E}, \mathsf{rep}(\mathcal{O}_{\mathsf{Sim}}) \rangle$$

We define $\mathsf{Real}_{\mathcal{E},\mathcal{V}^*}(\kappa)$ to be the execution of $(\mathcal{E}, \mathcal{V}^*)$ with $\mathcal{O}_{\mathcal{P}}$, and $\mathsf{Ideal}_{\mathcal{E},\mathsf{Sim}}(\kappa)$ the execution with $\mathcal{O}_{\mathsf{Sim}}$. The distinguishing advantage of $(\mathcal{E}, \mathcal{V}^*, \mathcal{D})$ is

$$\mathrm{Adv}^{\mathrm{zk}}_{\mathcal{E},\mathcal{V}^*,\mathcal{D}}(\kappa) \coloneqq |\mathbb{P}(\mathcal{D}(\mathsf{Real}_{\mathcal{E},\mathcal{V}^*}(\kappa)) = 1) - \mathbb{P}(\mathcal{D}(\mathsf{Ideal}_{\mathcal{E},\mathsf{Sim}}(\kappa)) = 1)|.$$

A (designated) adversary $(\mathcal{E}, \mathcal{V}^*, \mathcal{D})$ is $\mathcal{T}$-time if $\mathsf{time}_{\mathcal{E}+\mathcal{P}+\mathcal{V}^*+\mathcal{D}}(\mathcal{D}(\mathsf{Real}_{\mathcal{E},\mathcal{V}^*})) \in \mathcal{T}$.

The argument system is **(uniform) sequential zero-knowledge** against $\mathcal{T}$-time adversaries w.r.t. $\mathcal{S}$-time Sim, if for any $\mathcal{T}$-time adversary $(\mathcal{E}, \mathcal{V}^*)$:

- $\mathsf{time}_{\mathcal{E}+\mathsf{Sim}+\mathcal{D}}(\mathcal{D}(\mathsf{Ideal}_{\mathcal{E},\mathsf{Sim}})) \in \mathcal{S}$, that is, $\mathsf{rep}(\mathcal{O}_{\mathsf{Sim}(\mathsf{code}(\mathcal{A}))})$ is weakly $(\mathcal{T}, \mathcal{S})$-efficient relative to $\mathsf{rep}(\mathcal{O}_{\langle \mathcal{P},\mathcal{V}^* \rangle})$.
- $\mathrm{Adv}^{\mathrm{zk}}_{\mathcal{E},\mathcal{V}^*,\mathcal{D}}(\kappa)$ is negligible

We also say that protocols with sequential zero-knowledge simulators *compose sequentially*. We dropped the input generator $\mathcal{I}$, since its complexity class is the same as that of the environment $\mathcal{E}$, For clarity, we did not "include" $\mathcal{D}$ in $\mathcal{E}$, although the resulting definition would be equivalent.

A few remarks are in order. One could fix the universal adversary $\mathcal{V}_{\text{univ}}$ (which executes a given program) as $\mathcal{V}^*$ in Definition 5.11. For compatibility with Definition 5.2, we allow any $\mathcal{V}^*$.

Remark 5.12. Sequential zero-knowledge where $\mathcal{E}$ is restricted to a single query is equivalent to auxiliary input zero-knowledge. This is easily seen since Definition 5.2 allows $\mathcal{I}$ to pass *state* to $\mathcal{D}$.

Caution 5.13. Taken literally, Definition 5.11 is unsuitable for expected time. Inefficiencies similar to the setting of bb-rw oracles arise. However, as with rewinding strategies, we use the usual convenient notation. We leave implicit, that an efficient implementation which is logically equivalent is easily derived.[44] In Definition 5.11, passing the state of $\mathcal{V}^*$ via *aux* runs into these problems. For simplicity, assume *aux* is shared memory between $\mathcal{E}$ and $\mathcal{V}^*$.

5.3.2. Sequential composition lemma

We are now ready to state and prove the sequential composition lemma for zero-knowledge.

Lemma 5.14 (Sequential composition lemma). *Let $(\mathcal{P}, \mathcal{V})$ be an argument system. Suppose* Sim *is a simulator for auxiliary input zero-knowledge (which handles CEPT adversaries in CEPT). Then $(\mathcal{P}, \mathcal{V})$ is sequential zero-knowledge (with the same simulator* Sim *which also handles CEPT adversaries against sequential zero-knowledge in CEPT).*

The proof is an almost trivial consequence of the hybrid lemma.

Proof. Let $(\mathcal{E}, \mathcal{V}^*, \mathcal{D})$ be a CEPT adversary. Let $\mathcal{O}_{\mathcal{P}}(x, w, \mathit{aux})$ and $\mathcal{O}_{\mathsf{Sim}}(x, w, \mathit{aux})$ be as in Definition 5.11. By definition,

$$\mathsf{Real}_{\mathcal{E},\mathcal{V}^*}(\kappa) = \mathsf{out}_{\mathcal{E}}\langle \mathcal{E}, \mathsf{rep}(\mathcal{O}_{\mathcal{P}})\rangle \qquad \text{and} \qquad \mathsf{Ideal}_{\mathcal{I},\mathsf{Sim}}(\kappa) = \mathsf{out}_{\mathcal{E}}\langle \mathcal{E}, \mathsf{rep}(\mathcal{O}_{\mathsf{Sim}})\rangle$$

Define a distinguisher $\mathcal{A}$ for $\mathsf{rep}(\mathcal{O}_{\mathcal{P}})$ and $\mathsf{rep}(\mathcal{O}_{\mathsf{Sim}})$ as $\mathcal{D}(\mathsf{out}_{\mathcal{E}}\langle \mathcal{E}, \mathsf{rep}(\mathcal{O})\rangle)$. Now, we are in the usual setting of oracle (in)distinguishability. Since Sim is an auxiliary input zero-knowledge simulator for $(\mathcal{P}, \mathcal{V})$, we have that $\mathcal{O}_{\mathsf{Sim}}$ is weakly efficient relative to $\mathcal{O}_{\mathcal{P}}$ and that $\mathcal{O}_{\mathcal{P}} \overset{c}{\approx} \mathcal{O}_{\mathsf{Sim}}$. Thus, the hybrid lemma for CEPT, Lemma 4.7, is applicable. Hence $\mathsf{rep}(\mathcal{O}_{\mathcal{P}})$ is weakly relative efficient to $\mathsf{rep}(\mathcal{O}_{\mathsf{Sim}})$ and $\mathsf{rep}(\mathcal{O}_{\mathcal{P}}) \overset{c}{\approx} \mathsf{rep}(\mathcal{O}_{\mathsf{Sim}})$. This concludes the proof. □

6. Benign simulation

In this section, we define benign simulation. This abstracts the proof strategy for G3C_{GK} in Section 5.2. Namely, we define *rewinding strategies* to abstract the rewinding step, and we define "*simple assumption*" to abstract the left-right hiding and binding property of the commitment. Put together, we define benign simulators as simulators which have a proof of security analogous to the one of G3C_{GK}.

6.1. Rewinding strategies

Rewinding strategies encapsulate the rewinding schedule of a simulator. Unlike simulators, their properties are unconditional.

[44] The problem is that passing around state is extremely wasteful, and involves copying the state to and from message interfaces. Generally speaking, almost anything, which does not go over a "real" network, should not be passed by copying. This can be solved in any number of ways. E.g. allow shared memory/tapes between machines, or introduce an additional interactive machine which represents that shared tape, and pass around (interface) access to memory/machine, and so on. It should be evident that it is easy but tedious to formalize this.

Reminder (Black-box queries). By abuse of notation, we typically write $\mathsf{A}^{\mathcal{O}}$ instead of $\mathsf{A}^{\mathsf{bbrw}(\mathcal{O})}$, if it is understood that A has bb-rw access to $\mathcal{O}$. Our presentation treats $\mathsf{bbrw}(\mathcal{O})$ as a NextMsg oracle, but it is understood that a *logical* query $(m_1, \ldots, m_\ell)$ is implemented efficiently by a *short* handle to the state of $\mathsf{bb\text{-}rw}(\mathcal{O})$ after processing $(m_1, \ldots, m_{\ell-1})$, and the message m_ℓ in that state.

6.1.1. Definitions

Our definition of rewinding strategies is specialized for zero-knowledge, but it generalizes to other settings easily.

Definition 6.1. A **rewinding strategy** RWS for a proof system $(\mathcal{P}, \mathcal{V})$ is an oracle algorithm with timed bb-rw access to the (malicious deterministic) verifier $\mathcal{V}^*$. The output of RWS is a state of $\mathsf{bbrw}(\mathcal{O})$ (or an abort message), which we denote by the (logical) query leading to it.

A rewinding strategy RWS has **runtime tightness** poly, if the following holds: Let be $(\mathcal{I}, \mathcal{V}^*)$ any adversary (modelled as a timeful oracle). Let $T \coloneqq \mathsf{time}_{\mathcal{P}+\mathcal{V}^*}(\langle \mathcal{P}, \mathcal{V}^* \rangle_{\mathcal{I}})$, and let $S \coloneqq \mathsf{time}_{\mathsf{RWS}+\mathcal{V}^*}(\mathsf{RWS}^{\mathcal{V}^*(x,aux)}(x,w))$ with input distribution $\mathcal{I}$. Then $\mathbb{E}(S) \leq \mathsf{poly} \cdot \mathbb{E}(T)$ for all $(\mathcal{I}, \mathcal{V}^*)$.[45]

Equivalently, for *deterministic* timeful $\mathcal{I}$, i.e. any sequence $(x_\kappa, w_\kappa, \mathit{aux}_\kappa) \in \mathcal{R}$ and any *deterministic* timeful $\mathcal{V}^*$, the analogous claim holds.

The notion of *runtime tightness* of RWS is strong and unconditional. Up to minor technical details, it is equivalent to the notion of "normal machine" implicit in [Gol10, Definition 6]. The equivalence of using probabilistic and deterministic adversaries follows easily: Certainly, probabilistic covers deterministic. For the converse, one uses the tightness bound poly and linearity of expectation.

Remark 6.2 (Preservation of EPT). It is clear that a rewinding strategy RWS with *polynomial* runtime tightness **preserves EPT**, i.e. in the setting of Definition 6.1, if $\mathsf{time}_{\mathcal{P}+\mathcal{V}^*}(\langle \mathcal{P}, \mathcal{V}^* \rangle_{\mathcal{I}}) \in \mathcal{EPT}$, then $\mathsf{time}_{\mathsf{RWS}+\mathcal{V}^*}(\mathsf{RWS}^{\mathcal{V}^*(x,aux)}) \in \mathcal{EPT}$.

Before we tackle preservation of CEPT, we introduce more parameters of rewinding strategies.

Definition 6.3 (Properties of rewinding strategies.). Let $(\mathcal{P}, \mathcal{V})$ be a proof system and RWS a rewinding strategy. Let $\mathcal{LQ}$ be the set of all possible (logical) queries. Suppose $\mathcal{V}^*$ is some (malicious) deterministic verifier (as a timeful oracle). Let κ, (x, w), *aux* be inputs. Let $\mathit{query} \in \mathcal{LQ}$ be a (logical) query to $\mathsf{bbrw}(\mathcal{V}^*)$. Let $\mathsf{pr}_{\mathrm{real}}(\mathit{query})$ be the probability that, in a real interaction $\langle \mathcal{P}(x,w), \mathcal{V}^*(x, \mathit{aux}) \rangle$, the prover queries *query*, that is[46]

$$\mathsf{pr}_{\mathrm{real}}(\mathit{query}) = \mathbb{P}(\mathit{query} \in \mathsf{qseq}_{\mathcal{P}}(\langle \mathcal{P}(x,w), \mathcal{V}^*(x, \mathit{aux}) \rangle)).$$

Let $\mathsf{pr}_{\mathrm{rws}}(\mathit{query})$ be the probability, that $\mathsf{RWS}^{\mathcal{V}^*}(x,w)$ queries *query*, that is

$$\mathsf{pr}_{\mathrm{rws}}(\mathit{query}) = \mathbb{P}(\mathit{query} \in \mathsf{qseq}_{\mathsf{RWS}}(\mathsf{RWS}^{\mathcal{V}^*(x,aux)}(x,w))).$$

We say a rewinding strategy RWS has **probability tightness** $\mathsf{poly}_{\mathsf{pr}}(\kappa)$ if

$$\mathsf{pr}_{\mathrm{rws}}(\mathit{query}) \leq \mathsf{poly}_{\mathsf{pr}}(\kappa) \cdot \mathsf{pr}_{\mathrm{real}}(\mathit{query})$$

for all queries $\mathit{query} \in \mathcal{LQ}$. (In other words: $\mathsf{D}_{\mathsf{rat}}(\mathsf{pr}_{\mathrm{rws}}/\mathsf{pr}_{\mathrm{real}}) \leq \mathsf{poly}_{\mathsf{pr}}$.)

The output skew of RWS for an execution with $(\mathcal{I}, \mathcal{V}^*)$ (where $(x, w, \mathit{aux}) \leftarrow \mathcal{I}(\kappa)$) is similarly defined by the ratio $\mathsf{D}_{\mathsf{rat}}(Y/X)$ of the output distributions, Y of RWS and X of $\mathcal{P}$ running with $\mathcal{V}^*$ on input sampled by $\mathcal{I}$.[47] We say RWS has **output skew** $\delta = \delta(\kappa)$, if for every (deterministic) $(\mathcal{I}, \mathcal{V}^*)$

[45] We define that $\infty \leq \infty$.

[46] By abuse of notation, we write $\mathsf{qseq}_{\mathcal{P}}(\langle \mathcal{P}(x,w), \mathcal{V}^*(x, \mathit{aux}) \rangle)$ for the sequence of *logical* queries to $\mathsf{NextMsg}_{\mathcal{V}^*}$, i.e. $\mathsf{bbrw}(\mathcal{V}^*)$. Formally, $\mathsf{qseq}_{\mathcal{P}}(\ldots)$ is the sequence of message sent by $\mathcal{P}$, (and $\mathcal{P}$ does not treat $\mathcal{V}^*$ as $\mathsf{bbrw}(\mathcal{V}^*)$). So actually, we consider the sequence of prefixes of $\mathsf{qseq}_{\mathcal{P}}(\ldots)$, which correspond to the logical queries to $\mathsf{bbrw}(\mathcal{V}^*)$ which result in the same execution as $\langle \mathcal{P}, \mathcal{V} \rangle$.

[47] More correctly, X and Y are the distributions of the state of the timed bb-rw $\mathcal{V}^*$.

which halts (with probability 1), the output skew for $(\mathcal{I}, \mathcal{V}^*)$ is at most $1 + \delta(\kappa)$. We say RWS has **perfect output (distribution)** if for all $(\mathcal{I}, \mathcal{V}^*)$ which always halt with probability 1 the output of RWS is distributed identically (that is, $\delta = 0$) to the real execution.

We note that the properties in Definition 6.3 are unconditional. Also, non-halting executions can affect the output distribution, as they will be encountered by RWS with higher probability than in the real execution, increasing the probability that RWS "outputs" `nohalt`. In any situation where *statistical* properties are good enough, one can assume that all algorithms halt (e.g. by truncation or modifying the machine model).

Finally, we define our notion of normality. The definition is similar to Goldreich's definition of normality in [Gol10].[48]

Definition 6.4 (Normal RWS). A rewinding strategy RWS is **normal** if it has polynomial runtime tightness, polynomial probability tightness, and perfect output distribution.

Perfect output distribution is vital for later use of RWS (as a part of security proofs). Negligible output skew would suffice, but natural rewinding strategies seem to satisfy perfect output skew, so we require that for simplicity.

6.1.2. Basic results

Now, we state our main result for normal rewinding strategies.

Lemma 6.5 (Normal rewinding strategies preserve CEPT). *Let* RWS *be a normal rewinding strategy for* $(\mathcal{P}, \mathcal{V})$. *Let* $(\mathcal{I}, \mathcal{V}^*)$ *be a CEPT adversary for zero-knowledge, that is* $\mathsf{time}_{\mathcal{I}+\mathcal{P}+\mathcal{V}^*}(\langle \mathcal{P}, \mathcal{V}^* \rangle_{\mathcal{I}}) \in \mathcal{CEPT}$. *Then* $\mathsf{time}_{\mathcal{I}+\mathsf{RWS}+\mathcal{V}^*}(\mathsf{RWS}^{\mathcal{V}^*(x,aux)}(x,w)) \in \mathcal{CEPT}$, *where* $(x, w, aux, state) \leftarrow \mathcal{I}(\kappa)$.

More precisely, suppose $\mathsf{poly}_{\mathsf{time}}$ *is a runtime tightness and* $\mathsf{poly}_{\mathsf{virt}}$ *a probability tightness of* RWS *(against EPT adversaries). If* $\mathsf{time}_{\mathcal{P}+\mathcal{V}^*}(\langle \mathcal{P}, \mathcal{V}^* \rangle_{\mathcal{I}})$ *is virtually* (t, ε)*-time, then* $\mathsf{time}_{\mathsf{RWS}+\mathcal{V}^*}(\mathsf{RWS}^{\mathcal{V}^*})$ *is virtually* $(\mathsf{poly}_{\mathsf{time}} \cdot t, \mathsf{poly}_{\mathsf{virt}} \cdot \varepsilon)$*-time. In other words,* RWS *is efficient relative to* $\langle \mathcal{P}, \cdot \rangle$ *with runtime tightness* $(\mathsf{poly}_{\mathsf{time}}, \mathsf{poly}_{\mathsf{virt}})$.

The proof exploits that "bad queries", which result in overly long runs of $\langle \mathcal{P}, \mathcal{V}^* \rangle_{\mathcal{I}}$ happen at most polynomially more often with RWS, due to normality. Since bad queries happen with probability ε, the claim follows. A detailed proof follows.

Proof. By Lemma 3.12, we know that there is a modification $\mathcal{V}'$ of $\mathcal{V}^*$ such that $\mathsf{time}_{\mathcal{V}'}(\langle \mathcal{P}, \mathcal{V}' \rangle_{\mathcal{I}})$ is EPT, where $\mathcal{V}'$ is a timeful oracle which aborts bad executions with `timeout`. By normality, also $\mathsf{time}_{\mathcal{V}'}(\mathsf{RWS}^{\mathcal{V}'})$ is EPT. We call a (logical) query *query* $= (m_1, \ldots, m_n)$ to $\mathcal{V}'$ which returns `timeout` a *timeout query*. The probability that such a timeout query happens in a real execution with $\mathcal{P}$ is at most ε (by construction).

The only case where RWS encounters a difference between $(\mathcal{I}, \mathcal{V}^*)$ and $(\mathcal{I}, \mathcal{V}')$ is if RWS asks a timeout query, i.e. if $\mathcal{V}'$ returns `timeout`. By normality of RWS, the probability of asking a timeout query is only polynomially higher than the probability that $\mathcal{P}$ asks a timeout query. The latter is at most ε, hence the former is bounded by $\mathsf{poly}_{\mathsf{virt}} \cdot \varepsilon$. Thus, the runtime $\mathsf{time}_{\mathsf{RWS}+\mathcal{V}^*}(\mathsf{RWS}^{\mathcal{V}^*})$ is CEPT with virtually expected time $(\mathsf{poly}_{\mathsf{time}} t, \mathsf{poly}_{\mathsf{virt}} \varepsilon)$. The claim for the total runtime follows analogously. □

We note that runtime tightness already implies probability tightness (see Remark E.30). However, the implied bounds are far from optimal. Following lemma is a simple way to get a tight(er) bound on probability tightness.

[48] Goldreich remarks [Gol10, Footnote 24] that his notion of normality *of a simulator* is probably satisfied if the runtime analysis is *unconditional*. We separate the analysis into rewinding strategies and indistinguishability transitions, since our notion of runtime and efficiency of simulators is *not unconditional*. Disregarding this, the notions essentially coincide.

Lemma 6.6. *Let* RWS *be a rewinding strategy for* $(\mathcal{P}, \mathcal{V})$*, and let* $(\mathcal{I}, \mathcal{V}^*)$ *be a timeful adversary. Let* $\mathcal{Q}_i \subseteq \mathsf{qseq}_{\mathsf{RWS}}(\mathsf{RWS}^{\mathcal{V}^*})$ *be the list of queries of length* i *from* RWS *to* $\mathsf{bbrw}(\mathcal{V}^*)$*; that is* $\mathcal{Q}_i$ *consists of queries* $(m_1, \ldots, m_i)$*. Let* $Q_i = \mathsf{card}(\mathcal{Q}_i)$*. Note that* $\mathcal{Q}_i$ *and* Q_i *are random variables. Let* $Q = \sum_{i=0}^{\infty} Q_i$ *be the total number of queries. Suppose that for all adversaries,* $\mathbb{E}(Q_i) \leq M_i$ *for some* M_i*.*

Let $\mathsf{pr}_{\mathrm{rws}}(\mathit{query})$ *resp.* $\mathsf{pr}_{\mathrm{real}}(\mathit{query})$ *be the probability that* RWS *resp.* $\mathcal{P}$ *queries query, as in Definition 6.3. Write* $\mathcal{Q}_i[j]$ *for the* j*-th query in* $\mathcal{Q}_i$*. Suppose that for all* i *and all logical queries query of length* i

$$\forall j \in \mathbb{N}_0 \colon \quad \mathbb{P}(\mathit{query} = \mathcal{Q}_i[j] \mid Q_i \geq j) \leq \mathsf{pr}_{\mathrm{real}}(\mathit{query}),$$

where the probability is over the randomness of RWS *and* $\mathcal{P}$*. Then*

$$\mathsf{pr}_{\mathrm{rws}}(\mathit{query}) \leq M_i \cdot \mathsf{pr}_{\mathrm{real}}(\mathit{query}).$$

In particular, the probability tightness of RWS *is bounded by* $M = \max_i M_i$*.*

The basic idea behind Lemma 6.6 is that for any $(i-1)$-length history $m' = (m_1, \ldots, m_{i-1})$, the probability that the prover queries m_i (conditioned on m') is identical to the probability that RWS queries m_i "conditioned on m'". The "conditioning RWS on m'" part needs a suitable definition. In special cases, e.g. "tree-based" rewinding strategies, this can be done hands on. Lemma 6.6 gives a general formalization of this idea (without needing to condition on some m').

It is often (almost) trivial to verify the conditions of Lemma 6.6. Moreover, we are not aware of (natural) rewinding strategies which do not satisfy normality, even outside the context of zero-knowledge.

Proof of Lemma 6.6. The proof is almost trivial. Consider the setting and notation of Lemma 6.6. Let *query* be a logical query of length i. We have

$$\mathbb{P}(\exists j \colon \mathit{query} = \mathcal{Q}_i[j]) \;\leq\; \sum_{j=0}^{\infty} \mathbb{P}(\mathit{query} = \mathcal{Q}_i[j]) \;=\; \sum_{j=0}^{\infty} \mathbb{P}(\mathit{query} = \mathcal{Q}_i[j] \mid Q_i \geq j)\, \mathbb{P}(Q_i \geq j),$$

by a union bound, and we have

$$\sum_{j=0}^{\infty} \mathbb{P}(\mathit{query} = \mathcal{Q}_i[j] \mid Q_i \geq j)\, \mathbb{P}(Q_i \geq j) \;=\; \sum_{j=0}^{\infty} \mathsf{pr}_{\mathrm{real}}(\mathit{query})\, \mathbb{P}(Q_i \geq j) \;\leq\; \mathsf{pr}_{\mathrm{real}}(\mathit{query}) \cdot \mathbb{E}(Q_i).$$

by assumption (and by $\mathbb{E}(Q_i) = \sum_{j=0}^{\infty} \mathbb{P}(Q_i \geq j)$). □

The criterion in Lemma 6.6 is "global" and not "local", making it somewhat inconvenient. Instead of applying Lemma 6.6, it is often simple(r) to derive more precise bounds and directly prove normality.

Remark 6.7 (Partial RWS). A typical proof strategy for normality is to view RWS as a composition of (partial) strategies. For example, many rewinding strategies are "tree-based" and each layer corresponds to a (partial) rewinding strategy, which calls lower layers as substrategies. This approach lends itself to a simple and precise analysis of runtime tightness, probability tightness and "query tightness". For example, if calls to substrategies not skewed, probability tightness behaves multiplicatively. Checking normality like this relies on "local" properties, which by composition yield the "global" properties.

Remark 6.8. Halevi and Micali [HM98] define "valid distributions" [of transcripts] for extraction in the context of proofs of knowledge. Their definition requires that a polynomial number of total executions are made (with the extractor in the role of the verifier), and each execution has a transcript (i.e. queries) which is distributed like for an honest verifier. Separate runs may be stochastically dependent. Lemma 6.6 deals with partial transcripts, expected polynomially many executions, and probability tightness (not runtime), but is otherwise similar to [HM98].

6.1.3. Examples of normal rewinding strategies

We give some examples for rewinding strategies which are normal. Most claims follows easily from their original efficiency analysis.

Example 6.9 (The classic cut-and-choose protocols). The classic protocols for graph 3-colouring, graph hamiltonicity, as well as graph-(non)-isomorphpism [GMW86; Blu86] use normal rewinding strategies.

Example 6.10 (Constant round zero-knowledge). Our motivating example [GK96], the simplification of Rosen [Ros04], and the proof of knowledge of Lindell [Lin13] have normal rewinding strategies.

Example 6.11 (Concurrent zero-knowledge). The concurrent zero-knowledge proof systems of Kilian and Petrank [KP01] and its variation [PTV14] also rely on normal rewinding strategies. Indeed, their strategy is strictly PPT (in oracle-excluded time).

Example 6.12 (Blum coin-toss). The simulator for the coin-toss protocol [Blu81; Lin17] also gives rise to normal rewinding strategies. It is strictly PPT (in oracle-excluded time).

6.2. Simple assumptions and repeated trials

To obtain nice results, we want nice "base assumptions" to reduce security to. We call these "*simple assumptions*". For simplicity, we do not allow (shared) setups, such as a common random string, and are very restrictive w.r.t. the runtime of such oracles.

Definition 6.13 (PPTpa). A timeful oracle $\mathcal{O}$ is **a priori PPT per activation (PPTpa)**, if there is a polynomial poly such that every invocation of $\mathcal{O}$ has runtime bounded by $\mathsf{poly}(\kappa)$.

The property we need from a priori PPTpa is that, if a distinguisher yields an inefficient system, then the oracle is never to blame, i.e. even excluding its runtime, the system is inefficient. There are less strict efficiency notions which satisfy this as well.

Definition 6.14 (Simple assumption). Let C_0 and C_1 be two oracles, induced by algorithms which are *a priori PPT per activation*. The assumption that C_0 and C_1 are indistinguishable (w.r.t. PPT adversaries) called a **simple assumption**. We also say C_0 and C_1 form a simple assumption.

Example 6.15. Many assumptions are simple, for example one-way functions, trap-door one-way permutations, pseudorandom functions, hiding and binding properties of commitments, IND-CPA and IND-CCA security of public key encryption, and so on. Counterexamples are 1-more assumptions, e.g. the one-more RSA assumption. Knowledge assumptions are also not simple. Note that assumptions which can be reduced to simple assumptions need not be simple, e.g. soundness of (non-extractable) proof systems.

By definition, simple assumptions are essentially falsifiable assumptions [Nao03] as defined by Gentry and Wichs [GW10]. However, the (invisible) intent of simple assumptions is that they have a simple notion of repeated trials, and behave well in this setting. Since our primary setting is the plain model, simple assumptions are natural, but we stress that our techniques work for a much broader class of game-based assumptions, including non-falsifiable assumptions.[49]

Simple assumptions are secure under repeated trials against PPT (or CEPT) adversaries.

Lemma 6.16 (Hybrid lemma for simple assumptions). *Let C_0 and C_1 be two oracles forming a simple assumption, and let $q = q(\kappa)$ where $q = \infty$ is allowed. Suppose $\mathcal{D}$ is a CEPT distinguisher for q-repeated trials, with $|\mathrm{Adv}^{dist}_{\mathcal{D},\mathsf{rep}_q(C_0),\mathsf{rep}_q(C_1)}| \geq \varepsilon = 1/\mathsf{poly}$ infinitely often. Suppose $\mathsf{time}_{\mathcal{D}}(\mathcal{D}^{\mathsf{rep}_q(C_0)})$ is bounded by (t_0, ν_0). Let $M(\kappa) \geq \min(q(\kappa), 4\varepsilon^{-1}t_0)$ be an (efficiently computable) polynomial upper bound. Then there is an a priori PPT distinguisher $\mathcal{A}$ with advantage at least $\frac{1}{M}(\frac{\varepsilon}{4} - \nu_0)$ infinitely often.*

[49] Typical 1-more assumptions have a meaningful notion of security under repeated trials as well, but Definition 2.14 is too coarse to capture this, as it postulates independent instances. For example, given two 1-more-dlog oracles for a deterministic group generator, it is easy to win in one of the 1-more dlog instances; but by correlating the repeated oracles, one can also embed a 1-more-dlog challenge.

This immediately yields:

Corollary 6.17. *Let $\mathcal{C}_0$ and $\mathcal{C}_1$ form a simple assumption, in particular, $\mathcal{C}_0 \overset{c}{\approx} \mathcal{C}_1$. Then $\mathsf{rep}(\mathcal{C}_0)$ and $\mathsf{rep}(\mathcal{C}_1)$ form a simple assumption, in particular, $\mathsf{rep}(\mathcal{C}_0) \overset{c}{\approx} \mathsf{rep}(\mathcal{C}_1)$.*

Proof of Lemma 6.16. First apply Corollary 4.2 to get an a priori PPT distinguisher $\mathcal{A}'$. Note that we treat distinguishing under repeated trials as distinguishing $\mathcal{O}_0^* = \mathsf{rep}(\mathcal{C}_0)$ and $\mathcal{O}_1^* = \mathsf{rep}(\mathcal{C}_1)$. Thus, we end up with advantage $\frac{\varepsilon}{4} - \nu_0$ and runtime bound roughly $4\varepsilon^{-1}t_0$, where (t_0, ν_0) is virtually expected time of $\mathcal{D}$ as in Lemma 4.1. In particular, $\mathcal{A}'$ can make at most $M(\kappa)$ queries.

Now, we rely on the efficient implementation of $\mathcal{C}_0, \mathcal{C}_1$ to implement the hybrid distinguisher. The claim follows from the (standard a priori) PPT) hybrid lemma. □

6.3. Benign simulators

Our definition of a benign simulator abstracts the proof strategy for $\mathrm{G3C_{GK}}$. Before we give the definition, we demonstrate the idea.

Example 6.18 (Structure of the security reduction for $\mathrm{G3C_{GK}}$). Consider the protocol $\mathrm{G3C_{GK}}$ in Section 5.2.1 and the security proof in Section 5.2.2. Let $(\mathcal{I}, \mathcal{V}^*, \mathcal{D})$ be an adversary. Since the simulator cannot depend on $\mathcal{I}$ and $\mathcal{D}$, they are of no importance in the following. Indeed, they should be viewed as one entity, the "distinguisher", whereas $\mathcal{V}^*$ is the actual "attacker". Below, we omit the inputs x, w, *aux*.

Let $\mathsf{A}_0 = \mathsf{A}_0(\mathcal{V}^*)$ denote the algorithm $\mathsf{out}_{\mathcal{V}^*}\langle \mathcal{P}, \mathcal{V}^* \rangle$. Let $\widetilde{\mathsf{A}}_0 = \widetilde{\mathsf{A}}_0(\mathcal{V}^*)$ denote the algorithm which introduces all rewindings, as in Section 5.2.2, G_1. Moreover, $\widetilde{\mathsf{A}}_0$ makes any commitment computations into explicit calls to subroutines. (Let us call this *boxing*, and the act of "forgetting" subroutine calls *unboxing*.)

We note the following: For any $\mathcal{V}^*$, $\mathsf{A}_0 \equiv \widetilde{\mathsf{A}}_1$ (i.e. they are perfectly indistinguishable), and if A_0 is efficient, then so is $\widetilde{\mathsf{A}}_0$. More concretely: For every $\mathcal{I}, \mathcal{D}$, if the completed system for A_0 is CEPT (i.e. with inputs sampled by $\mathcal{I}$ and with $\mathcal{D}$ applied to the output of A_0), so is the completed system for $\widetilde{\mathsf{A}}_0$.

Similarly, let $\mathsf{A}_1 \coloneqq \mathsf{Sim}(\mathcal{V}^*)$ and let $\widetilde{\mathsf{A}}_1 = \widetilde{\mathsf{A}}_1(\mathcal{V}^*)$ be the simulator with boxed calls to Com. Clearly, for any $\mathcal{V}^*$, $\widetilde{\mathsf{A}}_1 \equiv \mathsf{A}_1$, and if $\widetilde{\mathsf{A}}_1$ is efficient (i.e. CEPT), so is A_1.

Consider the two indistinguishable oracles $\mathcal{O}_0, \mathcal{O}_1$, which represent the (repeated) binding and hiding experiments in the security proof, squeezed into one oracle. It is straightforward to define an (oracle) algorithm $\mathrm{R} = \mathrm{R}(\mathcal{V}^*)$, which encapsulates the reduction given in the games following G_1 in Section 5.2.2, such that for R, it holds that $\widetilde{\mathsf{A}}_0 \equiv \mathrm{R}^{\mathcal{O}_0}$ and $\mathrm{R}^{\mathcal{O}_1} \equiv \widetilde{\mathsf{A}}_1$. Moreover, $\mathrm{R}^{\mathcal{O}_0}$ is efficient if $\widetilde{\mathsf{A}}_0$ is. Furthermore, since $\mathcal{O}_0$ and $\mathcal{O}_1$ are indistinguishable, if $\mathrm{R}^{\mathcal{O}_0}$ is CEPT, so is $\mathrm{R}^{\mathcal{O}_1}$. (This step relies on CEPT and fails for EPT.)

Consequently, $\mathsf{Sim}(\mathcal{V}^*)$ is CEPT whenever $\langle \mathcal{P}, \mathcal{V}^* \rangle$ is CEPT, and $\mathsf{Sim}(\mathcal{V}^*)$ and $\langle \mathcal{P}, \mathcal{V}^* \rangle$ are computationally indistinguishable. Pictorially, the security proof worked as follows:

$$\mathsf{A}_0 \underset{e}{\xrightarrow{\equiv}} \widetilde{\mathsf{A}}_1 \underset{e}{\xrightarrow{\equiv}} \mathrm{R}^{\mathcal{O}_0} \overset{c}{\approx} \mathrm{R}^{\mathcal{O}_1} \underset{e}{\xrightarrow{\equiv}} \widetilde{\mathsf{A}}_1 \underset{e}{\xrightarrow{\equiv}} \mathsf{A}_1,$$

where $\mathsf{A} \underset{e}{\xrightarrow{\equiv}} \mathsf{B}$ denotes that A and B are perfectly indistinguishable and that if B is efficient (given $\mathcal{V}^*$), so is A. More precisely, we have

$$\langle \mathcal{P}, \cdot \rangle \underset{e}{\xrightarrow{\equiv}} \mathsf{RWS}(\cdot) \underset{e}{\xrightarrow{\equiv}} \mathrm{R}^{\mathcal{O}_0}(\cdot) \overset{c}{\approx} \mathrm{R}^{\mathcal{O}_1}(\cdot) \underset{e}{\xrightarrow{\equiv}} \widetilde{\mathsf{Sim}}(\cdot) \underset{e}{\xrightarrow{\equiv}} \mathsf{Sim}(\cdot),$$

where we made explicit, that this construction is functional in the adversary (the missing argument denoted "$\cdot$"). We also note that the intermediate steps ($\widetilde{\mathsf{A}}_0$, $\widetilde{\mathsf{A}}_1$, resp. RWS, $\widetilde{\mathsf{Sim}}$) can be omitted.

As a first step, we have define what a "reduction" is. Simple "reductions" are just connections of two interactive algorithms (which depend on $\mathcal{A}$) by an indistinguishability assumption. The name "reduction" is debatable, and the definition very restrictive, but sufficient for our purposes.

Definition 6.19 (Simple reductions). A **simple reduction** under an (implicit) simple assumption $(\mathcal{C}_0, \mathcal{C}_1)$ is an oracle algorithm R which expects expects access to an oracle $\mathcal{C}_b$ and $\mathsf{code}(\mathcal{A})$ as input. Given $\mathcal{O}$ and $\mathsf{code}(\mathcal{A})$, $\mathrm{R}^{\mathcal{O}_b}(\mathcal{A})$ implements (an interactive) algorithm.

Our definition of benign simulation requires a security proof as sketched in Example 6.18, and is basically an abstract formalization of that proof strategy. For completeness, we give a more traditional approach in Appendix F, which relies on indistinguishability of queries similar to [KL08]. We view both approaches as complementary: Our definition of benign simulation is *easily applicable* to typical protocols (and all of our examples), whereas the query-indistinguishability condition is something one can arguably expect from almost any simulator, which *broadens* the class of simulators which handle CEPT adversaries in CEPT. In any case, a bb-rw simulation with a normal rewinding strategy is assumed.

Definition 6.20 (Benign simulation). Let $(\mathcal{P}, \mathcal{V})$ be an argument system. Let Sim be a *(timed) bb-rw* simulator with **associated rewinding strategy** RWS and **associated simple reduction** R under simple assumption $(\mathcal{C}_0, \mathcal{C}_1)$. Moreover, the reduction $\mathrm{R}^{\mathcal{C}_b}(\mathcal{V}^*)$ has the interface of RWS, i.e. it expects $(\mathsf{code}(\mathcal{V}^*), x, w, \mathit{aux})$. Suppose that, for any adversary $\mathcal{V}^*$:

(1) RWS is a *normal* rewinding strategy.
(2) $\mathsf{RWS}^{\mathcal{V}^*} \equiv \mathrm{R}^{\mathcal{C}_0}(\mathcal{V}^*)$ and $\mathrm{R}^{\mathcal{C}_0}(\mathcal{V}^*)$ is efficient relative to $\mathsf{RWS}^{\mathcal{V}^*}$ with polynomial runtime tightness.
(3) $\mathrm{R}^{\mathcal{C}_1}(\mathcal{V}^*) \equiv \mathsf{Sim}(\mathcal{V}^*)$ and $\mathsf{Sim}(\mathcal{V}^*)$ is efficient relative to $\mathrm{R}^{\mathcal{C}_1}(\mathcal{V}^*)$ with polynomial runtime tightness.
(4) $\mathcal{C}_0$ and $\mathcal{C}_1$ form a simple assumption, and are indistinguishable, i.e. $\mathcal{C}_0 \overset{c}{\approx} \mathcal{C}_1$.

Then Sim is **benign** (under the assumption $\mathcal{C}_0 \overset{c}{\approx} \mathcal{C}_1$).

6.3.1. Iterated benign reductions

Our definition of benign allows only *one* "reduction step" using $\mathcal{C}_0 \overset{c}{\approx} \mathcal{C}_1$. Many security proofs can be squeezed into this setting. However, a simple relaxation is useful.

Definition 6.21 (Iterated benign). In the setting of Definition 6.20, we call Sim **iterated benign**, if there is a *constant* k and a sequence of "intermediate simulators" $\mathsf{Sim}_0, \ldots, \mathsf{Sim}_k$, which expect as input $(\mathsf{code}(\mathcal{A}), x, w, \mathit{aux})$ so that

(1) $\mathsf{Sim}_0 \equiv \langle \mathcal{P}, \cdot \rangle$ and $\mathsf{Sim}_k \equiv \mathsf{Sim}$ (where Sim_k ignores w).
(2) Sim_i and Sim_{i+1} are related by a benign reduction (as in Definition 6.20, with oracles $\mathcal{C}_{i,b}$, $i = 1, \ldots, k$, $b = 0, 1$.).

We stress that iterated benign only allows a *constant* number of "hops". The reason is that runtime may double for each hop, so superconstantly many "hops" *could* make the runtime explode. Thus, hybrid arguments must be put into the (simple) assumptions. Also note that RWS can be absorbed into R, and the relative efficiency requirement. While we the (possible) use of RWS explicit in Definition 6.20, we left it implicit in Definition 6.21.

6.3.2. Examples of (iterated) benign simulators

All of our examples can be easily expressed via (iterated) benign simulators. We stress that hybrid arguments must be incorporated into the (simple) assumptions $\mathcal{C}_{i,0} \overset{c}{\approx} \mathcal{C}_{i,1}$.

Example 6.22. The classic, the constant round, and the concurrent zero-knowledge protocol examples [GMW86; Blu86; GK96; Ros04; Lin13; KP01; PTV14] from Section 6.1.3 have benign simulation.

6.3.3. Zero-knowledge and benign simulation

We only give results for benign simulation. Extending these to iterated benign is straightforward and left to the reader.

Lemma 6.23. *Suppose $(\mathcal{P},\mathcal{V})$ is an argument system. Let* Sim *be a benign simulator. Then* Sim *is a zero-knowledge simulator which handles CEPT adversaries (in CEPT).*

Proof. Suppose $(\mathcal{I},\mathcal{V}^*,\mathcal{D})$ is an adversary which is CEPT in the real protocol. For brevity, whenever we call an (interactive) algorithm CEPT in the following, we mean that (if necessary) the inputs are generated by $\mathcal{I}$ (and $\mathcal{D}$ is applied to the output).

Suppose for simplicity that it halts with probability 1. Then the output of a normal rewinding strategy RWS is distributed like the real protocol output.[50] By normality, RWS is CEPT. By relative efficiency, $\mathrm{R}^{\mathcal{C}_0}(\mathcal{V}^*)$ is CEPT. Also, by assumption, the "reduction" $\mathrm{R}^{\mathcal{C}_0}(\mathcal{V}^*)$ behaves (as a system) exactly like RWS. By indistinguishability of $\mathcal{C}_0$ and $\mathcal{C}_1$, the standard reduction shows that $\mathrm{R}^{\mathcal{C}_1}(\mathcal{V}^*)$ is CEPT and the output of $\mathrm{R}^{\mathcal{C}_1}(\mathcal{V}^*)$ is (computationally) indistinguishable from $\mathrm{R}^{\mathcal{C}_0}(\mathcal{V}^*)$ (and hence the real protocol). By relative efficiency of Sim, $\mathsf{Sim}(\mathcal{V}^*)$ is CEPT (with environment $\mathcal{I},\mathcal{D}$). Since $\mathrm{R}^{\mathcal{C}_1}(\mathcal{V}^*)$ is behaves (as a system) exactly as $\mathsf{Sim}(\mathcal{V}^*)$, the output of $\mathsf{Sim}(\mathcal{V}^*)$ and $\langle\mathcal{P},\mathcal{V}^*\rangle$ is indistinguishable. Thus Sim handles CEPT adversaries in CEPT. □

By Lemma 6.23, all of our examples in Example 6.22 are not only secure against a priori PPT adversaries, but have CEPT simulation against *designated CEPT* adversaries.

Remark 6.24 (More precise runtime bounds). We saw for $\mathrm{G3C}_{\mathrm{GK}}$, that the runtime of the simulator Sim and the rewinding strategy RWS are *very closely related*. For this, we used "boxing" and "unboxing" (and that commitment computations only depend on message lengths). Such a close relation of runtime is typical, since in most security proofs only rewinding and bookkeeping introduces (significant) changes in the runtime. Hence, our extendability results are relatively *crude feasibility results*, assuring that zero-knowledge extends to CEPT adversaries.

6.4. Sequential zero-knowledge from benign simulation

To prove sequential that sequential zero-knowledge follows from auxiliary input zero-knowledge, we had to rely on the hybrid argument, which hides a lot of complexity. For benign simulation, it is easy to prove that it composes sequentially. Conceptually this follows from:

- Using that rewinding strategies "compose sequentially".
- Using that relative efficiency *with runtime tightness* "composes sequentially".
- Using that simple assumptions "compose sequentially", which is a very fancy way to say that we rely on "repeated trials".
- Hence, benign "composes sequentially".

Remark 6.25 (Lifting normality and relative efficiency). For brevity's sake, we do not explicitly lift rewinding strategies and relative efficiency to the sequential composition setting, i.e. we do not explicitly define what "composes sequentially" means in that setting. It is straightforward to define by using an (environmental) adversary and replacing access to the objects $\mathcal{O}_0$, $\mathcal{O}_1$ of interest (e.g. RWS and $\langle\mathcal{P},\cdot\rangle$ for normality) by repeated access, i.e. $\mathsf{rep}(\mathcal{O}_0)$, $\mathsf{rep}(\mathcal{O}_1)$. We note that the tightness parameters are unaffected (since the notions were already "perfect").

Lemma 6.26 (Sequential zero-knowledge from benign simulation). *Let $(\mathcal{P},\mathcal{V})$ be an argument system. Suppose* Sim *is a benign simulator (for auxiliary input zero-knowledge). Then $(\mathcal{P},\mathcal{V})$ is sequential zero-knowledge.*

[50] If $\langle\mathcal{P},\mathcal{V}^*\rangle_{\mathcal{I}}$ halts with probability $1-\nu$, then the output (including `nohalt`) has statistical distance at most $\mathsf{poly}_{\mathsf{virt}}\cdot\nu$. Since ν is bounded by the virtuality of $\langle\mathcal{P},\mathcal{V}^*\rangle_{\mathcal{I}}$ anyway, the rest of the proof works without change.

Proof sketch. Let $(\mathcal{E}, \mathcal{V}^*)$ be the adversary trying to distinguish $\mathcal{O}_{\mathscr{P}}$ and $\mathcal{O}_{\mathsf{Sim}}$. Let RWS be the normal rewinding strategy of Sim. Let R be reduction and C_0, C_1 be the simple assumption.

Step 1 (Sequential composition of RWS): Let $\mathsf{poly}_{\mathsf{time}}$ and $\mathsf{poly}_{\mathsf{virt}}$ be the runtime and probability tightness of RWS. Let $\mathcal{O}_{\mathscr{P}} \mathrel{\widehat{=}} \mathsf{rep}(\langle \mathscr{P}, \mathcal{V}^* \rangle)$ and let $\mathcal{O}_{\mathsf{RWS}} \mathrel{\widehat{=}} \mathsf{rep}(\mathsf{RWS}^{\mathcal{V}^*})$. Suppose for simplicity that $(\mathcal{E}, \mathcal{V}^*)$ always halts. Then we know that for *any input*, the state of $\mathcal{V}^*$ after RWS is identically distributed to the state after interaction with $\mathscr{P}$ (by normality). Hence, replacing $\mathcal{O}_{\mathscr{P}}$ with $\mathcal{O}_{\mathsf{RWS}}$ only affects the runtime. Now, we lift Lemma 6.5 to the sequential setting.

Define $T_{\mathsf{RWS},i}$ resp. $T_{\mathscr{P},i}$ as the time spent in the i-th invocation of $\mathcal{O}_{\mathsf{RWS}}$ resp. $\mathcal{O}_{\mathscr{P}}$. Note that

$$\begin{aligned}\mathbb{E}(\mathsf{time}_{\mathsf{RWS}+\mathcal{V}^*}(\langle \mathcal{E}, \mathcal{O}_{\mathsf{RWS}} \rangle)) &= \sum_i \mathbb{E}(T_{\mathsf{RWS},i}) \\ &\leq \mathsf{poly}_{\mathsf{time}} \cdot \sum_i \mathbb{E}(T_{\mathscr{P},i}) \\ &= \mathsf{poly}_{\mathsf{time}} \cdot \mathbb{E}(\mathsf{time}_{\mathscr{P}+\mathcal{V}^*}(\langle \mathcal{E}, \mathcal{O}_{\mathscr{P}} \rangle))\end{aligned}$$

where normality is applied for each i.

Suppose $(\mathcal{E}', \mathcal{V}')$ are `timeout`-modifications according to Lemma 3.12. By probability tightness, the probability that the i-th iteration of RWS runs into a `timeout` event is at most $\mathsf{poly}_{\mathsf{virt}}$-fold the probability for $\mathscr{P}$ to run into a `timeout` event. Consequently, the virtuality is increased by at most a factor of $\mathsf{poly}_{\mathsf{virt}}$.

All in all, we have shown that $\mathcal{O}_{\mathsf{RWS}}$ is a "sequential rewinding strategy" with runtime tightness $\mathsf{poly}_{\mathsf{time}}$, probability tightness $\mathsf{poly}_{\mathsf{virt}}$, and perfect output distribution;[51] and we lifted Lemma 6.5.

Step 2 (Relative efficiency composes sequentially): Let $\mathcal{O}_{\mathrm{R}^{C_b}} \mathrel{\widehat{=}} \mathsf{rep}(\mathrm{R}^{C_b}(\mathcal{V}^*))$ for $b \in \{0,1\}$, and let $\mathcal{O}_{\mathsf{Sim}} \mathrel{\widehat{=}} \mathsf{rep}(\mathsf{Sim}(\mathcal{V}^*))$. Suppose Sim is efficient relative to R^{C_1} with runtime tightness $(\mathsf{poly}_{\mathsf{time}}, \mathsf{poly}_{\mathsf{virt}})$. Then the oracle $\mathcal{O}_{\mathsf{Sim}}$ is efficient relative to $\mathcal{O}_{\mathrm{R}^{C_1}}$ with runtime tightness poly. Namely, for any $(\mathcal{E}, \mathcal{V}^*)$,

$$\begin{aligned}\mathbb{E}(\mathsf{time}_{\mathsf{Sim}+\mathcal{V}^*}(\langle \mathcal{E}, \mathcal{O}_{\mathsf{Sim}} \rangle)) &= \sum_i \mathbb{E}(T_{\mathsf{Sim},i}) \\ &\leq \mathsf{poly}_{\mathsf{time}} \sum_i \mathbb{E}(T_{\mathrm{R}^{C_1},i}) \\ &= \mathsf{poly}_{\mathsf{time}} \cdot \mathbb{E}(\mathsf{time}_{\mathrm{R}^{C_1}}(\langle \mathcal{E}, \mathcal{O}_{\mathscr{P}} \rangle))\end{aligned}$$

where $T_{\mathsf{Sim},i}$ resp. $T_{\mathrm{R}^{C_1}}$ denotes the time for the i-th invocation of the respective oracle. This again follows by comparing i-th invocations, and using that output distributions are identical by assumption. And as for RWS, we can lift the runtime guarantees to the sequential setting, including virtualities. That is, if the virtually expected time is (t, ε) with $\mathcal{O}_{\mathrm{R}^{C_1}}$, then it is $(\mathsf{poly}_{\mathsf{time}} \cdot t, \mathsf{poly}_{\mathsf{virt}} \cdot \varepsilon)$ with $\mathcal{O}_{\mathsf{Sim}}$. The same holds for $\mathcal{O}_{\mathsf{RWS}}$ and $\mathcal{O}_{\mathrm{R}^{C_0}}$.

Step 3 (Indistinguishability of $\mathcal{O}_{\mathrm{R}^{C_0}}$ and $\mathcal{O}_{\mathrm{R}^{C_1}}$): It is obvious that indistinguishability of $\mathcal{O}_{\mathrm{R}^{C_0}}$ and $\mathcal{O}_{\mathrm{R}^{C_1}}$ reduces to indistinguishability of C_0 and C_1 under repeated trials. (Each invocation of $\mathcal{O}_{\mathrm{R}^{C_0}}$ (resp. $\mathcal{O}_{\mathrm{R}^{C_1}}$) is another trial.) By Corollary 6.17, simple assumptions are indistinguishable under repeated trials. (It is vital that R^{C_0} is CEPT. That follows from Steps 1 and 2.)

Step 4 (Benign composes sequentially): From Steps 1 to 3, it follows immediately that benign "composes sequentially". More concretely, it follows that $(\mathcal{E}, \mathcal{V}^*)$ cannot distinguish $\mathcal{O}_{\mathscr{P}}$ and $\mathcal{O}_{\mathsf{Sim}}$, and in particular, an execution with $\mathcal{O}_{\mathsf{Sim}}$ is again CEPT. □

[51] If the probability for non-halting executions is not 0, the easiest way is to argue by truncating after say 2^κ steps and using statistical closeness to $(\mathcal{E}, \mathcal{V}^*)$. But a closer inspection shows that any $\mathsf{poly}_{\mathsf{time}}$ is fine since $\infty \leq \infty$. For $\mathsf{poly}_{\mathsf{virt}}$ a closer inspection shows that it does not change either. This is unsurprising since virtuality must at least remove non-halting executions anyway.

7. Sketched application to SFE

We very briefly recall security definitions for SFE, but assume basic familiarity with the topic. Again, we adopt a uniform complexity setting with universal simulation. As with zero-knowledge, we therefore need an "environmental" adversary.

7.1. Definitions

Let n be a constant in κ and consider n interacting parties. In the setting of SFE, n parties wish to jointly compute a (probabilistic) functionality $\mathsf{f}\colon (\{0,1\}^*)^n \to (\{0,1\}^*)^n$ implemented by the algorithm f. We demand that f is a priori PPT in κ. The parties input $x_1, \ldots, x_n$ and, at the end of the protocol, output $y_1, \ldots, y_n$. A **protocol** π consists of algorithms $(\pi_1, \ldots, \pi_n)$, such that each party i executes $\pi_i(x_i)$.[52] We assume the parties have *secure channels* for communication, i.e. an eavesdropping adversary only learns message lengths and communications proceeds in rounds.

We call a party **corrupted**, if it is controlled by the adversary. We restrict to **static corruption**, that is, for execution of a protocol π the subset $I \subseteq \{1, \ldots, n\}$ of the corrupted parties is fixed from the start (and does not (adaptively) grow).

To shorten our exposition, we start with the **hybrid model**, and treat the real model as a special case. In the f-hybrid model, we assume (repeated) access to an ideal functionality f. A protocol π may use f, and we sometimes write π^f to emphasize this. Let π^f be a protocol implementing a functionality g in the f-hybrid model. Let $\mathcal{A}$ be an adversary corrupting the set $I \subseteq \{1, \ldots, n\}$ of parties Let $\vec{x} = (x_1, \ldots, x_n)$, $\vec{r} = (r_1, \ldots, r_n, r_{\mathcal{A}}, r')$ denote the inputs and randomness of the parties. Here r' denotes the randomness used in ideal functionalities. By *aux* we denote the auxiliary input of $\mathcal{A}$. The adversary's input will be $\{x_i\}_{i\in I}$, *aux* and I. The computation of π proceeds in rounds. The parties can also query an instance of the functionality f. In the end, all parties return outputs y_i, and the adversary outputs $y_{\mathcal{A}}$; we write $\vec{y} = (y_1, \ldots, y_n, y_{\mathcal{A}})$ for all outputs.

We denote by $\mathsf{Hybrid}^f_{\pi,\mathcal{A}}(\kappa, \vec{x}, \mathit{aux}, I; \vec{r})$ the output $\vec{y}$ of the execution of the protocol π^f where adversary $\mathcal{A}$ controls the parties in I and the inputs to all parties is $\vec{x}$ (and randomness $\vec{r}$). Since the parties have access to f, we call this the f-**hybrid model**.

In the **real model**, $\mathsf{Real}_{\pi,\mathcal{A}}(\kappa, \vec{x}, \mathit{aux}, I; \vec{r})$, denotes the output of a real execution. This is defined as in the hybrid model, except that there is no hybrid functionality (i.e. f is the null-functionality).

We denote by $\mathsf{Ideal}_{g,\mathsf{Sim}}(\kappa, \{x_i\}_{i\in I}, \mathsf{code}(\mathcal{A}), \mathit{aux}, I; \vec{r})$ the output $\vec{y} = (y_1, \ldots, y_n, y_{\mathsf{Sim}})$ of an execution in the **ideal model** with functionality g and ideal adversary Sim, called (universal) simulator. Here, the honest parties hand their inputs to g and output what they receive from g; inputs for corrupted parties may be provided by Sim. The simulator is given $\{x_i\}_{i\in I}$, *aux*, I, and $\mathsf{code}(\mathcal{A})$ as input. (As usual, we often omit $\mathsf{code}(\mathcal{A})$ when it is clear from the context; it is only required for universal simulation.[53])

We extend the definition of Hybrid, Real and Ideal to adaptive sequential composition, where an "environment" $\mathcal{E}$ provides inputs to executions of Hybrid, Real or Ideal (which choose fresh randomness). We denote this by $\mathsf{Hybrid}^f_{\pi,\mathcal{A}}(\kappa, \mathcal{E})$, or $\mathsf{Real}_{\pi,\mathcal{A}}(\kappa, \mathcal{E})$, or $\mathsf{Ideal}_{g,\mathsf{Sim}}(\kappa, \mathcal{E})$. After each execution, $\mathcal{E}$ learns all outputs, and can (adaptively) choose inputs for further executions. More formally, $\mathcal{E}$ is given access to either $\mathcal{O}_{\pi,\mathcal{A}}$ or $\mathcal{O}_{g,\mathsf{Sim}}$, which take as inputs $(\vec{x}, \mathit{code}, \mathit{aux}, I)$ and output $\vec{y}$ (which includes $y_{\mathcal{A}}$), i.e. $\mathcal{O}_{\pi,\mathcal{A}}(\vec{x}, \mathit{aux}, I) = \mathsf{Real}_{\pi,\mathcal{A}}(\vec{x}, \mathit{aux}, I)$ and $\mathcal{O}_{g,\mathsf{Sim}}(\vec{x}, \mathit{aux}, I) = \mathsf{Ideal}_{g,\mathsf{Sim},I}(\vec{x}, \mathit{aux}, I)$. (We omitted *code*, since it is always $\mathsf{code}(\mathcal{A})$.) Note that $\mathcal{E}$ can adaptively choose the set I of corrupted parties as well.

Definition 7.1. Let π be a protocol for g in the f-hybrid model. We say an adversary $(\mathcal{E}, \mathcal{A})$ is $\mathcal{T}$-time (e.g. CEPT), if $\mathsf{time}(\mathsf{Hybrid}^f_{\pi,\mathcal{A}}(\kappa, \mathcal{E}))$ is $\mathcal{T}$-time, where $(\vec{y}, \mathit{aux}, \mathit{state}) \leftarrow \mathcal{I}$. (We stress that the time to compute f is included here.)

[52] Typically the π_i are a priori PPT, but as with zero-knowledge, we do not rely on this to specify and prove security.

[53] In our setting, universal and existential simulation coincide, just like for zero-knowledge.

The protocol π is said to **sequentially t-securely** compute g against $\mathcal{T}$-time adversaries, if there exists a (universal) ideal adversary Sim, for any $\mathcal{T}$-time adversary $(\mathcal{E}, \mathcal{A})$, where $\mathcal{E}$ only corrupts subset of size at most t, we have

- $\mathsf{Real}^f_{\pi,\mathcal{A}}(\kappa, \mathcal{E}) \overset{c}{\approx} \mathsf{Ideal}_{g,\mathsf{Sim}}(\kappa, \mathcal{E})$;
- $\mathsf{time}(\mathsf{Ideal}_{g,\mathsf{Sim}}(\kappa, \mathcal{E}))$ is $\mathcal{T}$-time (e.g. CEPT);

Auxiliary input t-security is defined by restricting $\mathcal{E}$ to one query only; in this case, we usually write $\mathcal{I}$ instead of $\mathcal{E}$.

Lemma 7.2. *Auxiliary input t-security and sequential t-security are equivalent.*

Proof sketch. This is a straightforward application of the hybrid lemma. □

Note that there is no hybrid functionality f in the ideal world. The simulator must provide the hybrid functionalities. In particular, it is essential that f itself is efficient. Also note that we require that Sim is universal in both $\mathcal{A}$ and I. This does not strengthen the security in our setting, but simplifies discussions.[54] Some more remarks are in order.

Remark 7.3. We note that neither our results nor our definitions require black-box simulation. This is unlike [KL08]. However, the overhead of non-black-box simulation seems to preclude its use — at least the technique of [Bar01].

Remark 7.4. The notion of benign simulation can be extended to simulators for SFE.

Remark 7.5 (Zero-knowledge as an ideal functionality). The ideal zero-knowledge functionality takes as input (x, w) and outputs x to the verifier if $(w, x) \in \mathcal{R}$, else $\perp$. The protocol $\mathrm{G3C}_{\mathrm{GK}}$ is not (proven) t-secure as an ideal zero-knowledge functionality, because it does not seem to be a proof of knowledge, i.e. the witness cannot be extracted. Lindell [Lin13] describes a 5-move protocol, which is a zero-knowledge proof of knowledge, hence realizes the ideal zero-knowledge functionality. Its simulator handles CEPT adversaries in CEPT. (The simulation is benign.)

Remark 7.6 (Proofs of knowledge). Communication efficient (zero-knowledge) proofs *of knowledge* often a superlinear overhead for extraction in the witness size. This can break compatibility with CEPT completely, for example if extraction has a quadratic overhead in the witness size, then fat-tailed input distributions lead to inefficient extraction. Again, the problem is *expected* size input, and can be mitigated to some extent by size-guarding.

7.2. Modular sequential composition

In the following, we denote substituting an (ideal) subprotocol f by a (real) subprotocol ρ in π^f as π^ρ. Similarly we write $\pi^{f_1,\dots,f_m}$ resp. $\pi^{\rho_1,\dots,\rho_m}$ for substituting in multiple protocols. We need to assume that π proceeds in rounds, making only one subprotocol call per round. Moreover, all (honest) parties always call the same subprotocol.[55]

We can now state our adaption of [KL08, Theorem 12]. Unlike [KL08, Theorem 12], we assume the protocols are secure against CEPT adversaries (in our sense).

Theorem 7.7. *Let f_1, ..., f_m and g be ideal n-party functionalities. Let π be an n-party protocol that t-securely computes g against CEPT adversaries in the $(f_1, \dots, f_m)$-hybrid model. Suppose that π makes no more than one call to an ideal functionality in each round, that is, the functionalities f_i are used strictly sequentially. Let $\rho_1, \dots, \rho_m$ be n-party protocols so that ρ_i t-securely compute f_i against CEPT adversaries (in the real model). Then $\pi^{\rho_1,\dots,\rho_m}$ t-securely computes g against CEPT adversaries (in the real model).*

[54] We require constant n, so separate simulators (for the constantly many $I \subseteq \{1, \dots, n\}$) can be merged into one. Due to the a posteriori efficiency setting, existential simulators are universal anyway, as for zero-knowledge.

[55] See [Can00] for details the restrictions imposed on π.

The proof of Theorem 7.7 is straightforward — it is essentially as in [Can00]. That is:

(1) We construct from $(\mathcal{I}, \mathcal{A})$ the obvious adversary $(\mathcal{E}_\rho, \mathcal{A}_\rho)$ against sequential t-security of ρ. By assumption, we can replace $\mathcal{A}_\rho$ with Sim_ρ (and ρ with f). The execution remains efficient and the output is indistinguishable, i.e.

$$\mathsf{Real}_{\rho,\mathcal{A}_\rho}(\kappa, \mathcal{E}_\rho) \overset{c}{\approx} \mathsf{Ideal}^f_{\rho,\mathsf{Sim}_\rho}(\kappa, \mathcal{E}_\rho)$$

Thus, π^ρ was effectively replaced by π^f.

(2) Now, we construct an adversary $(\mathcal{I}, \mathcal{A}_\pi)$ against t-security of π in the f-hybrid model. By assumption, we can replace $\mathcal{A}_\pi$ with Sim_π (and π with g). Again, the execution remains efficient and the output is indistinguishable, i.e.

$$\mathsf{Hybrid}^f_{\pi,\mathcal{A}_\pi}(\kappa, \mathcal{I}) \overset{c}{\approx} \mathsf{Ideal}_{g,\mathsf{Sim}_\pi}(\kappa, \mathcal{I})$$

This concludes the proof.

Note that we can avoid the requirement of [KL08], that the simulator for ρ is efficient *in any interaction*. Weak relative efficiency of simulation is sufficient. The hybrid lemma takes care of the all the hairy details.[56]

Now, we sketch the proof in more detail. For brevity's sake, we will not repeat that t-security *against CEPT adversaries* is considered in every statement of "X securely computes Y". We note that, by assumption on $\pi^{f_1,\ldots,f_m}$, the real protocol $\pi^{\rho_1,\ldots,\rho_m}$ does not interleave executions. That is, only one instance of a subprotocol ρ_i is executed at a time.

Proof sketch. First, we simplify the situation by considering only one hybrid functionality f and protocol ρ. Since m is a constant, it will be evident that the proof easily extends, e.g. by going through m hybrid models.

Before we continue, we clarify and revert an important notational difference: We used the notation $\mathsf{rep}(\mathcal{O})$ for *repeated* access to independent instance of $\mathcal{O}$. In SFE/MPC, it is common that π^ρ resp. π^g denotes that π has *repeated* access to independent instances of ρ resp. g.

As noted before Definition 7.1, we view security as oracle indistinguishability. As with zero-knowledge, the hybrid lemma shows that $\mathsf{rep}(\mathcal{O}_{\rho,\mathcal{A}_\rho})$ is weakly efficient relative to $\mathsf{rep}(\mathcal{O}_{f,\mathsf{Sim}_\rho})$ and $\mathsf{rep}(\mathcal{O}_{\rho,\mathcal{A}_\rho}) \overset{c}{\approx} \mathsf{rep}(\mathcal{O}_{f,\mathsf{Sim}_\rho})$. That is, the analogue of sequential t-security (for a single protocol ρ) holds.

We argue in games. **Game** G_0 is the real execution π^ρ, that is $\mathsf{Real}_{\pi,\mathcal{A}}(\kappa, \mathcal{I})$.

In **Game** G_1, we prepare to replace all instances ρ by f. For this, we interpret the calling "environment" of ρ as $\mathcal{E}_\rho$. That is, $\mathcal{E}_\rho$ executes π using access to $\mathsf{rep}(\mathcal{O}_{\rho,\mathcal{A}_\rho})$. The adversary $\mathcal{A}$ is now split and executed partially by $\mathcal{E}_\rho$ and $\mathcal{A}_\rho$. Here $\mathcal{E}_\rho$ simulates the everything (the game, honest parties and $\mathcal{A}$) outside the subprotocol calls to ρ, whereas $\mathcal{A}_\rho$ emulates $\mathcal{A}$ (only) the subprotocol calls (and receives the state of $\mathcal{A}$ via *aux*). Here, it is essential that the calls are sequential. The changes from G_0 to G_1 are conceptual. Everything is efficient if (and only if) it was efficient before and the output is identically distributed.

In **Game** G_2, we replace $\mathcal{O}_{\rho,\mathcal{A}_\rho}$ by $\mathcal{O}_{f,\mathsf{Sim}_\rho}$. As noted before, the hybrid lemma implies that

$$\mathsf{G}_1 \equiv \mathsf{Real}_{\rho,\mathcal{A}_\rho}(\mathcal{E}_\rho) \equiv \mathcal{E}_\rho^{\mathsf{rep}(\mathcal{O}_{\rho,\mathcal{A}_\rho})} \overset{c}{\approx} \mathcal{E}_\rho^{\mathsf{rep}(\mathcal{O}_{f,\mathsf{Sim}_\rho})} \equiv \mathsf{Ideal}_{f,\mathsf{Sim}_\rho}(\mathcal{E}_\rho) \equiv \mathsf{G}_2$$

and both executions are efficient. Thus, we have substituted ρ by f in π. Now, we are effectively in the f-hybrid model.

Game G_3 undoes the changes of G_1, i.e. we revert to $\mathcal{E} \mathrel{\widehat{=}} \mathcal{I}$ and π, but now, we have $\pi^{\mathsf{rep}(f)}$ instead of $\pi^{\mathsf{rep}(\rho)}$. We call the resulting "network" adversary $\mathcal{A}_\pi$. This change is conceptual and does not affect output or efficiency.

[56] It is unsurprising that the proof of the modular sequential composition theorem in [KL08], is more complex, since it effectively *is* the hybrid argument.

In **Game** G_4, we replace $\pi^{\mathsf{rep}(f)}$ (resp. $\mathscr{A}_\pi$) by g (resp. Sim_π). Since π t-securely computes g in the f-hybrid model, we find

$$\mathsf{G}_3 \equiv \mathsf{Hybrid}^{f}_{\pi,\mathscr{A}_\pi}(\mathscr{G}) \equiv \mathscr{E}_\pi^{\mathcal{O}_{\pi,\mathscr{A}_\pi}} \overset{c}{\approx} \mathscr{E}_\pi^{\mathcal{O}_{g,\mathsf{Sim}_\pi}} \equiv \mathsf{Ideal}_{g,\mathsf{Sim}_\pi}(\mathscr{E}_\pi) \equiv \mathsf{G}_4$$

and G_4 is efficient if G_4 is. Thus $\mathsf{Ideal}_{g,\mathsf{Sim}_\pi}(\mathscr{E}_\pi)$ CEPT.

The construction of the simulator Sim for π^ρ follows from the above. That is, Sim runs Sim_ρ for $\mathscr{A}_\rho$ in subprotocol calls to ρ and Sim_π for π.

□

8. Conclusion and open problems

At the example of zero-knowledge and a sketched application to SFE, we demonstrated that the notion of computationally expected polynomial time is a useful and viable alternative to EPT. We also gave a "philosophical" motivation why EPT should be enlarged to CEPT, namely distinguishing-closedness. However, we leave open many minor and major questions and directions.

Beyond negligible advantage. The most important question may well be the *(in)compatibility of CPPT/CEPT and superpolynomial hardness assumptions.* Concretely, consider a one-way function where we assume that no PPT adversary can invert with probability better than $O(2^{-\kappa/2})$. W.r.t. CPPT/CEPT, such assumptions cannot exist, since with probability $O(2^{-\kappa/4})$, a CPPT/CEPT adversary may brute-force a preimage.

It is a critical question, whether this is a fundamental problem, or just another technical artefact. If CPPT/CEPT is incompatible with subexponential hardness assumptions, then protocols which rely on such are very likely incompatible with CPPT

Quantifiability, tightness, constructivity. For a more quantifiable notion of security, we need to better tackle the question of *tightness* of reductions, simulations, etc. The interpretation and treatment of the virtuality error for a good notion of tightness is non-trivial. Moreover, constructivity of security reductions is an interesting and important question, as we used the existence of (in general not computable) polynomial bounds in many places. In Appendices E.3.1 and E.3.2, we explore these questions very briefly.

Efficiency artefacts. In several situations, expected polynomial size inputs and messages resulted in rather strong requirements and fickle behaviour. Size-guards (Appendix E.4.3) are a mitigation. A more natural alternative is to investigate the efficiency class of *expected time strict space (EPT/SPS)* machines.

More abstract questions. Our "general" treatment of runtime provides the central results only for algebra-tailed runtime classes. Indeed, we even lack a definition of well-behaved runtime classes, for which we can expect such results to hold. Such a definition and extensions, as well as incorporating different advantage classes, are open. This may also lead to insights regarding superpolynomial hardness and CEPT, or vice versa.

Acknowledgements. I am grateful to Alexander Koch and Jörn Müller-Quade for feedback on an entirely different approach on EPT, and to Dennis Hofheinz for essentially breaking said approach. I also extend my gratitude to the reviewers of CRYPTO'20/21, and to Akin Ünal and Marcel Tiepelt, whose suggestions helped to improve the overall presentation. Special thanks go to the reviewers of TCC'20 and Dakshita Khurana for great feedback, which eventually resulted in the addition of the hybrid lemma.

References

[Bar01] Boaz Barak. "How to Go Beyond the Black-Box Simulation Barrier". In: *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*. IEEE Computer Society, 2001, pp. 106–115. DOI: 10.1109/SFCS.2001.959885. URL: https://doi.org/10.1109/SFCS.2001.959885.

[Bel02] Mihir Bellare. "A Note on Negligible Functions". In: *J. Cryptology* 15.4 (2002), pp. 271–284.

[BG11] Mihir Bellare and Oded Goldreich. "On Probabilistic versus Deterministic Provers in the Definition of Proofs of Knowledge". In: *Studies in Complexity and Cryptography*. Vol. 6650. Lecture Notes in Computer Science. Springer, 2011, pp. 114–123.

[BL04] Boaz Barak and Yehuda Lindell. "Strict Polynomial-Time in Simulation and Extraction". In: *SIAM J. Comput.* 33.4 (2004), pp. 738–818.

[Blu81] Manuel Blum. "Coin Flipping by Telephone". In: *CRYPTO*. U. C. Santa Barbara, Dept. of Elec. and Computer Eng., ECE Report No 82-04, 1981, pp. 11–15.

[Blu86] Manuel Blum. "How to prove a theorem so no one else can claim it". In: *Proceedings of the International Congress of Mathematicians*. Vol. 1. 1986, p. 2.

[BS20] Dan Boneh and Victor Shoup. *A Graduate Course in Applied Cryptography*. Version 0.5. 2020. URL: https://toc.cryptobook.us/.

[BT06] Andrej Bogdanov and Luca Trevisan. "Average-Case Complexity". In: *Foundations and Trends in Theoretical Computer Science* 2.1 (2006). DOI: 10.1561/0400000004. URL: https://doi.org/10.1561/0400000004.

[Can00] Ran Canetti. "Security and Composition of Multiparty Cryptographic Protocols". In: *J. Cryptol.* 13.1 (2000), pp. 143–202.

[Cha+14] Siu-on Chan, Ilias Diakonikolas, Paul Valiant, and Gregory Valiant. "Optimal Algorithms for Testing Closeness of Discrete Distributions". In: *SODA*. SIAM, 2014, pp. 1193–1203.

[CLP15] Kai-Min Chung, Edward Lui, and Rafael Pass. "From Weak to Strong Zero-Knowledge and Applications". In: *TCC (1)*. Vol. 9014. Lecture Notes in Computer Science. Springer, 2015, pp. 66–92.

[DG12] Ning Ding and Dawu Gu. "On Constant-Round Precise Zero-Knowledge". In: *ICICS*. Vol. 7618. Lecture Notes in Computer Science. Springer, 2012, pp. 178–190.

[Dwo+03] Cynthia Dwork, Moni Naor, Omer Reingold, and Larry J. Stockmeyer. "Magic Functions". In: *J. ACM* 50.6 (2003), pp. 852–921. DOI: 10.1145/950620.950623. URL: https://doi.org/10.1145/950620.950623.

[Fei90] Uriel Feige. "Alternative models for zero-knowledge interactive proofs". PhD thesis. Weizmann Institute of Science, 1990.

[GK96] Oded Goldreich and Ariel Kahan. "How to Construct Constant-Round Zero-Knowledge Proof Systems for NP". In: *J. Cryptology* 9.3 (1996), pp. 167–190.

[GM98] Oded Goldreich and Bernd Meyer. "Computational Indistinguishability: Algorithms vs. Circuits". In: *Theor. Comput. Sci.* 191.1-2 (1998), pp. 215–218. DOI: 10.1016/S0304-3975(97)00162-X. URL: https://doi.org/10.1016/S0304-3975(97)00162-X.

[GMW86] Oded Goldreich, Silvio Micali, and Avi Wigderson. "Proofs that Yield Nothing But their Validity and a Methodology of Cryptographic Protocol Design (Extended Abstract)". In: *FOCS*. IEEE Computer Society, 1986, pp. 174–187.

[Gol01] Oded Goldreich. *The Foundations of Cryptography - Volume 1: Basic Techniques*. Cambridge University Press, 2001. ISBN: 0-521-79172-3. DOI: 10.1017/CBO9780511546891. URL: http://www.wisdom.weizmann.ac.il/\%7Eoded/foc-vol1.html.

[Gol10] Oded Goldreich. "On Expected Probabilistic Polynomial-Time Adversaries: A Suggestion for Restricted Definitions and Their Benefits". In: *J. Cryptology* 23.1 (2010), pp. 1–36.

[Gol11a] Oded Goldreich. "Average Case Complexity, Revisited". In: *Studies in Complexity and Cryptography*. Vol. 6650. Lecture Notes in Computer Science. Springer, 2011, pp. 422–450.

[Gol11b] Oded Goldreich. "Notes on Levin's Theory of Average-Case Complexity". In: *Studies in Complexity and Cryptography*. Vol. 6650. Lecture Notes in Computer Science. Springer, 2011, pp. 233–247.

[Gol93] Oded Goldreich. "A Uniform-Complexity Treatment of Encryption and Zero-Knowledge". In: *J. Cryptology* 6.1 (1993), pp. 21–53. DOI: 10.1007/BF02620230. URL: https://doi.org/10.1007/BF02620230.

[GS98] Oded Goldreich and Madhu Sudan. "Computational Indistinguishability: A Sample Hierarchy". In: *Proceedings of the 13th Annual IEEE Conference on Computational Complexity, Buffalo, New York, USA, June 15-18, 1998*. IEEE Computer Society, 1998, pp. 24–33. DOI: 10.1109/CCC.1998.694588. URL: https://doi.org/10.1109/CCC.1998.694588.

[GW10] Craig Gentry and Daniel Wichs. "Separating Succinct Non-Interactive Arguments From All Falsifiable Assumptions". In: *IACR Cryptol. ePrint Arch.* 2010 (2010), p. 610.

[HM98] Shai Halevi and Silvio Micali. "More on Proofs of Knowledge". In: *IACR Cryptol. ePrint Arch.* 1998 (1998), p. 15. URL: http://eprint.iacr.org/1998/015.

[HUM13] Dennis Hofheinz, Dominique Unruh, and Jörn Müller-Quade. "Polynomial Runtime and Composability". In: *J. Cryptology* 26.3 (2013), pp. 375–441. DOI: 10.1007/s00145-012-9127-4. URL: https://doi.org/10.1007/s00145-012-9127-4.

[KL08] Jonathan Katz and Yehuda Lindell. "Handling Expected Polynomial-Time Strategies in Simulation-Based Security Proofs". In: *J. Cryptology* 21.3 (2008), pp. 303–349.

[KM13] Neal Koblitz and Alfred Menezes. "Another look at non-uniformity". In: *Groups Complexity Cryptology* 5.2 (2013), pp. 117–139. DOI: 10.1515/gcc-2013-0008. URL: https://doi.org/10.1515/gcc-2013-0008.

[KP01] Joe Kilian and Erez Petrank. "Concurrent and resettable zero-knowledge in poly-loalgorithm rounds". In: *STOC*. ACM, 2001, pp. 560–569.

[Lev86] Leonid A. Levin. "Average Case Complete Problems". In: *SIAM J. Comput.* 15.1 (1986), pp. 285–286.

[Lin13] Yehuda Lindell. "A Note on Constant-Round Zero-Knowledge Proofs of Knowledge". In: *J. Cryptol.* 26.4 (2013), pp. 638–654.

[Lin17] Yehuda Lindell. "How to Simulate It - A Tutorial on the Simulation Proof Technique". In: *Tutorials on the Foundations of Cryptography*. Springer International Publishing, 2017, pp. 277–346.

[LM20] David Lanzenberger and Ueli Maurer. "Coupling of Random Systems". In: *IACR Cryptol. ePrint Arch.* 2020 (2020), p. 1187.

[Mey94] Bernd Meyer. "Constructive Separation of Classes of Indistinguishable Ensembles". In: *Proceedings of the Ninth Annual Structure in Complexity Theory Conference, Amsterdam, The Netherlands, June 28 - July 1, 1994*. IEEE Computer Society, 1994, pp. 198–204. DOI: 10.1109/SCT.1994.315804. URL: https://doi.org/10.1109/SCT.1994.315804.

[MP06] Silvio Micali and Rafael Pass. "Local zero knowledge". In: *STOC*. ACM, 2006, pp. 306–315.

[Nao03] Moni Naor. "On Cryptographic Assumptions and Challenges". In: *CRYPTO*. Vol. 2729. Lecture Notes in Computer Science. Springer, 2003, pp. 96–109.

[Pas06] Rafael Pass. "A precise computational approach to knowledge". PhD thesis. Massachusetts Institute of Technology, Cambridge, MA, USA, 2006.

[PTV14] Rafael Pass, Wei-Lung Dustin Tseng, and Muthuramakrishnan Venkitasubramaniam. "Concurrent Zero Knowledge, Revisited". In: *J. Cryptology* 27.1 (2014), pp. 45–66.

[Ros04] Alon Rosen. "A Note on Constant-Round Zero-Knowledge Proofs for NP". In: *TCC*. Vol. 2951. Lecture Notes in Computer Science. Springer, 2004, pp. 191–202.

A. Machine models

We do not want to go into much detail about the machine model, and will essentially assume that it is admissible. Admissibility carries certain explicit semi-formal requirements. As our machine model, we have some RAM-like model in mind. Indeed, "concrete efficiency" is relatively important when dealing with *expected* time. Recall that there are (runtime) distributions T over $\mathbb{N}_0$ with $\mathbb{E}(T) < \infty$ but $\mathbb{E}(T^2) = \infty$. Thus, we require that certain operations can be carried out efficiently (e.g. with logarithmic overhead). Importantly, we require efficient arithmetic and the abilitiy to use standard efficient construction, such as arrays or more sophisticated *data structures*, which allow efficient computation in a RAM model (or *multi-tape* Turing machine). We also require *efficient emulation* of (efficient) programs, oracles, or interactive systems in the sense that "emulating" an execution does not affect the runtime too much. Moreover, emulation allows to truncate, suspend, resume, rewind, or similarly affect executions based on efficiently computable events (such as the number of steps emulated, or messages received).

Remark A.1 (Non-Halting). Non-halting computations are an irksome technical artefact. To deal with them explicitly, we define the symbol `nohalt` as the "output" of such a computation, and assume that any system which receives `nohalt` also outputs `nohalt`, if not specified otherwise. Alternatively, one can follow [Gol10] and assume all algorithms halt after a finite number $n(\kappa)$ of steps. This introduces (arbitrarily) small deviations for "perfectness", e.g. it is again impossible to sample from $\mathsf{Ber}(1/3)$.

A.1. Systems, oracles, algorithms

Before considering machine models and specific properties, we sketch the high level abstractions. We view algorithms and oracles as systems, which offer (communication) interfaces. Interfaces allow to receive and/or send messages. For example, the input (resp. output) interface typically receives (resp. sends) exactly one message, the input (resp. output). To model "laziness", one may view the interface less strictly, and allow the input (resp. output) interface to read symbol for symbol. Thus, a calling algorithm need not provide the full input (resp. output) at once. This is relevant in our setting, where input (resp. output) lengths are not a priori bounded.

We do not formalize the means of interfacing precisely, but argue in a hand-wavy manner. (In our case, with many competing definitions of machine and communication models, we believe it is better to be explicitly imprecise, than importing a lot of unnecessary details.)

We work with three related notions: *Systems*, *oracles*, and *algorithms*. A **deterministic system** is defined by its interfaces and "input-output behaviour" only, i.e. it is a "mathematical object". A **(probabilistic) system** is a random variable S, such that any realization of S is a deterministic system.[57] A system has no notion of "runtime", or "random tape". By connecting interfaces, systems may interact. This forms a new system. Any system has an implicit input, the security parameter. A system is **closed** if the only input is its security parameter, and it offers only an output interface.

An **algorithm** is given by *code* (perhaps non-uniformly) and bound to a *machine model*. The code and machine model describe its behaviour as a system, and impart it with a notion of runtime and "random tape". (Randomness need not be modelled by a random tape.)

By **oracle** or **party**, we denote systems or algorithms to which only interface access is used. For example, black-box rewinding access (bb-rw) to an adversary means access to an oracle (with an underlying algorithm in this case). If not indicated otherwise, an oracle $\mathcal{O}$ is an algorithm (to which only interface access is provided).

In our setting, a convenient abstraction are **timed** oracles, which allow execution for an a priori *bounded time*, and which *report the elapsed time* to the caller when answering a query (or report

[57] Our definition of system is ad-hoc. A compatible, precise notion was recently (concurrently) introduced in [LM20]. We allow two probabilistic to behave identically, whereas in [LM20] equivalence classes are considered (and what we call system is called "probabilistic discrete system"). We prefer to work with concrete representatives, as having a concrete probability space at hand significantly simplifies definitions and reasoning, though it is not strictly necessary.

`timeout`, if it did not complete in time). See Appendix A.3 for a more precise specification. Timed bb-rw simulators can make use of this to truncate overlong executions, and this corresponds to *extended black-box access* in [KL08].

Another useful abstraction, mostly for convenience in the setting of a posteriori efficiency, are **timeful** oracles (or timeful systems). **Timeful** oracles are systems, which provide a *purported elapsed runtime* to the machine model. Importantly, timeful oracles are not bound by complexity notions or machine models, except satisfying consistency restrictions, e.g. their purported runtime must be long enough to have written the answer to the interface. Hardness assumptions, such as timelock puzzles are void against timeful oracles. Thus, they are a means to formalize unconditional runtime guarantees for algorithms with oracle-access, e.g. bb-rw simulators, but also serve as a convenient abstraction, e.g. for Lemma 3.12. A timeful oracle also yields a timed oracle in the obvious way.

A.2. Abstract machine model operations and interaction

From an abstract point of view, we want a machine model with following properties:[58]

Efficient arithmetic which does not thwart our results.
Efficient data structures such as arrays (i.e. random access), or something morally equivalent.
Abstract subroutines such as oracle calls, or a message sending function.
Abstract access to subroutine results. This is non-trivial, in particular if subroutines need not be efficient. Thus, even for a RAM-model, accessing the result of an oracle needs some tape-like access method.[59]
Interactive machines which communicate and are activated in some sensible way.
Efficient emulation ensures that one can efficiently execute code and emulate many interacting algorithms with little overhead.
A notion of runtime which is *local*, i.e. one can separate between time spent within some machine, subroutine, or oracles, and account accordingly.

Let us formalize our wishes a bit. Concerning arithmetic and data structures, we want typical algorithms to be efficient. In particular, distinguishing distributions by sampling often enough and computing the empirical distribution should be "efficient" in the sample size n, see Appendix C.4. For data structures, we may have to deal with excessively large inputs, thus, we may need suitable encodings, e.g. a tuple should allow access to any of its components efficiently, even with tape-like access. For example, representing (x, y) by concatenation only works if x is guaranteed to be short, but is inefficient if x is very long. Interleaving always works for tuples of constant dimension.

Now, we formalize the locality of runtime. Let $\mathsf{A}^{\mathcal{O}_1,\dots,\mathcal{O}_N}$ be an oracle machine (with access to N oracles). We require that

$$\mathsf{time}_{\mathsf{A}+\mathcal{O}_1+\mathcal{O}_2+\dots}(\mathsf{A}^{\mathcal{O}_1,\dots,\mathcal{O}_N}) = \mathsf{time}_{\mathsf{A}}(\mathsf{A}^{\mathcal{O}_1,\dots,\mathcal{O}_N}) + \mathsf{time}_{\mathcal{O}_1}(\mathsf{A}^{\mathcal{O}_1,\dots,\mathcal{O}_N}) + \mathsf{time}_{\mathcal{O}_2}(\mathsf{A}^{\mathcal{O}_1,\dots,\mathcal{O}_N}) + \dots \tag{A.1}$$

though "morally equivalent" relaxation suffice for most results. (Note that our algorithm takes no input. In case of randomized algorithms, the runtimes for A, $\mathcal{O}_1$, ...$\mathcal{O}_N$ are *not* stochastically independent.)

Finally, a sensible machine model guarantees efficient emulation. Namely, if $\mathsf{time}_{\mathsf{A}+\mathcal{O}_1+\mathcal{O}_2+\dots}(\mathsf{A}^{\mathcal{O}_1,\dots,\mathcal{O}_N})$ is efficient so is the runtime of the algorithm $\mathcal{B}$ which *emulates* the execution of all oracles. In other words, converting an (interacting) system of machines into a single machine $\mathcal{B}$ by emulating all parties (or oracles) *preserve efficiency*. Furthermore, emulation should efficiently allow to gather (and act upon)

[58] Another requirement, which is natural enough that we did not prominently require it, is that to send message of length n, some time is required. We assume n steps for length n as a lower bound.

[59] The problem here is: If the result of an oracle is *huge*, any access may exhaust the alloted runtime. This is nonsense (and completely breaks our results). For that reason, some (trivial, efficient) encoding for such unbounded objects are necessary, e.g. bitwise tape-like. Concretely, our runtime oracles might output gigantic runtimes, which a runtime distinguisher need not completely read to discern them from polynomial time.

execution statistics, most importantly the elapsed runtime of the emulated code, and the possibility to truncate an oracle emulation after a number of steps. Emulation should behave just like one expects from a virtual machine, in particular, be possible step-for-step.

Note that preservation of efficiency depends on the machine model and the notion of efficiency itself. For example, if emulation has a logarithmic overhead, then linear time is not preserved under emulation, but quasi-linear time may be. Emulation overhead which is linear (or better sublinear) in the number of emulated steps is a very convenient property of a machine model. We write $\mathsf{emuovhd}_{\kappa,N}(k)$ for the time steps required to emulate k steps (of a N machine/oracle system in some implicit machine model). Usually, the security parameter κ and number of oracles N are suppressed.

The communication model. We will assume an communication model where messages of arbitrary size can be sent, and parties have incoming messages queues. These do not count towards their space, and they do not pay runtime for receiving a message, only for reading it. Tape-like access to messages seems most natural, so we assume that. For technical reasons, one may wish provide the possibility of dropping (i.e. skipping) a (partially read) message. This allows a party to ignore large messages, keeping its runtime in check. Another possibility is to use fixed size messages (packages), and make the transfer of longer messages an "explicit" protocol. With this approach, our simplified view of "inputs as messages" is broken. This surfaces a technical detail, namely that reading from tapes and interacting with an interface which provides the same information is essentially the same, but technically different. By suitably restricting adversaries and algorithms, or introducing "unidirectional channels" (e.g. dummy transmitter parties) for passing inputs (after termination), this can be reconciled.

There are also different strategies for dealing with messages from super-constantly many parties, e.g. one tape-like message queue for all, one message queue per party, etc. Since our focus is (essentially) a two-party setting, we leave technical details, problems, solutions and their relations to the reader.

Non-uniformity. For non-uniform machines, we propose an advice interface just like the randomness interface.[60] That is, the advice string has infinite length. This seems to be the most natural choice from a machine-model perspective. Complexity classes can then restrict access further, e.g. to expected or strict polynomial size advice. Note that non-uniformity comes with its own more or less subtle anomalies, see e.g. [KM13].

A.3. Timed black-box emulation with rewinding access

We define *(timed) black-box emulation* similar to [KL08], which differs from standard black-box emulation essentially by making the "runtime/instruction counter" part of the visible black-box interface and by allowing runtime truncation.

Definition A.2 (Timed black-box emulation with rewinding access (bb-rw)). A **black-box emulation** oracle $\mathcal{O}$ gives oracle access to a "virtual machine" running some (once and for all) specified program/code. The code may involve multiple (abstracted) parties. Unless otherwise specified, $\mathcal{O}$ behaves *deterministically* in the sense that the randomness of the emulated programs is sampled and fixed prior to interaction.[61] We do *not* let the caller choose the randomness.

The black-box interface depends on the specific type.

- **Fully** black-box emulators take an input message m and return their program's answer a.

[60] The advice interface should follow the same restrictions as the "random tape' (see Remark A.6), in particular it should not provide memory to not conflate advice complexity with space complexity.

[61] When such an oracle is implemented, the "random tape" (or the respective notion in the machine model) is sampled (and fixed) lazily, just like a random oracle.

- **Timed** black-box emulators take a pair (m, t), where t is a maximum time bound, and return a pair (a, s), where s is the number of steps emulated. If s would exceed the alloted time t, the emulation is aborted and `timeout` is returned. A time bound of $t = \infty$ is allowed. (Execution may be resumed after `timeout`.)
- Black-box emulation **with rewinding access** (bb-rw) allows the state of the emulated program to be stored and loaded. While, a state is identified by its partial transcript of (previous) queries, other means of identification, such as handles, are used to ensure efficiency. Loading, storing, and deleting program states is done by special types of messages.[62]

Note that we distinguished black-box oracles with rewinding access from "normal" oracles. The reason is that the "next-message" approach usually used to implement black-box access is not efficient enough with expected time.

Example A.3 (Runtime squaring for NextMsg). Consider following interaction $\langle \mathsf{A}(n), \mathsf{B} \rangle$: First A sends n to B. Then A pings B n times, each times B returns a secret, which A uses in the next ping. Obviously, this interaction runs in time $O(n)$. Consider a distributions N of inputs n on $\mathbb{N}$ with the property that $\mathbb{E}(N) < \infty$ but $\mathbb{E}(N^2) = \infty$. Then emulation with next-message-function NextMsg is not efficient. The reason is that NextMsg always (re)computes from scratch, which needs about $\sum_{i=1}^{n} i \approx \frac{1}{2}n^2$ steps.

Fortunately, most problems like this arise from repeated computations (or repeated copying) being expensive, and are solved by making recomputing (or copying) superfluous. Computations can be cached, as done in bb-rw implementations. Copying can be reduced by sharing memory access, or passing around access to a machine or interface which implements such a shared memory access.

Remark A.4 (Cached UID NextMsg access). Caching (all) visited states and using short unique identifiers (UID) for visited states (instead of resending the history of messages leading to a state), yields a NextMsg-like function which is a suitable bb-rw oracle implementation (in all situations we have tried). Cached state and short UIDs prevent the quadratic computational overhead, but require *expected* polynomial space. Judiciously caching only important states is typically possible, so that usually strict polynomial space solutions exist.

Keeping track of identifiers and the rewinding tree can be done with efficient data structures. (Polylogarithmic overhead is admissible by Corollary A.8.)

Remark A.5. For admissible models, emulation of algorithms allows (efficient) runtime cutoffs. Cloning a machine's state, and resuming from a given state should also be (efficiently) possible. (Or we may add it as an new assumption.)

Remark A.6 (Space overhead). We have only considered *time* overhead of emulation. This is justified, as it bounds the space/memory overhead. However, memory overhead is an interesting quantity on its own. For example, one might argue that *expected poly-time (EPT)*, but *strict poly-space (SPS)*, is a "more natural" class of feasible computation than expected poly-time and expected poly-space.[63]

While SPS seems to prevent many technical artefacts, it unveils certain others. Depending on the implementation of the randomness interface (e.g. input, read-only tape, coin-toss, ...) emulation and bb-rw oracle implementations may not be SPS, because space and randomness complexity are mixed. If read-only access to an (infinite) random tape is given, then emulating two such tapes by "splitting" one works well. If randomness is a coin-toss interface, which upon invocation returns a fresh random bit, then emulation still works. However, to implement a bb-rw oracle $\mathsf{bbrw}(\mathcal{O})$, which gives access to $\mathcal{O}$ with *fixed* randomness, requires to remember all used randomness. This can require expected polynomial space.

How this can be resolved elegantly is an interesting question. One could rely on derandomization, e.g. with an (a priori PPT) pseudorandom function, to simulate a long enough random string with small

[62] Note that all of the code and interfaces which are in our control, e.g. the interface of the black-box are assumed to be nice and well-typed.

[63] Of course, the actual complexity class of interest allows EPT-SPS violation with negligible probability.

space. Alternatively, one could try to work with probabilistic bb-rw oracles, which, when rewound to a state use fresh randomness for new queries, i.e. the same query may yield different answers. Our problem with deterministic versus probabilistic access seems related to [BG11].

Similar to randomness, non-uniform (infinite) advice may need to be saved by a bb-rw implementation, leading to space overhead. Again, it depends on the concrete modelling.

A.4. (Probably) Admissible machine models

To the best of our knowledge, RAM models, and also multi-tape Turing machines, are admissible if one works with polynomial time or larger runtime classes.[64] Following trivial lemma is useful to see that efficient emulation is not hard to achieve, even for *expected* time.

Lemma A.7. *Let $f\colon \mathbb{N}_0 \to \mathbb{N}_0$ be any (monotone) strictly increasing function with (monotone) increasing left-inverse g, i.e. $g \circ f = \mathrm{id}$ (but not necessarily $f \circ g = \mathrm{id}$). Suppose T is a runtime and smaller than f, i.e. $\mathbb{P}(T_\kappa > f(\kappa)) = 0$ (for all κ). Let h be another monotone function. Then $\mathbb{E}(h(g(T_\kappa))T_\kappa) \leq h(\kappa)\mathbb{E}(T_\kappa)$.*

Proof. Use $h(g(T_\kappa)) \leq h(g(f(\kappa))) \leq h(\kappa)$. □

Corollary A.8. *Let* poly *be any monotone polynomial, and $\mathbb{E}(T_\kappa) \leq t(\kappa)$ for a polynomial bound t, and $T \leq 2^\kappa$. Then $\mathbb{E}(\mathsf{poly}(\log(T_\kappa))T_\kappa)$ is polynomially bounded (namely by $\leq \mathsf{poly}(\kappa)t(\kappa)$).*

Proof. Use Lemma A.7 with $f(\kappa) = 2^\kappa$, $g(\kappa) = \log_2(\kappa)$. and $h = \mathsf{poly}$, □

Note that $T_\kappa \leq 2^\kappa$ is easily achieved via a runtime cutoff after 2^κ steps.[65] This induces a statistically negligible change in the output of any expected polynomial time algorithm.[66] Thus, we see that polylogarithmic multiplicative overhead in emulation is not a problem for expected polynomial time computations. By taking a smaller superpolynomial bound, e.g. $f(\kappa) = \kappa^{\log(\kappa)}$, we get we a bit more freedom in the emulation overhead.

Remark A.9 (Interaction of Corollary A.8 and virtuality). CEPT and CPPT ignore negligible events, because they can be hidden in the virtuality. So, Corollary A.8 may always be applied after conditioning on the event $\{T_\kappa \leq 2^\kappa\}$, i.e. after using the "virtuality slack". Consequently, polylog overhead is not a problem for CEPT.

We end our discussion of machine models by taking a closer look two exemplary machine models.

Example A.10 (Single-tape Turing machines are not admissible). Consider single-tape Turing machine as the model of computation. It is easy to construct an *interactive* algorithm for computing whether a string is a palindrome which runs in *linear* time (in the length of the input string). However, it is well-known that single-tape Turing machines need quadratic time to recognize this language. Thus, the emulation overhead is (at least) quadratic. Hence, it is single-tape Turing machines are not an admissible model of computation.

Example A.11 (RAM models). Various RAM models seem appropriate for our cause. A model of computation in which the RAM's word size grows with the "problem size" seems particularly well-suited for cryptography; indeed security parameter κ is a natural measure for the "problem size".

B. Supplementary definitions

This section contains supplementary definitions which are commonplace (in many variations).

[64] We have not carried out formal proofs.
[65] Technically, we have to do an earlier cutoff, since emulation and cutoff also consume runtime. But this is a minor issue.
[66] Unfortunately, such a truncation can affect *perfect* properties, such as perfect correctness, leading to technical artefacts.

B.1. Commitment schemes

A commitment scheme allows a committer to commit to some value. The receiver does not learn that value until it is unveiled (the commitment is opened). Moreover, the commitment can be opened to at most one value, ensuring that the committer cannot change the value.

Formally, a commitment scheme is a two-phase protocol. For simplicity, we assume non-interactive commitments. Moreover, our commitment schemes consist of *a priori* PPT algorithms and have message space $\mathcal{M}_\kappa = \{0,1\}^\kappa$.

Definition B.1. A **(non-interactive) commitment scheme** Com **(with setup)** with message space $\mathcal{M}_\kappa = \{0,1\}^\kappa$ consists of following a priori PPT algorithms.

- $\mathsf{Gen}(\kappa; r)$ returns a commitment key ck.
- $\mathsf{VfyCK}(\mathrm{ck})$ verifies well-formedness of ck and accepts or rejects.
- $\mathsf{Com}(\mathrm{ck}, m; r)$ returns a pair (c, d) of commitment and decommitment for message m and randomness r.
- $\mathsf{VfyOpen}(\mathrm{ck}, c, m, d)$ accepts or rejects an opening of a commitment c to message m and decommitment d.

A commitment scheme must be perfectly correct, that is $\forall \mathrm{ck} \leftarrow \mathsf{Gen}(\kappa)\colon \mathsf{VfyCK}(\mathrm{ck}) = \mathtt{acc}$ and $\forall \mathrm{ck} \leftarrow \mathsf{Gen}(\kappa),\ m \in \mathcal{M}_\kappa,\ (c,d) \leftarrow \mathsf{Com}(\mathrm{ck})\colon \mathsf{VfyOpen}(\mathrm{ck}, c, m, d) = \mathtt{acc}$.

In the following, let $\mathcal{T} \in \{\mathcal{PPT}, \mathcal{EPT}, \mathcal{CPPT}, \mathcal{CEPT}\}$. (The results and definition can be adapted to any suitable, e.g. algebra-tailed runtime class.)

Definition B.2 (Binding). Let Com be a (non-interactive) commitment scheme and let $\mathcal{A}$ be an adversary in following game $\mathsf{Bind}_{\mathsf{Com},\mathcal{A}}$.

- Run $\mathrm{ck} \leftarrow \mathsf{Gen}(\kappa)$. The adversary returns $(c, m_0, d_0, m_1, d_1) \leftarrow \mathcal{A}(\kappa, \mathrm{ck})$.
- Return $\mathtt{win}$ iff $\mathsf{VfyOpen}(\mathrm{ck}, c, m_b, d_b) = \mathtt{acc}$ for $b = 0, 1$ and $m_0 \neq m_1$.

Let $\mathrm{Adv}^{\mathrm{bind}}_{\mathsf{Com},\mathcal{A}}(\kappa) = \mathbb{P}(\mathsf{Bind}_{\mathsf{Com},\mathcal{A}}(\kappa) = \mathtt{win})$. Then Com is computationally (resp. statistically, resp. perfectly) **binding** for $\mathcal{T}$-time (resp. -time-query, resp. unbounded) adversaries, if for any such adversary $\mathcal{A}$ we have $\mathrm{Adv}^{\mathrm{bind}}_{\mathsf{Com},\mathcal{A}}(\kappa) \leq \mathsf{negl}$ (resp. "$\leq \mathsf{negl}$", resp. "$= 0$").

By $\mathcal{T}$-time adversary, we mean the total time of game is $\mathcal{T}$-time. Since the commitment scheme's algorithms are a priori PPT time, efficiency of the game boils down to efficiency of $\mathcal{A}$.

Definition B.3 (Multi-Hiding LR-version). Let Com be a (non-interactive) commitment scheme and let $\mathcal{A}$ be an adversary in following game $\mathsf{Hide}_{\mathsf{Com},\mathcal{A}}$.

- Run $(\mathrm{ck}, \mathit{state}) \leftarrow \mathcal{A}(\kappa)$.
- If $\mathsf{VfyCK}(\mathrm{ck}) = \mathtt{rej}$, return $\mathtt{lose}$. Else run $b' \leftarrow \mathcal{A}^{\mathcal{O}_b}(\mathit{state}, \mathrm{ck})$, where $\mathcal{O}_b(m_0, m_1)$ checks if $m_0, m_1 \in \mathcal{M}_\kappa$ and[67] returns $c_b = \mathsf{Com}(\mathrm{ck}, m_b)$. Note that the adversary may repeatedly query $\mathcal{O}_b$.
- Return $\mathtt{win}$ if $b = b'$ else $\mathtt{lose}$

Let $\mathrm{Adv}^{\mathrm{hide}}_{\mathsf{Com},\mathcal{A}}(\kappa) = |2\,\mathbb{P}(\mathsf{Bind}_{\mathsf{Com},\mathcal{A}}(\kappa) = \mathtt{win}) - 1|$. Then Com is computationally (resp. statistically, resp. perfectly) **hiding** for any $\mathcal{T}$-time (resp. -time-query, resp. unbounded) adversary, if for any such $\mathcal{A}$ we have $\mathrm{Adv}^{\mathrm{bind}}_{\mathsf{Com},\mathcal{A}}(\kappa) \leq \mathsf{negl}$ (resp. "$\leq \mathsf{negl}$", resp. "$= 0$").

Finally, we note that CEPT adversaries are "no better" than a priori PPT adversaries.

Lemma B.4. *Suppose* Com *is computationally (resp. statistically, resp. perfectly) hiding (resp. binding) against a priori PPT adversaries. Then it also is against CEPT adversaries.*

[67] The message length is always κ in this case.

Proof sketch. Use that Com consists of a priori PPT algorithms and a standard truncation to a priori PPT to obtain an adversary $\mathcal{A}'$ with advantage at least half the advantage of $\mathcal{A}$ infinitely often. □

Remark B.5. The graph 3-colouring protocol $\mathrm{G3C}_{\mathrm{GK}}$ of Goldreich and Kahan [GK96] relies on a weaker "a posteriori hiding" property for the statistically hiding commitment scheme. Here, VfyCK may depend on secrets, e.g. the randomness of Gen, allowing more candidates schemes. The verification secrets are only revealed after the binding property is not needed anymore.

Concretely, in [GK96], the verifier commits to challenges, which must be statistically hidden during the protocol. However, it suffices that the verifier is ensured of this statistical hiding property at the end of the protocol. Thus, the change to VfyCK is possible there.

C. ⋆ Technical lemmata

In this section, we gather some lemmata for various purposes. Appendix C.2 contain some simple facts on statistical distance. In Appendix C.3, some cryptographic results concerning distinguishing and general hybrid arguments are given. And Appendix C.4 contains naive closeness tests.

C.1. Tail bounds

Tail bounds for distributions are the core tool for (runtime) cutoffs. For example, they allow to estimate how much the adversarial advantage suffers if we truncate.

Definition C.1 (Tail bounds). Let X be some distribution on $\mathbb{R}_{\geq 0}$. We call a (right-)continuous decreasing function $\mathsf{tail}\colon \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ a **tail bound of** X if $\forall x \in \mathbb{R}_{\geq 0}\colon \mathbb{P}(X > n) \leq \mathsf{tail}(n)$.

Moreover, we write $\mathsf{tail}^{\dagger}\colon \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0} \cup \{\infty\}$ for $\mathsf{tail}^{\dagger}(\alpha) = \inf\{n \mid \mathsf{tail}(n) \leq \alpha\}$, which satisfies $\mathsf{tail}(\mathsf{tail}^{\dagger}(\alpha)) \leq \alpha$. More generally, we call an upper bound bnd of some sequence $(x_n)_n$ a tail bound, i.e. $x_n \leq \mathsf{bnd}(n)$ for all n.

Tail bounds generalize to distributions over $\mathbb{R}_{\geq 0} \cup \{\infty, \mathtt{timeout}\}$, etc.

The optimal tail bound is $\mathsf{tail}(n) = 1 - \mathsf{CDF}_X(n)$, where CDF_X is the cumulative distribution function of X. We use $\mathsf{tail}^{\dagger}(\alpha)$ to conveniently denote the minimal n_α with $\mathsf{tail}(n_\alpha) \leq \alpha$, which exists due to continuity of tail.

For *strict* runtimes, e.g. strict polynomial time, the time bound is an admissible tail bound. More generally, we recall following lemma:

Lemma C.2 (Markov bound). *Let X be a distribution on $\mathbb{R}_0$ and suppose $\mathbb{E}(X) \leq t$. Then $\mathsf{tail}(n) = \frac{1}{n}t$ is an admissible tail bound and $\mathsf{tail}^{\dagger}(\alpha) = \frac{1}{\alpha}t$. For $\|X\|_p = (\mathbb{E}(X^p))^{1/p} \leq t$ with $p \geq 1$, we have $\mathsf{tail}(n) = (\frac{t}{n})^p$, and hence $\mathsf{tail}(n) \leq \frac{t}{n}$ if $n \geq t$.*

For simple corollaries concerning runtime truncation and bounds, see Appendix C.3.

C.2. Simple facts

In this section, we state some simple facts. Most are used with, or about, random variables, conditional variables, and the behaviour of statistical distance.

C.2.1. Statistical distance

Following lemma is useful to bound statistical distances of products of densities.

Lemma C.3. *Let $p_i, q_i \in [0, 1]$ for $i = 1, \ldots, n$. Then*

$$\left|\prod_{i=1}^{n} p_i - \prod_{i=1}^{n} q_i\right| \leq \sum_{i=1}^{n} |p_i - q_i|.$$

In particular, $\Delta(X \times Y, X' \times Y') \leq \Delta(X, Y) + \Delta(X', Y')$ *for random variables* X, Y, X', Y' *(not necessarily independent).*

More precisely, let $p_{(1,\dots,k)} \coloneqq \prod_{i=1}^{k} p_i$*, and let* $q_{(k,\dots)} \coloneqq \prod_{i=k}^{n} q_i$*, and let* $\delta_i \coloneqq |p_i - q_i|$ *then*

$$\left|\prod_{i=1}^{n} p_i - \prod_{i=1}^{n} q_i\right| \leq \sum_{i=1}^{n} p_{(1,\dots,i-1)} \delta_i q_{(i+1,\dots)}.$$

Assuming the products are finite, this continues to hold for $n = \infty$*.*

Proof. This follows from a straightforward induction (using $|p_i|, |q_i| \leq 1$) to simplify. The claim regarding statistical distance follows by an application of the inequality under the integral. □

Next, we note how conditional distributions and statistical distance are connected.

Remark C.4. Let X be random variable and let Y independently distributed like X conditioned on some event of probability ε. Then $\Delta(X, Y) = \varepsilon$.

(This follows easily since Y has the density $\mathbb{P}(\mathcal{E})^{-1} \mathbb{1}_{\mathcal{E}}$ as density w.r.t. X, where $\mathbb{1}_{\mathcal{E}}$ hence $2\,\Delta(X, Y) = \|\mathbb{1} - \mathbb{P}(\mathcal{E})^{-1} \mathbb{1}_{\mathcal{E}}\|_1 = \mathbb{P}(\mathcal{E}) + \mathbb{P}(\mathcal{E}) = 2\varepsilon$.)

Following is a simple result of CDFs.

Corollary C.5. *Let* X *and* Y *be two random variables over* $\mathbb{N}_0 \cup \{\infty\}$ *and let* $N \in \mathbb{N}_0$*. Suppose* X *(resp.* Y*) are truncated to* $X^{\leq N}$ *(resp.* $Y^{\leq N}$*) (i.e. they output* `timeout` *if they exceed* N*). Then*

$$\Delta(X, Y) - \mathbb{P}(X > N) \leq \Delta(X^{\leq N}, Y^{\leq N}) \leq \Delta(X, Y).$$

Proof. We show $\Delta(X, Y) - \Delta(X^{\leq N}, Y^{\leq N}) \geq \mathbb{P}(X > N)$. The left-hand side is $\sum_{k=n}^{\infty} |p_X(k) - p_Y(k)| - |\sum_{k=n}^{\infty} p_X(k) - p_Y(k)|$. This can be interpreted as ℓ_1-norms and the claim follows by general inequalities, see Lemma C.6. □

Lemma C.6. *Let* x, y *be be two elements in a normed vector space and suppose* $\|y\| \leq \varepsilon$*. Then*

$$|\|x - y\| - |\|x\| - \|y\|\|| \leq 2\|y\| \leq 2\varepsilon$$

The inequality is tight ($y = -x$*).*

Proof. We consider two cases. Suppose $\|x\| \leq \|y\|$. Then we find

$$\|x - y\| - |\|x\| - \|y\|| = \|x - y\| - \|x\| + \|y\| \leq \|x - y - x\| + \|y\| = 2\|y\|.$$

For the case $\|y\| \leq \|x\|$ we find by symmetry (of $|a - b|$) that

$$\|y - x\| - |\|y\| - \|x\|| \leq 2\|y\| \leq 2\varepsilon.$$

This finishes the proof. □

C.2.2. Domination (with slack)

We give some simple properties of domination (with slack).

Lemma C.7 (Properties of domination). *Let* $\mathbb{R}' = \mathbb{R} \cup \{\infty, \texttt{timeout}\}$*. Let* $X, Y : \Omega \to \mathbb{R}'$ *be random variables and suppose* $X \overset{d}{\leq}_L Y$ *for* $L \geq 1$*. Then:*

(1) For any monotonely increasing continuous function $f : \mathbb{R}' \to \mathbb{R}'$*, we have* $f(X) \overset{d}{\leq}_L f(Y)$*.*

(2) In particular, for $X, Y : \Omega \to \mathbb{R}'$*, we have* $X^{\leq t} \overset{d}{\leq}_L Y^{\leq t}$*.*

(3) *For any* $\nu \in [0,1]$, *we have* $X^\nu \overset{d}{\leq_L} Y^\nu$. *Moreover, even conditioned on* `¬timeout`, *the respective* X', Y' *satisfy* $X' \overset{d}{\leq_L} Y'$

(4) *If* $X, Y \geq 0$, $\|X\|_p \leq L\|Y\|_p$ *for* $p \in [1,\infty]$.

(5) *For* $i = 1,\dots,n$ *let* $X_i, Y_i \colon \Omega \to \mathbb{R}'$ *with* $X_i \overset{d}{\leq_L} Y_i$ *and* $\lambda_i > 1$ *with* $\sum_{i=1}^n \lambda_i = 1$. *Then* $\sum_{i=1}^n X_i \overset{d}{\leq_M} \sum_{i=1}^n \lambda_i^{-1} Y_i$. *In particular,* $\sum_{i=1}^n X_i \overset{d}{\leq_L} n \sum_{i=1}^n Y_i$.

Proof. Let f be as claimed. Let g_+ be defined as $g_+(y) = \inf f^{-1}(\{z \mid y \leq z\})$ and let g_- be defined as $g_+(y) = \sup f^{-1}(\{z \mid z \leq y\})$) Then $g_-(f(x)) \leq x \leq g_+(f(x))$ by definition, and $g_\pm$ are monotone. Since f is right-continuous, $f(x) \leq y \iff x \leq g_+(y)$. Consequently, for all $c \in \mathbb{R}'$

$$\mathbb{P}(f(X) > c) = 1 - \mathbb{P}(f(X) \leq c) = 1 - \mathbb{P}(X \leq g_+(c)) = \mathbb{P}(X > g_+(c)).$$

Now, we find for all $c \in \mathbb{R}'$

$$\mathbb{P}(f(X) > c) = \mathbb{P}(X > g_+(c)) \leq L\,\mathbb{P}(Y > g_+(c)) = L\,\mathbb{P}(f(Y) > c),$$

which proves the first claim. The second follows immediately by setting f appropriately. The last claim, directly follows in simple cases (namely if quantile-truncation coincides with truncation at some c). In general, we use that

$$\mathbb{P}(X > c) \leq L\,\mathbb{P}(Y > c) \iff \mathbb{P}(Y \leq c) \leq \frac{L-1}{L}\,\mathbb{P}(X > c)$$

and the definition of quantile cutoff

$$\mathbb{P}(X^\nu \leq c) = \max\{1, \mathbb{P}(X \leq c)\}.$$

Conditioning on `¬timeout` simply means using $\max\{1, \frac{1}{1-\nu}\,\mathbb{P}(X \leq c)\}$ as the new CDF. In both cases, the claim easily follows.

For the norm inequality, let $F = \mathsf{CDF}_X(\,\cdot\,)$, $G = \mathsf{CDF}_Y(\,\cdot\,)$ and note that $1 - F \leq L(1-G)$ by assumption. Also recall that $\|X\|_1 = \mathbb{E}(X) = \int_0^\infty 1 - F(x)\mathrm{d}x$ for any distribution $X \geq 0$. We assume $p < \infty$, and leave $p = \infty$ to the reader. Thus, $\|X\|_p^p = \|X^p\|_1^p = \int_0^\infty 1 - F(x^{1/p})\mathrm{d}x$. Finally $\int_0^\infty 1 - F(x^{1/p})\mathrm{d}x \leq \int_0^\infty L(1 - G(x^{1/p}))\mathrm{d}x = L^p\|Y\|_p^p$.

For item (5), note that $\sum_i X_i > t$ implies that there exists some i such that $X_i > \lambda_i t$. Thus

$$\mathbb{P}(\sum_i X_i > t) \leq \sum_i \mathbb{P}(X_i > \lambda_i t) \leq \sum_i L\,\mathbb{P}(Y_i > \lambda_i t) \leq L \sum_i \mathbb{P}(\lambda_i^{-1} Y_i > t)$$

and the claim follows. □

C.3. Useful lemmata

In this section, we give some simple lemmata, which are useful tools for moving back and forth between strict and expected time. The results given in this section are not asymptotic, and given for simple objects. Nevertheless, it is straightforward to show that all constructions can be directly applied in the asymptotic setting.

C.3.1. Runtime truncations

We give generic variants of runtime truncation lemmata.

Corollary C.8. *Suppose* A *is some algorithm. Suppose* $\mathsf{A}(x)$ *takes an expected number of* t_x *steps on input* x. *Then the output distribution of* $\mathsf{A}(x)^{\leq N}$, *has statistical distance at most* $\frac{t_x}{N}$ *from* $\mathsf{A}(x)$.

Corollary C.8 bounds the quality loss when converting expected to strict time algorithms. For example, if A is a distinguisher with advantage ε, and $t_x \leq t$ for all inputs, then truncating runtime after $2\varepsilon^{-1}t$ steps yields a distinguisher with advantage $\frac{1}{2}\varepsilon$. If $t = \mathsf{poly}$ and $\varepsilon \geq 1/\mathsf{poly}$, then this transforms an *expected* polynomial time distinguisher into a *strict* polynomial time distinguisher.

Corollary C.9 (Non-asymptotic generic "standard reduction"). *Suppose $\mathcal{D}^{\mathcal{O}}$ is a distinguisher with advantage ε for* timed *oracles $\mathcal{O}_0$, $\mathcal{O}_1$. Let $T_0 = \mathsf{time}_{\mathcal{D}+\mathcal{O}}(\mathcal{D}^{\mathcal{O}_0})$, and let $N = \mathsf{tail}^{\dagger}_{T_0}(\frac{\varepsilon}{4})$. Then there is an $\mathcal{A}$ with runtime $S_b = \mathsf{time}_{\mathcal{D}+\mathcal{O}}(\mathcal{A}^{\mathcal{O}_b})$ for $b = 0, 1$ bounded roughly by N (plus overhead for computing N and emulating $\mathcal{D}$), and $\mathcal{A}$ distinguishes $\mathcal{O}_0$ and $\mathcal{O}_1$ with advantage $\frac{\varepsilon}{4}$.*

More precisely, $\mathcal{A}$ truncates the total time of $\mathcal{D} + \mathcal{O}$ to at most N steps, hence the runtime distribution of $\mathcal{A}$ is close to that of $\mathcal{D} + \mathcal{O}$. Moreover, there are two possible candidates for $\mathcal{A}$: One outputs the output of $\mathcal{D}$, and a random guess in case of `timeout`*. The other outputs* 1 *in case of* `timeout` *and* 0 *else. At least one of these algorithms has advantage $\frac{\varepsilon}{4}$.*

We note again, that only the runtime with $\mathcal{O}_0$ and its tail bound are of importance for the runtime cutoff. Also, one can trade-off runtime for advantage, e.g. by truncating at $N = \mathsf{tail}^{\dagger}_{T_0}(\frac{\varepsilon}{\mathsf{poly}})$. This cutoff argument and its variations play the role of the standard reduction to PPT (Corollary 4.2) in the general setting. We point out, that runtime is not the only (complexity) measure of interest which can be used in Corollary C.9. Besides elapsed runtime of $\mathcal{D} + \mathcal{O}$, the elapsed runtime of only $\mathcal{D}$, consumed memory, number of queries, query length, etc., are possible measures to which Corollary C.9 generalizes straightforwardly.

Proof sketch. Distinguisher $\mathcal{A}$ emulates $\mathcal{D}$ and truncates $\mathcal{D}$'s and $\mathcal{O}$'s combined steps to N. That is, $\mathcal{A}$ keeps track of the steps $t_{\mathcal{D}}$ and $t_{\mathcal{O}}$ and relies on $\mathcal{O}$ being a *timed* oracle to allow it a time bound of $N - t_{\mathcal{O}} - t_{\mathcal{D}}$ when invoked. Note that $\mathcal{A}$ emulates an priori number bounded number of N steps. Truncating $\mathcal{D}^{\mathcal{O}_0}$ after N steps w.r.t. oracle-included steps ensures that the output of $\mathcal{D}^{\mathcal{O}_0}$ has statistical distance at most $\frac{\varepsilon}{4}$.

Suppose the output of $\mathcal{A}^{\mathcal{O}_1}$ has statistical distance δ of $\mathcal{D}^{\mathcal{O}_1}$. If $\delta \geq \frac{2\varepsilon}{4}$. then necessarily, the probability that $T_1 = \mathsf{time}_{\mathcal{D}+\mathcal{O}}(\mathcal{D}^{\mathcal{O}_1})$ exceeds N steps is larger than $\frac{2\varepsilon}{4}$. Thus, this runtime statistic can be used as a distinguishing property, with advantage at least $\frac{\varepsilon}{4}$ infinitely often. (Concretely, $\mathcal{A}$ returns 1 if N steps are exceeded and 0 otherwise.)

Now suppose the probability that $T_1 = \mathsf{time}_{\mathcal{D}+\mathcal{O}}(\mathcal{D}^{\mathcal{O}_1})$ exceeds N steps is less than $\frac{2\varepsilon}{4}$. Let $\mathcal{A}$ guesses randomly in case of `timeout`. Then possible loss in advantage is bounded by $\frac{\varepsilon}{4} + \frac{2\varepsilon}{4} = \frac{3\varepsilon}{4}$. This leaves an advantage of $\frac{\varepsilon}{4}$ and the claim follows. □

Importantly, the construction of the two distinguisher candidates is uniform, and translates to the asymptotic setting. One of them has infinitely often advantage at least $\frac{\varepsilon}{4}$.

C.3.2. Hybrid lemmata

Hybrid arguments and therefore the hybrid lemma are omnipresent in cryptography. Unfortunately, the standard hybrid lemma for strict polynomial time does not hold without change.

Example C.10 (Expected polynomial rounds). The need to deal with a priori infinitely many hybrids arises naturally from expected polynomial interaction: We have $\sum_{i\geq 1} 2^{-i} = 1$, so repeating some protocol (step) with probability $\frac{1}{2}$ implies an expected constant number of repetitions. But replacing each call by a simulation requires an infinite number of hybrid steps. Evidently, after replacing the first κ protocols by simulations, the remainder can be replaced in a single step, because more than κ repetitions are necessary only with probability $2^{-\kappa}$.

We state in general the truncation approach from Example C.10.

Corollary C.11 (Hybrid lemma). *Let $\mathcal{O}^0, \mathcal{O}^1, \ldots, \mathcal{O}^\infty$ be oracles. Let $\mathcal{Z}_0$, $\mathcal{Z}_1$ be two more oracles, and let Z be an algorithm as follows: Z takes as input an integer $i \in \mathbb{N}$. Moreover, $\mathsf{Z}(i, \mathcal{Z}_b)$ is implements an oracle which behaves exactly like $\mathcal{O}^{i+b}$.*

Let $\mathcal{D}$ be a distinguisher for $\mathcal{O}^0$ and $\mathcal{O}^1$ with advantage ε, that is

$$|\mathbb{P}(\mathcal{D}^{\mathcal{O}^0} = 1) - \mathbb{P}(\mathcal{D}^{\mathcal{O}^\infty} = 1)| \geq \varepsilon$$

and suppose that we have a (tail) bound bnd *with*

$$|\mathbb{P}(\mathcal{D}^{\mathcal{O}^i} = 1) - \mathbb{P}(\mathcal{D}^{\mathcal{O}^\infty} = 1)| \leq \mathsf{bnd}(i).$$

Then for every $\alpha \leq \varepsilon$ there is a distinguisher $\mathcal{D}'$ which distinguishes $\mathcal{Z}_0$ and $\mathcal{Z}_1$ with advantage[68]

$$\varepsilon' = \frac{\varepsilon - \alpha}{N_\alpha} \quad \text{where } N_\alpha \coloneqq \mathsf{bnd}^\dagger(\alpha).$$

More concretely, $\mathcal{D}'$ picks a random $i \leftarrow \{0, N_\alpha - 1\}$, runs $\mathcal{D}$ on $\mathsf{Z}(i, \mathcal{Z}_b)$ and returns $\mathcal{D}$'s guess bit as its own. Thus, the runtime distribution of $\mathcal{D}'$ is closely related to that of $\mathcal{D}$ and Z.

Proof of Corollary C.11. We reduce the proof to the standard hybrid lemma. Note that it suffices to apply the standard hybrid lemma (with a finite number of steps) to $\mathcal{O}^0, \ldots, \mathcal{O}^{N_\alpha}$. Because, by the very definition N_α we know that $|\mathbb{P}(\mathcal{D}^{\mathcal{O}^0} = 1) - \mathbb{P}(\mathcal{D}^{\mathcal{O}^{N_\alpha}} = 1)| \geq \varepsilon - \alpha = \varepsilon'$. Now, the standard hybrid lemma yields our distinguisher and advantage. □

Our statement of the hybrid lemma differs from the standard one in minor points.[69] It allows an a priori infinite number of hybrids. And it postulates a bound bnd on the closeness of the i-th and final hybrid. Typically bnd bounds the statistical distance of the i-th and final hybrid and is derived as a tail bound, e.g. Markov bound (Lemma C.2) on runtime or number of oracle queries.

While one may hope for an "expected number of hybrids" loss, this is impossible in general, since an adversary could focus its advantage on the "tail hybrids". Any black-box-like reduction is unlikely to achieve better bounds.

Example C.12 (Optimality of the (truncated) hybrid argument). Consider following (non-adaptive) distinguishing game: The adversary sends a number n to the challenger. The challenger prepares n truly random r_i or n pseudo-random $r_i = \mathsf{PRG}(s_i)$, and the adversary must distinguish. Consider an adversary with distribution N of n, so that $\mathbb{E}(N) \leq 3$. The hope, that the hybrid argument may only lose a factor of 3 in advantage, is false. One the one hand, it may be that all the advantage of an adversary is in the tail of the distribution. Without a (non-obvious) way to reach the (distribution of) state, there is no other way than to run the adversary long enough. This affects any black-box reductions. A pathological adversary may furthermore distribute its advantage evenly over the hybrids as well, e.g. by first picking the number q of queries, and then breaking the i-th embedding with probability $\frac{1}{q}$. Consequently, improving on the hybrid lemma seems close to impossible. Better reductions have to make use of more information.

As is well known and for completeness demonstrated in Example C.12, the tails of distributions are a limiting factor for (hybrid) reductions. Nevertheless, Corollary C.11 is useful and generally good enough, though it may have poor tightness properties.

[68] We note that $N_\alpha = \infty$ is possible, in which case $\varepsilon' = 0$.

[69] Sometimes, the hybrid lemma is stated in a weaker form, merely ensuring the existence of an index i where distinguishing hybrids i and $i + 1$ has advantage $\geq \varepsilon/m$. This does not naively extended to the asymptotic setting. Assuming that for all i, $\mathcal{O}^i \overset{c}{\approx} \mathcal{O}^{i+1}$ (asymptotically) *does not* imply $\mathcal{O}^0 \overset{c}{\approx} \mathcal{O}^\infty$ (asymptotically). Trivial counterexamples exist. Hence, the reduction to a (single) fixed indistinguishability assumption is essential for asymptotic usage of Corollary C.11.

C.4. Testing closeness of distributions

Given two distributions, we need a way to efficiently test how close they are. Again, we give a non-asymptotic lemma. But we note that in the cryptographic setting, we will tell apart (families of) distributions which are statistically far (in the asymptotic sense).

Problem C.13 (Closeness promise problem). Let X, Y be distributions (typically on $\{1, \dots, n\}$). The **closeness promise problem** (with parameter $\varepsilon > 0$) is the following: Decide whether $X \overset{d}{\equiv} Y$ or $\Delta(X, Y) > \varepsilon$. A tester A is an algorithm which, given sample (oracle) access to X and Y outputs a verdict (i.e. a bit) whether $X = Y$ or not. The error of a tester is (at most) δ, if

$$\mathbb{P}(\mathscr{D}^{X,X'} = \texttt{same}) \geq 1 - \delta \quad \text{and} \quad \mathbb{P}(\mathscr{D}^{X,Y} = \texttt{different}) \geq 1 - \delta$$

We speak of *testing* instead of *distinguishing* since it is a slightly stronger notion. A distinguisher may guess randomly if $X \overset{d}{\equiv} Y$, but always decide $X \neq Y$ correctly, but a tester may not. In particular, a tester with error δ has distinguishing advantage $1 - 2\delta$.

Lemma C.14. *Let X, Y be distributions with support contained in on $\{1, \dots, m\}$ and consider the closeness promise problem. Let $\varepsilon, \delta \in (0, 1]$. Then there is an algorithm A which solves the closeness promise problem with error δ and requires*

$$N = \lceil 6m\varepsilon^{-2} \log(2\delta^{-1}) \rceil$$

samples (of both X and Y). Moreover, A is makes a linear number of arithmetic operations (in N).

The result generalizes to any X, Y with support contained in $\mathcal{S}$, where $\mathsf{card}(\mathcal{S}) = m$, since only comparison operations for elements are needed.

We note that better closeness testing algorithms are known, namely in [Cha+14] an *optimal* closeness tester is given. That tester has linear runtime in the number of samples N as well.

Proof of Lemma C.14. Our tester simply uses the Kolmogorov–Smirnov test. That is, compute the empirical CDF F_X and F_Y (with N samples each) and test whether $\|F_X - F_Y\|_\infty < \varepsilon$. By applying a Chernoff bound argument in case $X \overset{d}{\not\equiv} Y$, and using the sharp Dvoretzky–Kiefer–Wolfowitz inequality by Massart in case $X \overset{d}{\equiv} Y$, we arrive at the claimed result. (Our constants are chosen so that we obtain $(\varepsilon/3, \delta/2)$ approximations of the true CDF's. By a standard argument using the triangle inequality, one obtains our claims.) □

As with the hybrid lemma, we have to deal with distributions with infinite support. Using tail bounds, we stretch Lemma C.14 to this case.

Corollary C.15. *Let X, Y be distributions on $\mathbb{N}_0$ and consider the closeness promise problem. Let $\varepsilon, \delta \in (0, 1]$ and let $\mathsf{tail}_X(\cdot)$ be a tail bound for X. Suppose $\varepsilon' = \varepsilon - \alpha$, where $\alpha > 0$, let $m' = \mathsf{tail}_X^\dagger(\alpha)$. Then there is an algorithm A which solves the closeness promise problem with error δ and requires*

$$N' = \lceil 6(m' + 1)\varepsilon'^{-2} \log(2\delta^{-1}) \rceil$$

samples (of both X and Y). Moreover, A is only requires a linear number of arithmetic operations (in N').

We note that $\mathbb{N}_0 \cup \{\infty\}$ (and the like) are also domains for which Corollary C.15 holds. It should also be evident, that this generalizes, as long as we can approximate X and Y suitably precise over a suitably small domain. Hence tail bounds are just a special case, and replacing X, Y by suitable close X', Y' works as well.

Proof. The algorithm simply maps the distributions X, Y to new distributions by mapping any sample s to $\max\{s, m\}$.[70] This changes the statistical distance by at most $\mathsf{tail}_X(m)$, see Corollary C.5. Now, apply Lemma C.14. □

Following remark, while a triviality, points out one core tool of this work.

Important Remark C.16 (Statistical and computational indistinguishability coincide for "small" support). From Lemma C.14 and Corollary C.15, we already observe the following: Asymptotically, any pair of (families of) distributions X, Y, where one, say X, has (essentially) polynomial sized support $\{0, \ldots, \mathsf{poly}(\kappa)\}$ are *computationally* triple-oracle indistinguishable under repeated sampling in polynomial time, if and only if, they are *statistically* triple-oracle indistinguishable under repeated sampling.

Remark C.17. Merely considering the domain, independently of X is a very rough point of view. After all, X could be concentrated on a tiny subset of $\{0, \ldots, n\}$. In particular, relying on $\mathsf{supp}(X) \subseteq \mathbb{N}_0$ and using a total ordering and tail bounds, is not at all necessary. We consider a more sensitive closeness testing lemma a useful tool for more precise analysis. But the coarse (non-optimal) results stated here are good enough for our purposes.

D. ⋆ General runtime definitions

This section is (only) for the inclined reader. It contains our "general" treatment of runtime classes, that is, our framework and the many definitions necessary to talk about runtime classes and their properties. Unfortunately, we fall short of going beyond algebra-tailed runtime classes, hence by and large, nothing of essence is covered that is not already visible for polynomial time, PPT, EPT and CEPT.

D.1. Preliminaries: Bound algebras

Most of our arguments work for runtime classes related to bound algebras, for example, the algebra of polynomials.

Definition D.1 (Bound algebras). A **bound algebra** $\mathcal{B}$ is a subset of $\mathbb{R}_{\geq 0}^{\mathbb{N}_0}$, i.e. a subset of sequences in $\mathbb{R}_{\geq 0}$, which satisfies:

- $\mathcal{B}$ is the subset of non-negative sequences of a subalgebra of $\mathbb{R}^{\mathbb{N}_0}$. In particular, it is closed under multiplication and it contains the constant 0 and constant 1 sequences.[71]
- $\mathcal{B}$ is closed under domination, i.e. $(x_\kappa)_\kappa \in \mathcal{B}$, then so is any $(y_\kappa)_\kappa$ with $y_\kappa \leq x_\kappa$ (for all κ).
- $\mathcal{B}$ is "asymptotically monotone": If $(x_\kappa)_\kappa \in \mathcal{B}$, then so is $(y_\kappa)_\kappa$ with $y_\kappa \coloneqq \max_{i=1}^{\kappa} x_i$.

A subset $\mathcal{G} \subseteq \mathcal{B}$ is *generates* $\mathcal{B}$ if for any $(x_\kappa) \in \mathcal{B}$ there is a $(y_\kappa) \in \mathcal{G}$ with $(x_\kappa) \leq (y_\kappa)$. The set $\mathsf{Negl}_{\mathcal{B}}$ of $\mathcal{B}$**-negligible functions**, is defined as $\mathsf{Negl}_{\mathcal{B}} = \{f \mid \limsup_{\kappa\to\infty} |f(\kappa)\mathsf{bnd}(\kappa)| = 0\}$.

When we work with bounds we often implicitly assume they are monotone.

Example D.2. Suitable function algebras, e.g. polynomials, or polylogarithmic functions, or $f(\kappa) = n^{\mathsf{polylog}(\kappa)}$, etc., induce a bound algebra. Importantly, there typically are monotone generating subsets (of countable size), e.g. $\{(c\kappa^c) \mid c \in \mathbb{N}_0\}$, which generate $\mathcal{B}$.

[70] Note that this mapping does not need to "read" all of s (given e.g. tape-access starting from the least significant bit). In particular, in suitable machine models, we do not run into problems where some values s are gigantic and could not be read without compromising efficiency.

[71] The associated subalgebra of $\mathcal{B}$ is unique.

D.2. Runtime distributions

Our definitions of (polynomial) runtime are such that an algorithm's (or protocol's) runtime is bounded *in the security parameter κ alone*. The input space of an algorithm is (a family) $\mathcal{X}_\kappa$.[72] Often, our algorithms have no (explicit) input, but receive implicit input via oracles, e.g. when distinguishing distributions given sampling access. In any case, we focus on "a posteriori" runtime, i.e. consider runtime $\mathsf{time}_{\mathsf{A}}(\mathsf{A}(x))$ where $x \leftarrow \mathcal{X}$ for some input *distribution* (that is $\mathsf{A}(\mathcal{X})$ is a system *without* inputs).

Caution D.3. Recall that we generally suppress mentioning dependencies on the security parameter, i.e. we typically write $\mathsf{A}(x)$ instead of $\mathsf{A}(\kappa, x)$ if κ. The security parameter is (implicit) "input" to every algorithm. In fact, usually, A is given no inputs (but κ). Similarly, runtime obviously depends on the machine model even though we do not mention this.

Definition D.4 (Runtime distribution). A *(input-free)* **runtime (distribution)** T is a family $(T_\kappa)_\kappa$ of distributions $T_\kappa \in \mathsf{Dists}(\mathbb{N}_0 \cup \{\infty\})$ parameterized by κ; more precisely, it is a map $T\colon \mathbb{N}_0 \to \mathsf{Dists}(\mathbb{N}_0 \cup \{\infty\})$ from security parameter to probability distributions over $\mathbb{N}_0 \cup \{\infty\}$. A runtime T is **induced** by an algorithm A if $T_\kappa = \mathsf{time}_{\mathsf{A}}(\mathsf{A}(\kappa))$. We typically suppress κ and simply write $T = \mathsf{time}_{\mathsf{A}}(\mathsf{A})$.

We allow the symbol `timeout` in a runtime distribution T (formally changing to $\mathsf{Dists}(\mathbb{N}_0 \cup \{\texttt{timeout}\})$).[73]

Remark D.5. Runtime (distributions) with input, or *input-dependent* runtimes are functions mapping input $x \in \mathcal{X}_\kappa$ to a runtime distribution, that is $T_\kappa\colon \mathcal{X}_\kappa \to \mathsf{Dists}(\mathbb{N}_0 \cup \{\infty\})$ for all κ. It is **induced** by A if $T_\kappa(x) = \mathsf{time}(\mathsf{A}(\kappa, x))$. The definition of input-dependent runtime (as a random variable) is similar.

For now, we only consider the input-free setting, i.e. $\mathcal{X} = \{\star\}$. Input is implicitly made available via oracle access.

Caution D.6. In this and future sections, we conflate *runtimes* (random variables) *runtime distributions*. The reason is, that we almost always care only about the runtime distribution, except in cases where we "split" up the runtime of an algorithm into a sum of stochastically dependent runtimes (e.g. of A and $\mathcal{O}$).

D.3. Runtime classes

To talk about "efficient" computation, we need to say which runtime distributions we consider "efficient". The set of all "efficient" runtimes then forms the respective runtime class. Exemplary runtime classes are $\mathcal{PPT}$ and $\mathcal{EPT}$. We refine Definition 2.4 here, to only include sets of runtimes which have some basic properties.

Definition D.7. A **runtime class** $\mathcal{T}$ is a set of *input-free* runtime distributions so that:

Constants: The constant 0 and constant 1 runtime are in $\mathcal{T}$.
Closed under domination: that is, if $T \in T$ and $S \leq T$ then $S \in \mathcal{T}$.[74]
Closed under addition, i.e. $\mathcal{T} + \mathcal{T} \subseteq \mathcal{T}$, where $T + S$ is viewed as a sum of distributions.

An (oracle) algorithm A runs in $\mathcal{T}$**-time** if $\mathsf{time}(\mathsf{A}) \in \mathcal{T}$.

Closure under domination says that no "inefficient" algorithm (i.e. runtime outside $\mathcal{T}$) can be made efficient by doing *more* steps. Additive closure roughly ensures that independent execution of any constant number of efficient algorithms is efficient. The definition of runtime class is most likely incomplete. We just give enough properties so that our results hold. Sensible runtime classes should offer more guarantees, but we have not identified the "right" properties, see Appendix E.9 for more.

[72] Recall a well-known problem: The input space may not be (efficiently) recognizable. Thus, an algorithm may be fed with malformed input (or oracles/interaction). In general, this voids any runtime guarantees. Thus, for protocols, we want strong runtime guarantees, which are not restricted do well-formed input.

[73] We could also allow ∞ there, but generally timeouts stop overlong executions.

[74] More precisely, $S \leq T$ iff for all κ we have $S_\kappa \leq T_\kappa$, i.e. T_κ dominates S_κ in distribution.

Example D.8. We give some exemplary polynomial runtime classes.

Strict polynomial time: The runtime class $\mathcal{PPT}$ contains (by definition) all runtimes T for which there exists a polynomial poly such that $T \leq \mathsf{poly}$.

Expected polynomial time: The runtime class $\mathcal{EPT}$ contains (by definition) all runtimes T for which there exists a polynomial poly such that $\mathbb{E}(T) \leq \mathsf{poly}$, i.e. $\mathbb{E}(T_\kappa) \leq \mathsf{poly}(\kappa)$ for all κ.

Polynomial $\|\cdot\|_q$-time: By polynomially bounding $\|T\|_q$ (for $q \in [1,\infty]$), we generalize both strict ($q = \infty$) and expected time ($q = 1$). For example $q = 2$ implies polynomially bounded variation (and expectation).

Quasi-linear time: If we require $T_\kappa \leq \kappa \cdot \mathsf{polylog}(\kappa)$ we obtain quasi-linear runtime. This class only satisfies weak composition properties, and is not covered by our results.

Now, we generalize polynomial time bounds to algebra bounds. For that, we need following definition.

Definition D.9. We say that a runtime class $\mathcal{T}$ is **weakly compatible** with a bound algebra $\mathcal{B}$, if for any $\mathsf{bnd}_0 \in \mathcal{B}$, there is a $\mathsf{bnd}_1 \in \mathcal{B}$ so that bnd_1 can be computed in $\mathcal{T}$-time. More concretely, $\mathsf{bnd}_1(\kappa)$ can be computed in time T_κ for $T \in \mathcal{T}$.

We call $\mathcal{T}$ **(strongly) compatible** with $\mathcal{B}$ if additionally strict $\mathcal{B}$-time (see Definition D.10 below) is contained in $\mathcal{T}$.

Compatibility ensures that $\mathcal{T}$ and $\mathcal{B}$ behave well in reduction arguments. (Strong) Compatibility is simpler to work with than weak compatibility, since for example $\mathcal{PPT}$ is weakly compatible with $\mathcal{B} = 2^{O(\kappa)}$, but does evidently not contain all strict $\mathcal{B}$ algorithms.

Definition D.10 (Bound algebras and runtime classes). Instead of polynomials, some (suitable) algebra $\mathcal{B}$ may be used for time bounds, e.g. $n^{\mathsf{polylog}(n)}$, see Definition D.1. By definition, we always require that the defined runtime class $\mathcal{T}$ is *compatible* with the bound algebra $\mathcal{B}$.

Algebra-bounded $\|\cdot\|_q$-time: We write $\mathsf{RTC}_q(\mathcal{B})$ for the runtime class containing all runtimes T with $\|T_\kappa\|_q \leq \mathsf{bnd}(\kappa)$ for some $\mathsf{bnd} \in \mathcal{B}$.

Algebra-tailed time: We generalize algebra-bounded time as follows: A runtime class $\mathcal{T}$ is **$\mathcal{B}$-tailed**, for a bound algebra $\mathcal{B}$, if: For every $T \in \mathcal{T}$, for every $\mathsf{bnd}_{\mathsf{tail}} \in \mathcal{B}$, there is a $\mathsf{bnd}_T \in \mathcal{B}$, such that $\mathbb{P}(T_\kappa > \mathsf{bnd}_T(\kappa)) \leq \frac{1}{\mathsf{bnd}_{\mathsf{tail}}(\kappa)}$ for all κ.[75]

We also refer to algebra-bounded times via *strict (or expected) $\mathcal{B}$-time*.

By Lemma C.2, any algebra-bounded runtime class is also algebra-tailed. Namely pick $\mathsf{bnd}_T = t \cdot \mathsf{bnd}_{\mathsf{tail}} \geq \mathsf{tail}^\dagger(\frac{1}{\mathsf{bnd}_{\mathsf{tail}}})$, where $t = \|T\|_q$. Also, Levin's relexation of EPT is polynomially-tailed.

We will focus on algebra-tailed runtime classes and runtime classes we derived from them. Dealing with more general runtime classes is an interesting open problem, see Appendix E.9.

Lastly, we define "abstract" runtime cutoffs.

Definition D.11 (Runtime truncation). Let T be a runtime. We define the **runtime cutoff** or **runtime truncation** $T^{\leq N}$ of T after N steps as the distribution (or random variable) given by $T|_{(\cdot > N) \mapsto \texttt{timeout}}$, i.e. by mapping any $k > N$ to `timeout` (and the identity mapping otherwise). Runtime truncation is assumed to be an *efficient* oracle-transformation in any suitable machine model.[76]

Remark D.12. We stress that an efficient implementation of runtime cutoffs is vital for any results making use of them. We also note that this means that the *truncation bounds themselves must be efficiently computable*. This is ensured by the compatibility requirement in Definition D.10.

[75] Recall that asymptotics, should be part of $\mathcal{B}$, so we use for *all* κ (and not for almost all).

[76] This means that applying runtime cutoff to a *runtime oracle* is efficient. For example, given tape access to bit-encoded oracle results, we can read the minimal number of bits necessary to recognize $t > N$ and then return `timeout`.

D.4. $\mathcal{T}$-time triple-oracle indistinguishability

There are several notions of indistinguishability of distributions X_0, X_1 w.r.t. to $\mathcal{T}$-time algorithms. We choose indistinguishability *under repeated sampling* with *additional sampling access* to X_0 and X_1. The decision to give oracle sampling access the distributions X_0 and X_1, as well as the challenge distribution $Z \stackrel{d}{\equiv} X_b$ mirrors the fact that an algorithm can be (independently) executed many times, and should still remain efficient.[77] In particular, if $X_0 = \mathsf{time}(\mathsf{A}_0)$ is the runtime distribution of an efficient algorithm, and $X_1 = \mathsf{time}(\mathsf{A}_1)$ is inefficient, then X_1 is not efficiently samplable by emulating A_1. To simplify, we assume sampling access to both X_0 and X_1. Recall that we assume access to *binary* encodings of runtime.[78]

Another simplification is that we require *constant* distinguishing advantage. By standard amplification techniques, this is equivalent to non-$\mathcal{B}$-negligible success for algebra-tailed runtime classes.

Definition D.13 (Triple-oracle distinguishing). Let $\mathcal{O}_0$ and $\mathcal{O}_1$ be sampling oracles for distributions X_0, X_1 (i.e. oracles which return a fresh sample distributed as X_b when queried). Consider the distinguishing experiment $\mathrm{Exp}^{\text{3-dist}}_{\mathcal{A},\mathcal{O}_0,\mathcal{O}_1}$.

Experiment $\mathrm{Exp}^{\text{3-dist}}_{\mathcal{A},\mathcal{O}_0,\mathcal{O}_1}(\kappa)$
- $b \leftarrow \{0,1\}$
- Instantiate an independent $\mathcal{O}^* := \mathcal{O}_b$
- $b' \leftarrow \mathcal{A}^{\mathcal{O}_0,\mathcal{O}_1,\mathcal{O}^*}(\kappa)$
- **return** $b' \stackrel{?}{=} b$

The distinguishing advantage of an algorithm $\mathcal{D}$ is defined as

$$\begin{aligned}\mathrm{Adv}^{\text{3-dist}}_{\mathcal{D},\mathcal{O}_0,\mathcal{O}_1}(\kappa) :=& 2\,\mathbb{P}(\mathrm{Exp}^{\text{3-dist}}_{\mathcal{A},\mathcal{O}_0,\mathcal{O}_1}(\kappa) = 1)\\ =& |\mathbb{P}(\mathcal{D}^{\mathcal{O}_0,\mathcal{O}_1,\mathcal{O}_1^*}(\kappa) = 1) - \mathbb{P}(\mathcal{D}^{\mathcal{O}_0,\mathcal{O}_1,\mathcal{O}_0^*}(\kappa) = 1)|,\end{aligned}$$

where $\mathcal{O}_b^* = \mathcal{O}_b$, but independent. (The second equality only holds if $\mathcal{D}$ always returns a bit.) The randomness is taken over the algorithms and oracles randomness.

A distinguisher $\mathcal{D}$ is $\mathcal{T}$-time, if $\mathsf{time}_{\mathcal{D}}(\mathrm{Exp}^{\text{3-dist}}_{\mathcal{A},\mathcal{O}_0,\mathcal{O}_1}) \in \mathcal{T}$.[79] We call $\mathcal{O}_0$ and $\mathcal{O}_1$ **($\mathcal{T}$-time) computationally (triple-oracle) indistinguishable**, written $\mathcal{O}_0 \overset{c}{\approx}_{\mathcal{T}} \mathcal{O}_1$, if for all $\mathcal{T}$-time distinguishers $\mathcal{D}$,

$$\mathrm{Adv}^{\text{3-dist}}_{\mathcal{D},\mathcal{O}}(\kappa) \in o(1).$$

that is, any distinguisher has asymptotically vanishing advantage. Put differently, a computational distinguisher must have constant advantage $c > 0$ (for infinitely many κ). We define **$\mathcal{T}$-query statistical indistinguishability** as $\mathcal{T}$-time indistinguishability, where we only count a query to an oracle as a step (costing unit time).

We use Definition D.13 only for (runtime) distributions, and not general oracle-indistinguishability.

Remark D.14 (Why no general advantage classes?). For algebra-tailed runtime classes, using non-constant advantage, namely non-$\mathcal{B}$-negligible advantage, also works (due to amplification). We could define general "advantage classes", such as subexponentially negligible, polynomially negligible, or

[77] Our notion behaves nicely in almost any aspect, and agrees with standard notions if X_0 and X_1 are efficiently samplable (by a standard hybrid argument). We can amplify distinguishing advantage (as usual) and are guaranteed that *statistically indistinguishable distributions are statistically close.* Neither of this holds for the usual notions of one-sample or k-sample distinguishing, see for example [Mey94; GM98; GS98].

[78] A unary encoding would work as well, since we always reduce (a a priori strict time) distinguishers which use a strictly truncated version of the time. This truncated time can be read efficiently in both unary an binary.

[79] Equivalently, $\mathsf{time}_{\mathcal{D}}(\mathcal{D}^{\mathcal{O}_0,\mathcal{O}_1,\mathcal{O}_b}) \in \mathcal{T}$ for $b = 0, 1$.

$1 - \frac{1}{\kappa}$. One reason not to do this is our focus on indistinguishability of runtimes, not in general distributions. We crucially rely on tail bounds and triple-oracle indistinguishability, which leads to (maybe unnecessary) limitation. In the "low advantage regime", e.g. subexponential advantage, it seems that the arguments based tail bounds do not carry over. In the "high advantage regime", e.g. advantage of at most $1 - 1/\mathsf{poly}$, the use of triple-oracle (in particular repeated samples) makes possible results uninteresting and useless (due to amplification to $1 - \exp(\kappa)$).

Alternative proof techniques, which do not rely on (repeated) sample access and tail bounds as their central tool are required. It is likely, that the approach(es) in Appendices E.3.1 and E.3.2 could be extended. This is out of scope for this work.

Since it is a useful point of view, we slightly generalize distinguishing. Namely, instead of directly outputting a verdict, one may output some processed information, which is fed into another distinguisher (perhaps repeatedly).

Remark D.15 (Generalized distinguisher). Let us call a distinguisher, which outputs not only 0 or 1, but different or additional information, a *generalized distinguisher*. Clearly, if two distributions are (computationally) indistinguishable, then the output of any generalized distinguisher is also (computationally) indistinguishable.

The upshot of this deliberation is that *any efficiently computable statistic* of an execution of a distinguisher $\mathcal{D}$ must be *indistinguishable*. Otherwise, there is a distinguisher $\mathcal{D}'$ which emulates $\mathcal{D}$ and uses that statistic to attack indistinguishability. In particular, *runtime* is such a statistic, and the number of oracle queries is another.

Now, we apply the notion of $\mathcal{T}$-time triple-oracle indistinguishability to runtimes.

Definition D.16. Suppose $\mathcal{T}$ is a (input-free) runtime class. Let T resp. S be (arbitrary) runtimes and suppose $\mathcal{O}_0$ resp. $\mathcal{O}_1$ sample T resp. S. We call T and S **(computationally) $\mathcal{T}$-time (triple-oracle) indistinguishable** if the respective distributions are (computationally) $\mathcal{T}$-time triple-oracle indistinguishable. We also write $T \overset{c}{\approx}_{\mathcal{T}} S$. The definition of **statistically $\mathcal{T}$-query (triple-oracle) indistinguishable** runtimes is analogous, written $T \overset{s}{\approx}_{\mathcal{T}} S$.

In the following, we always mean *triple-oracle* indistinguishable, if not otherwise specified. We come back to standard indistinguishability only in Appendix D.7

D.5. Closed runtime classes

Now, we come to a central definition, which applies the principle that $\mathcal{T}$-time indistinguishable objects should be considered "identical" for all cryptographic intents and purposes to $\mathcal{T}$-time itself.

Definition D.17 ($\mathcal{T}$-closed). Suppose $\mathcal{T}$ and $\mathcal{S}$ are runtime classes. We call $\mathcal{S}$ **computationally** (resp. **statistically**) **$\mathcal{T}$-closed** if following holds: For all runtimes S, if there is a runtime $\widetilde{S} \in \mathcal{T}$ and $S \overset{c}{\approx}_{\mathcal{S}} \widetilde{S}$ (resp. $S \overset{s}{\approx}_{\mathcal{S}} \widetilde{S}$), then $S \in \mathcal{S}$.

We call a runtime class $\mathcal{T}$ computationally (resp. statistically) **closed**, if it is $\mathcal{T}$-closed.

Example 1.5 demonstrates that neither $\mathcal{PPT}$ nor $\mathcal{EPT}$ is a closed runtime class. Before we define the closure of a runtime class, we give some helpful definitions.

Definition D.18 (Generating set). Let $\mathcal{U}$ be a *set* of runtimes. We say $\mathcal{U}$ **generates** $\mathcal{T}$ if $\mathcal{U} \subseteq \mathcal{T}$ and for any runtime class $\mathcal{T}'$ containing $\mathcal{U}$, we have $\mathcal{T} \subseteq \mathcal{T}'$. Equivalently, $T \in \mathcal{T} \iff \exists S \in \mathcal{U} \colon T \leq S$. Equivalently, $\mathcal{T}$ is the minimal runtime class containing $\mathcal{U}$.[80]

This shows that indistinguishability w.r.t. any generating subset $\mathcal{U} \subseteq \mathcal{T}$ or w.r.t. $\mathcal{T}$ coincides. For example, for $\mathcal{PPT}$, the set $\{\mathsf{poly}(\kappa) = n\kappa^n \mid n \in \mathbb{N}\}$ is generating, since every runtime is dominated by a runtime in this set.

[80] It is easy to see that an arbitrary intersection of runtime classes is again a runtime class. Hence, the generated runtime class of $\mathcal{U}$ is the intersection of all runtime classes containing $\mathcal{U}$, in particular, it exists and is unique.

Remark D.19. We can translate generating sets to the setting of bound algebras. Indeed, in Definition D.10, we require a generating set of efficiently computable bounds.

The perhaps most important relation between sets of runtimes is the following.

Definition D.20 (D-dense). A subset of runtimes $\mathcal{U} \subseteq \mathcal{T}$ is called **computationally** (resp. **statistically**) **distinguishing-dense** (short **d-dense**) in runtime class $\mathcal{T}$ if for any pair of distributions X, Y (over $\mathbb{N}_0 \cup \{\infty\}$) we have

$$X \overset{c/s}{\approx}_{\mathcal{T}} Y \implies X \overset{c/s}{\approx}_{\mathcal{U}} Y$$

w.r.t. triple-oracle indistinguishability. In other words, if $\mathcal{T}$ can distinguish two distributions, so can $\mathcal{U}$. A weakening of d-dense is **runtime d-dense**, where X must be in $\mathcal{T}$.

We note that d-density of $\mathcal{U} \subseteq \mathcal{T}$ is much different from being generating. For example, $\mathcal{PPT} \subseteq \mathcal{EPT}$ is d-dense, since any (successful) *expected* polynomial time distinguisher can be transformed into a (still successful) *strict* polynomial time distinguisher, see Corollary C.8.

Lemma D.21. *Let $\mathcal{T} \subseteq \mathcal{S}$ be runtime classes. Suppose that $\mathcal{S}$ is computationally $\mathcal{T}$-closed and that $\mathcal{T}$ is computationally (runtime) d-dense in $\mathcal{S}$. Then $\mathcal{S}$ is computationally closed. The same holds in the statistical case.*

Proof. Let $\widetilde{T} \in \mathcal{S}$ and let T be some runtime. Suppose $\widetilde{T} \overset{c}{\approx}_{\mathcal{S}} T$. Then $\widetilde{T} \overset{c}{\approx}_{\mathcal{T}} T$, since $\mathcal{T}$ is runtime d-dense in $\mathcal{S}$ and $\widetilde{T} \in \mathcal{S}$. Then $T \in \mathcal{S}$, since $\mathcal{S}$ is computationally $\mathcal{T}$-closed. The statistical case follows analogously. □

We now give a (constructive) definition of the closure of a runtime class.

Definition D.22 (Closure). Let $\mathcal{T}$ and $\mathcal{S}$ be a runtime classes. We define the computational $\mathcal{S}$-**closure** $\mathrm{Cls}^c_{\mathcal{S}}(\mathcal{T})$ of $\mathcal{T}$ as

$$\mathrm{Cls}^c_{\mathcal{S}}(\mathcal{T}) \coloneqq \{S \colon \mathbb{N}_0 \to \mathsf{Dists}(\mathbb{N}_0 \cup \{\infty\}) \mid \exists T \in \mathcal{T} \colon S \overset{c}{\approx}_{\mathcal{S}} T\}.$$

The statistical $\mathcal{S}$-closure $\mathrm{Cls}^S_{\mathcal{S}}(\mathcal{T})$ is defined analogously. The **closure** $\overline{\mathcal{T}}$ of $\mathcal{T}$ is $\mathrm{Cls}^{c/s}_{\mathcal{T}}(\mathcal{T})$ (whether computational or statistical will be clear from the context).

An abstract notion of closure (e.g. minimal closed runtime class containing $\mathcal{T}$) and its equivalence with Definition D.22 would be a good justification for our definition. However, we do not even know whether we have a proper definition of runtime classes which could support such a result, see Appendix E.9.

Lemma D.23 (Closures are closed). *The closure $\overline{\mathcal{T}}$ of a runtime class $\mathcal{T}$ is closed. (This holds in the computational and the statistical case.)*

Proof. Consider a runtime $T \in \overline{\mathcal{T}}$ and some arbitrary runtime S and suppose that $T \overset{c}{\approx}_{\overline{\mathcal{T}}} S$. To show that $\overline{\mathcal{T}}$ is closed, we need $S \in \overline{\mathcal{T}}$. Since $\mathcal{T} \subseteq \overline{\mathcal{T}}$, we have $T \overset{c}{\approx}_{\mathcal{T}} S$. By definition of $\overline{\mathcal{T}}$, there is some $\widetilde{T} \in \mathcal{T}$ such that $\widetilde{T} \overset{c}{\approx}_{\mathcal{T}} T$. Now, we have $\widetilde{T} \overset{c}{\approx}_{\mathcal{T}} T \overset{c}{\approx}_{\mathcal{T}} S$. This implies $S \in \overline{\mathcal{T}}$ by definition of $\overline{\mathcal{T}}$.[81] This proves the claim. The statistical case follows analogously. □

We would like a stronger result. We state this in following conjecture, which has little support for general runtime classes.

Conjecture D.24 (Closures are small). For any "benign" runtime class $\mathcal{T}$, $\mathcal{T}$ is runtime d-dense in $\overline{\mathcal{T}}$.

[81] Triple-oracle indistinguishability is transitive for any *constant* number of hops.

We expect that runtime classes where Conjecture D.24 fails behave rather strangely. While we do not know what "benign" runtime classes are or how to prove Conjecture D.24 in general, it is simple for algebra-tailed runtime classes.

Lemma D.25. *Let $\mathcal{B}$ be a bound algebra and $\mathcal{T}$ be $\mathcal{B}$-tailed. Then,* strict *$\mathcal{B}$-time is d-dense in $\overline{\mathcal{T}}$. (This holds in the computational and statistical case.)*

Proof sketch. Suppose $\mathcal{D}$ is a $\overline{\mathcal{T}}$-time distinguisher of distributions X and Y with advantage $\geq \varepsilon$ (for infinitely many κ and constant ε). Let $T = \mathsf{time}(\mathcal{D})$. We know that $T \overset{c}{\approx}_{\mathcal{T}} \widetilde{T}$ for some $\widetilde{T} \in \mathcal{T}$. Thus, for any $\mathcal{T}$-computable bound bnd, we have $|\mathbb{P}(T_\kappa \leq \mathsf{bnd}(\kappa)) - \mathbb{P}(\widetilde{T}_\kappa \leq \mathsf{bnd}(\kappa))| \leq o(1)$. Otherwise T and $\widetilde{T}$ would be $\mathcal{T}$-time distinguishable.

Since $\mathcal{T}$ is $\mathcal{B}$-tailed, there exist an (efficiently computable) bound $\mathsf{bnd}(\kappa) \geq \mathsf{tail}^{\dagger}_{\widetilde{T}_\kappa}(\frac{2}{3}\varepsilon)$. Consequently, $\mathcal{D}^{\leq \mathsf{bnd}}$ is strict $\mathcal{B}$-time, hence $\mathcal{T}$-time, and retains a distinguishing advantage of $\frac{2}{3}\varepsilon - o(1)$ (infinitely often), which is at least $\frac{1}{2}\varepsilon$ infinitely often. □

We note an interesting step in the argument: The connection to $\mathcal{D}$'s runtime T is indirect, since we rely on $\widetilde{T}$ instead. We only needed suitable bounds for *truncation*. Indeed, runtime truncation seems to be the central (and only) tool at our disposal, and someway or another, it is what our proofs rely on.

Remark D.26 (Efficiency of truncations). Note that $\mathsf{time}_{\mathcal{D}}(\mathcal{D}^{\leq \mathsf{bnd}}) \leq \mathsf{time}_{\mathcal{D}}(\mathcal{D})$ (up to emulation overhead), that is, the truncation is "as efficient as" $\mathcal{D}$, and only loses advantage/output quality.

Remark D.27 (Non-negligible advantage). Lemma D.25 immediately extends to advantage $\varepsilon = 1/\mathsf{bnd}(\kappa)$ (for infinitely many κ). Just replace $o(1)$ by $\mathsf{negl}_{\mathcal{B}}$ and note that $\mathsf{tail}^{\dagger}_{\widetilde{T}_\kappa}(\alpha \frac{1}{\mathsf{bnd}(\kappa)}) \in \mathcal{B}$ for any constant $\alpha > 0$ and $\mathsf{bnd} \in \mathcal{B}$. This direct "conversion" to the usual setting of non-negligible advantage typically works for our results concerning algebra-tailed runtime classes.

Following lemma is useful to check if some runtime class $\mathcal{S}$ is the closure of $\mathcal{T}$.

Lemma D.28 (Closures are minimal). *Let $\mathcal{T} \subseteq \mathcal{S} \subseteq \overline{\mathcal{T}}$ be runtime classes. Suppose that $\mathcal{S}$ is $\mathcal{T}$-closed and $\mathcal{T}$ is d-dense in $\mathcal{S}$. Then $\mathcal{S} = \overline{\mathcal{T}}$. (This holds in the computational and statistical case.)*

Proof. Similar to Lemmas D.21 and D.23. (Any element in $\overline{\mathcal{T}}$ also lies $\mathcal{S}$.) □

Let us consider a simple concrete example.

Example D.29 (CPPT). We denote the closure of $\mathcal{PPT}$ as $\overline{\mathcal{PPT}}$ or $\mathcal{CPPT}$ and call it **computationally probabilistic polynomial time** (CPPT). In Appendix D.6, we find that statistical and computational closure coincide, hence "CPPT = SPPT". By definition, $\mathcal{CPPT}$ is

$$\mathcal{CPPT} = \{T \mid \exists \mathsf{poly}, \mathsf{negl}\colon\ \mathbb{P}(T_\kappa \geq \mathsf{poly}(\kappa)) \leq \mathsf{negl}(\kappa)\}.$$

In other words, CPPT relaxes PPT by allowing a negligible amount of superpolynomial executions. Now, we check that $\mathcal{CPPT} = \overline{\mathcal{PPT}}$. Clearly, $\mathcal{CPPT}$ contains $\mathcal{PPT}$. It is easy to see that, $\mathcal{PPT}$ is d-dense in $\mathcal{CPPT}$ and $\mathcal{CPPT}$ is $\mathcal{PPT}$-closed. Since also $\mathcal{CPPT} \subseteq \overline{\mathcal{PPT}}$, we find equality from Lemma D.21 and Lemma D.28.

D.6. Equivalence of runtime-indistinguishability for algebra-tailed runtime classes

In this section, we establish that for an algebra-tailed runtime class $\mathcal{T}$, statistical and computational $\mathcal{T}$-time indistinguishability coincide of runtime distributions. We give two such lemmata. The first one is simple and illustrates underlying reasons using strict algebra-bounded runtime classes. The second one extends this to algebra-tailed runtime classes. Both lemmata seem inherently limited to runtime classes containing a large enough "strict" subclass.

Lemma D.30. *Let $\mathcal{B}$ be a bound algebra and $\mathcal{T} = \mathsf{RTC}_\infty(\mathcal{B})$ be the corresponding* strict *runtime class. Let $T \in \mathcal{T}$ and let S be some runtime. Then $T \overset{s}{\approx}_{\mathcal{T}} S$ implies $T \overset{c}{\approx}_{\mathcal{T}} S$. More generally, if X and Y are distributions supported on a set S with cardinality $\mathsf{card}(S)$ in $\mathcal{B}$, then statistical and computational indistinguishability coincide. The (efficient) distinguisher is as in Lemma C.14 with parameters so that it runs in strict $\mathcal{B}$-time.*

Let X be a distribution or a random variable. For convenience, we write $X^{\{k\}}$ for the k-fold product distribution of *stochastically independent* products. That is, $(x_1, \dots, x_k) \leftarrow X^{\{k\}}$ is distributed as $x_i \leftarrow X$ for k independent samples x_i.

Proof. Note that computational distinguishability implies statistical distinguishability. To prove the converse, we invoke Lemma C.14. Let $\mathsf{bnd}_0 \in \mathcal{B}$ bound the support size of the distributions X, Y.[82] The key point is: If $X \overset{s}{\not\approx}_{\mathcal{T}} Y$, then the statistical distance is lower-bounded by $1/\mathsf{bnd}_{\mathsf{stat}}$ for some efficiently computable $\mathsf{bnd}_{\mathsf{stat}} \in \mathcal{B}$. Otherwise $\Delta(X^{\{\mathsf{bnd}\}}, Y^{\{\mathsf{bnd}\}}) \leq \mathsf{bnd} \cdot \Delta(X, Y) \in o(1)$ for all bnd, and hence $X^{\{\mathsf{bnd}\}}$, $Y^{\{\mathsf{bnd}\}}$ are statistically close, for any (statistical) distinguisher. A contradiction to triple-oracle distinguishability.

We invoke Lemma C.14 with $n = \mathsf{bnd}_0$, $\varepsilon = \frac{1}{2\mathsf{bnd}_{\mathsf{stat}}}$, and δ small enough, say $\delta = 1/8$. We obtain a distinguisher $\mathcal{D}$ with runtime roughly $24\mathsf{bnd}_0(\kappa)\mathsf{bnd}_{\mathsf{stat}}(\kappa)^2$ plus the overhead for evaluating $\mathsf{bnd}_0(\kappa), \mathsf{bnd}_{\mathsf{stat}}(\kappa)$. Thus, $\mathcal{D}$ is efficient. □

As we have seen, the equivalence between statistical and computational indistinguishability of runtimes follows because the support of a runtime distribution is "small", compared to the allotted runtime for distinguishers. This, of course, is by definition of runtime resp. "small".

Now, we generalize Lemma D.30 just like we generalized Lemma C.14 to Corollary C.15.

Corollary D.31. *Let $\mathcal{B}$ be a bound algebra and let $\mathcal{T}$ be a $\mathcal{B}$-tailed runtime class. Let X, Y be distributions over $\mathbb{N}_0 \cup \{\infty\}$ and suppose that X is $\mathcal{B}$-tailed, i.e. we have a tail bound tail_X such that*

$$\forall\, \mathsf{bnd} \in \mathcal{B}\colon\ \mathsf{tail}^{\dagger}_{X_\kappa}\left(\frac{1}{\mathsf{bnd}(\kappa)}\right) \in \mathcal{B}.$$

Then $X \overset{s}{\approx}_{\mathcal{T}} Y$ implies $X \overset{c}{\approx}_{\mathcal{T}} Y$. In particular, any runtime distribution $X = T \in \mathcal{T}$ is $\mathcal{B}$-tailed by assumption. The (efficient) distinguisher is as in Corollary C.15 with parameters so that it runs in strict *$\mathcal{B}$-time. In particular, $\mathsf{RTC}_\infty(\mathcal{B})$ is d-dense is in $\mathsf{RTC}_q(\mathcal{B})$.*

Proof. **Step 1**: We recall Corollary C.15 in our situation: Suppose $\Delta(X_\kappa, Y_\kappa) \geq \varepsilon(\kappa)$, and let $\delta > 0$, and $\alpha \in [0, \varepsilon]$. Then there is a distinguisher with advantage at least $1 - 2\delta$, which requires

$$N = \lceil 6N_\alpha(\varepsilon - \alpha)^{-2} \log(2\delta^{-1}) \rceil$$

samples, where $N_\alpha \coloneqq \mathsf{tail}^{\dagger}_X(\alpha)$ and has runtime quasi-linear in N (in admissible machine models).

Step 2: Arguing that the statistical distance $\Delta(X, Y)$ is lower-bounded by $1/\mathsf{bnd}$ infinitely often, is not as trivial as in Lemma D.30. Indeed, we rely on the general hybrid lemma (Corollary C.11) and hence on tail bounds. Suppose the statistical distinguisher has advantage $\geq c$ (infinitely often for constant c). By a standard hybrid argument, Corollary C.11, we find a distinguisher which accesses the challenge oracle only once, and has advantage at least

$$\frac{c - \beta}{N_\beta} \quad \text{where} \quad N_\beta \coloneqq \mathsf{tail}^{\dagger}_{\mathcal{D}_{\mathsf{stat}}}(\beta) \quad \text{for any } \beta \in [0, c].$$

Consequently, $\Delta(X, Y) \geq \frac{c-\beta}{N_\beta}$ for any choice of β. (Note that ε and N_β vary in κ.)

[82] To be precise, it is lower-bounded only for infinitely many κ.

Step 3: Putting Steps 1 and 2 together by (arbitrarily) choosing $\beta = c/2$ we find $\varepsilon = \frac{c}{2N_\beta}$. and $\alpha = \varepsilon/2$ we find

$$N = \lceil 6N_\alpha(\frac{1}{2}\varepsilon)^{-2}\log(2\delta^{-1})\rceil = \lceil 24N_\alpha N_\beta^2 \log(2\delta^{-1})\rceil.$$

Our constructed distinguisher $\mathcal{D}$ needs N samples and has advantage at least $1-2\delta$ for infinitely many κ. Now, $N_\alpha = \mathsf{tail}^\dagger_X(\alpha) \in \mathcal{B}$ by assumption that X is $\mathcal{B}$-tailed. Also, $N_\beta = \mathsf{tail}^\dagger_{\mathcal{D}}(\beta) \in \mathcal{B}$ for any constant β, since $\mathcal{D}_{\mathsf{stat}}$ is statistical $\mathcal{T}$-time, hence the number of oracle-queries is $\mathcal{B}$-tailed. Consequently, $N_\alpha N_\beta^2 \in \mathcal{B}$, and we find that $N \in \mathcal{B}$ for any suitable (e.g. constant) advantage $1-2\delta$. We obtain a strict $\mathcal{B}$-time distinguisher as promised. □

As in Remark D.27, one can directly generalize to non-negligible advantage.

Corollary D.32. *The result of Corollary D.31 extends to the* closure $\overline{\mathcal{T}}$ *of any (suitable) $\mathcal{B}$-time class $\mathcal{T}$. Moreover, it extends to any runtime class in which $\mathcal{T}$ is d-dense.*

D.7. From oracles to emulation and standard indistinguishability

In this section, we abstract properties of runtimes *induced* by algorithms in what we call *continuously samplable*. For such runtimes, we show the equivalence of standard indistinguishability and triple-oracle indistinguishability, which was as introduced for specially runtimes.

Up until now, we treated runtimes as distributions which are samplable via oracle access. This helped keep our options limited and the presentation clean. For applications, we deal with *induced* runtimes of algorithms, and we pay a *non-constant* price for sampling them. To sample the runtime of an algorithm, we *emulate* it. Fortunately, such induced runtimes have a very useful intrinsic property: They are continuously samplable in following sense. To know whether a concrete realization of T is larger than k, we have to emulate at most k steps. If emulation is efficient, and T is efficient, we can therefore sample efficiently. Similarly, if our runtime cutoff bnd is early enough to make $T^{\leq \mathsf{bnd}}$ efficient, then our sampling of $T^{\leq \mathsf{bnd}}$ is efficient. We abstract the central property in the following definition.

Definition D.33 (Continuously samplable). A runtime T is **continuously samplable** with overhead function $\mathsf{sampovhd}(k) = \mathsf{sampovhd}_\kappa(k)$, which quantifies the time for sampling T up to time $k \in \mathbb{N}_0 \cup \{\infty\}$; that is: $T^{\leq k}$ can be sampled in $\mathsf{sampovhd}(k)$ steps for all k. More concretely, there is a subroutine $\mathsf{Sample}_T(\kappa, k)$ with output distributed as $T^{\leq k}$ and runtime (strictly) bounded by $\mathsf{sampovhd}(k)$.

We will not specify the overhead $\mathsf{sampovhd}(k)$ and assume it to be "small enough" (e.g. $O(k\mathsf{polylog}(k))$).[83] In particular, for runtimes induced by algorithms, *sample and emulation overhead essentially coincide if one samples by emulation.* Hence emulation overhead must be small enough to work with the runtime class in question.

Notice that continuous samplability is not tied to any runtime classes per se. In particular, it does not imply efficient samplability without further assumptions.

Example D.34. Any runtime which is induced by an algorithm is continuously samplable. Including runtimes of inefficient algorithms.

Now, we show that for two *continuously samplable* runtimes T, S, where $T \in \mathcal{T}$ (i.e. T is efficient), *oracle-$\mathcal{T}$*-time (in)distinguishable and oracle-*included* $\mathcal{T}$-time (in)distinguishable coincide. This, finally, lets us relate the triple-oracle indistinguishability and standard indistinguishability (under repeated sampling).

[83] For PPT, $\mathsf{sampovhd}(\mathsf{poly}_1(\kappa)) \leq \mathsf{poly}_2(\kappa)$ would be good enough. For EPT, emulation requirements are stricter, since runtime may explode under squaring. Interestingly, we reduce only to, and only require, strict algebra-bounded times. Thus, the results in this section do not run into problems with expectation.

Lemma D.35. *Suppose that $\mathcal{B}$ is a bound algebra and $\mathcal{T}$ is $\mathcal{B}$-tailed. Suppose that $T \in \mathcal{T}$. Let S be any runtime. Furthermore, suppose that $\mathcal{D}$ is a $\mathcal{T}$-time (triple-oracle) distinguisher with advantage $\geq c$ (infinitely often).*

Then there is a distinguisher $\mathcal{A}$ with advantage $\geq \frac{c}{4}$ (infinitely often) and (a priori) strict oracle-included $\mathcal{B}$-time. More concretely, $\mathsf{time}_{\mathcal{A}}(\mathcal{A}^{\mathcal{O}_0,\mathcal{O}_1,\mathcal{O}^*}) \leq \mathsf{time}_{\mathcal{D}}(\mathcal{D}^{\mathcal{O}_0,\mathcal{O}_1,\mathcal{O}^*})$ up to overhead for emulation and computing the strict bound $\mathsf{bnd}(\kappa)$.

Suppose $\mathcal{A}$ is a distinguisher with runtime strictly bounded by bnd and oracle queries strictly bounded by $\mathsf{bnd}_{\mathrm{query}}$ Suppose T and S are continuously samplable. Then there is an $\mathcal{A}'$ which emulates $\mathcal{O}_0$ and $\mathcal{O}_1$ up to $\mathsf{bnd}_{\mathrm{trunc}} \in \mathcal{B}$ "steps", i.e. emulating $T^{\leq \mathsf{bnd}_{\mathrm{trunc}}}$, $S^{\leq \mathsf{bnd}_{\mathrm{trunc}}}$. By construction, $\mathcal{A}$ is strict $\mathcal{B}$-time with runtime bound roughly $\mathsf{bnd} + 16\mathsf{bnd}_{\mathrm{query}} \cdot \mathsf{bnd}_{\mathrm{trunc}}$ (up to overheads) and advantage at least $\frac{c}{16\mathsf{bnd}_{\mathrm{query}}}$ (infinitely often).

It is vital that $T \in \mathcal{T}$, and hence *efficiently* continuously samplable.

Proof. This first part of the claim is proven analogously to the "standard reduction to PPT", Corollary C.9. More concretely: Suppose $\mathcal{O}^* = \mathcal{O}_0$ and consider $\mathcal{D}$. Since $c \in \mathcal{B}$, there exists for some efficiently computable $\mathsf{bnd} \in \mathcal{B}$ because $\mathcal{T}$ is $\mathcal{B}$-tailed. The truncation $\mathcal{A}$ of $\mathcal{D}$ has output with statistical distance at most $\frac{1}{4}c$ (infinitely often). For $\mathcal{O}^* = \mathcal{O}_1$, we either obtain a statistical distance of $\frac{1}{2}c$, or a distinguishing of $\mathcal{O}_0$ and $\mathcal{O}_1$ which uses the runtime statistic $\mathbb{P}(S > \mathsf{bnd}) > \frac{1}{2}c$ of $\mathcal{D}$ as distinguishing statistic, just as in Corollary C.9. In any case, we obtain $\mathcal{A}$ as claimed.

The second part of the claim follows by definition of continuously samplable and efficiency of T. Namely, let $\mathsf{bnd}_{\mathrm{trunc}}$ so that $\mathbb{P}(T > \mathsf{bnd}_{\mathrm{trunc}}) \leq \frac{c}{16\mathsf{bnd}_{\mathrm{query}}}$, where bnd and $\mathsf{bnd}_{\mathrm{query}}$ are strict bounds for runtime and number of queries of $\mathcal{A}$. Since $\mathcal{T}$ is $\mathcal{B}$-tailed and $T \in \mathcal{T}$, an efficiently computable $\mathsf{bnd}_{\mathrm{trunc}} \in \mathcal{B}$ exists. Suppose that $\mathbb{P}(S > \mathsf{bnd}_{\mathrm{trunc}}) \leq \frac{c}{8\mathsf{bnd}_{\mathrm{query}}}$. Otherwise, using this distinguishing statistic yields $\mathcal{A}'$ with advantage $\frac{c}{16\mathsf{bnd}_{\mathrm{query}}}$ Now let $\mathcal{A}'$ run $\mathcal{A}$ with the each oracle call to $\mathcal{O}_b$ emulating up to $\mathsf{bnd}_{\mathrm{trunc}}$ "steps" via continuous sampling. The probability that an oracle call returns `timeout` is bounded by $\mathsf{bnd} \cdot \frac{c}{8\mathsf{bnd}_{\mathrm{query}}} = \frac{c}{8}$. In that case, $\mathcal{A}'$ returns a random guess. Thus, $\mathcal{A}'$ has advantage $\frac{c}{8}$ □

Lemma D.35 reduces triple-oracle distinguishing to distinguishing w.r.t. repeated samples. It has no requirements on the advantage c of the distinguisher $\mathcal{D}$ and preserves the number of challenge queries in $\mathcal{A}$ and $\mathcal{A}'$. Thus, we can first use a hybrid argument in the triple-oracle setting, reducing to a single challenge query. Then apply Lemma D.35. This finally yields the equivalence we wanted.

Corollary D.36 (Equivalence of triple-oracle and standard indistinguishability). *Let $\mathcal{B}$ be a bound algebra and $\mathcal{T}$ be $\mathcal{B}$-tailed. Let $T \in \mathcal{T}$ and let S be an arbitrary runtime. Then T and S are* triple-oracle *distinguishable with non-$\mathcal{B}$-negligible advantage, if and only if T and S are* standard *distinguishable with non-$\mathcal{B}$-negligible advantage. (There is $\mathcal{B}$-factor of loss involved in the reduction.)*

Finally, we stress that Corollary D.36 is a very loose reduction.

E. ⋆ Technical asides

E.1. Section 2

E.1.1. General comments

Remark E.1 (Unary or binary encodings of runtime). The use of unary encodings in cryptography is more of a compatibility "hack" than a necessity. On the on hand, it is often a convenient "hack". On the other hand, one has to keep in mind this implicit restriction, and for *statistical* indistinguishability a distinction between binary or unary data is superfluous.

If runtimes were enocded in unary, rather than binary, this would implicity restrict access to a prefix (i.e. a cutoff) which can be efficiently computed. This does not affect our results at all, since our distinguishers and proofs rely on exactly that. Nevertheless, we use binary encoded runtimes and explicit runtime restrictions.

Remark E.2 (Non-uniformity and efficiency). Non-uniform (in)security can affect whether an algorithm is considered efficient or not: Suppose there exists an *unkeyed* collision-resistant hash function. An algorithm's runtime might explode when given colliding inputs. Thus, in the uniform setting, the probability for runtime explosion is negligible, but with non-uniform advice, collisions are trivial. Hence efficiency and security depend on (non-)uniformity. On the other hand, since our results and proofs make only timed black-box use of (adversarial) algorithms, they work in both computational models (with suitable adaptions).

Remark E.3 (A priori CPPT). For PPT (and CPPT), the distinction of *a priori PPT* and (a posteriori) PPT is often insignificant. For example, any CPPT algorithm A with virtual runtime bound poly can be truncated to poly steps, giving a (statistically) indistinguishable a priori PPT algorithm A'. Thus, we can usually assume *a priori* PPT for PPT adversaries.

E.1.2. Non-uniform security

Our proposed notion of non-uniform security is still *probabilistic*. More concretely, we propose in Appendix A to give a probabilistic machine tape-like access to an infinite non-uniform advice string. Usually, non-uniform adversaries are modelled as a priori polynomial time *deterministic* algorithms with advice, or equivalently, polynomial size circuit families. For indistinguishability notions, allowing probabilistic algorithms is typically irrelevant: By standard reductions, a priori PPT adversaries suffice, and so does a priori polynomially bounded advice. By coin-fixing, i.e. fixing the optimal advice and optimal adversarial randomness, one achieves a deterministic a priori non-uniform polynomial time adversary with advantage which is lower-bounded by that of the original (probabilistic) adversary.

Example E.4. For oracle-distinguishing, we saw in Corollary 4.2, that CEPT distinguishers are no better than a priori PPT distinguishers. For an a priori PPT distinguisher $\mathcal{D}$, it is easy to see that fixing optimal coins yields a deterministic distinguisher $\mathcal{D}'$ with advantage lower-bounded by the advantage of $\mathcal{D}$.

Unfortunately, technical details regarding *preservation of efficiency* still enforce the use of input *distributions*. More concretely, by runtime squaring, there are simulators which are efficient for any input distribution with *strictly* polynomial input size, but become inefficient for distributions with *expected* polynomial input size. see Example E.26. Thus, an equivalence with the standard setting of non-uniform security is only guaranteed if security with size-guarding is considered, see Appendix E.4.3. While size-guarded security is a natural notion, imposing it when it is not needed is wasteful.

E.1.3. Oracle indistinguishability and games

We recall a (folklore) conversion between game-based notions and oracle-indistinguishability. Many cryptographic assumption can be cast as (efficient or inefficient) games, in which an adversary interacts with a *challenger* $\mathcal{C}$ (specifying the *experiment* or *game*), and at the end of the interaction, the challenger outputs a verdict $\texttt{win}/\texttt{lose}$ (or $1/0$). A hardness assumption is an upper bound for $\mathbb{P}(\mathsf{out}_{\mathcal{C}}\langle\mathcal{A},\mathcal{C}\rangle = \texttt{win})$, e.g. negl for one-wayness or $\frac{1}{2} + \mathsf{negl}$ for IND-CPA, where negl depends on $\mathcal{A}$.

It is generically possible to recast such games as oracle-indistinguishability assumptions: Let $\mathcal{O}_b$ for $b = 0, 1$ be defined as follows. The oracle acts as the game, until the verdict is output. If the verdict is $\texttt{win}$, then $\mathcal{O}_b$ sends b to the adversary. If the verdict is $\texttt{lose}$, then $\mathcal{O}_b$ sends $\bot$ instead. A straightforward calculation shows

$$\mathbb{P}(\mathsf{out}_{\mathcal{C}}\langle\mathcal{A},\mathcal{C}\rangle = \texttt{win}) = \mathbb{P}(\mathcal{D}^{\mathcal{O}_1} = 1) - \mathbb{P}(\mathcal{D}^{\mathcal{O}_0} = 1)$$

where $\mathcal{D}$ is derived from $\mathcal{A}$ by emulating $\mathcal{A}$ until the verdict, and then outputting b (if $\mathcal{A}$ won) or guessing randomly (if $\perp$ was received). Conversely, given $\mathcal{D}$, one defines $\mathcal{A}$ by acting like $\mathcal{D}$ (until the experiment ends). Since information-theoretically, $\mathcal{D}$ obtains learns about b only when it wins the game, $\mathcal{A}$'s success probability is at least that of $\mathcal{D}$. Applying the reverse conversion yields an $\mathcal{D}'$ with advantage equal the probability that $\mathcal{A}$ wins. In other words, both formalizations are equivalent (in any setting that allows the conversion, which encompasses any sensible setting).

The reverse transformation transforms oracle-indistinguishability into a "bit-guessing" experiment. Since the success probability in the experiment is compared to $\frac{1}{2}$, the *advantage* is defined by twice the success probability (so as to coincide with the distinguishing advantage).

E.2. Section 3

E.2.1. More on CEPT

Remark E.5 (Non-uniform advice). Observe that the proofs in this section used efficient approximation of (suitably close truncated) distributions as their central tool. This can be effectively trivialized by assuming non-uniform advice, which allows exponentially precise approximations of the truncated distributions, or even simpler, encoding the optimal distinguishing decision for each of the possible samples. Thus, non-uniform advice can replace sampling access, and triple-oracle and standard indistinguishability of runtime coincide (if at least one runtime is efficient).

Remark E.6 (Generalizations). The results in this section relied effectively on tail bounds and very natural properties of runtime classes (e.g., if $T \in \mathcal{T}$ and $S \overset{d}{\leq} T$, then $S \in \mathcal{T}$), and . They extend to any setting, where existence of suitable tail bounds is guaranteed. In Appendix D, we discuss this more formally, and define *algebra-tailed* runtime classes, to which the results extend in a suitable manner.

We also note that, as already seen in Corollary 3.10, it is not necessary that runtime are induced by algorithms, just that they can be approximated up to any polynomial precision in polynomial time. This can be viewed as the central requirement in other proofs and results as well, but it is tedious to formulate and seems only useful in special occasions, e.g., Lemma 4.13.

E.2.2. More on timeout oracles

We continue the discussion of Section 3.3 with a specific application to sequential composition.

Lemma E.7 (Sequential timeout oracles). *Let* A *be an interactive algorithm and* $\mathcal{O}_1, \mathcal{O}_2$ *be a (probablistic) timeful oracles. Suppose* $\mathcal{O}$ *is the sequential composition of* $\mathcal{O}_1$ *and* $\mathcal{O}_2$*. That is,* $\mathcal{O}$ *first runs* $\mathcal{O}_1$*. At some point,* $\mathcal{O}_1$ *terminates with input* y *for* $\mathcal{O}_2$*, which is passed to* $\mathcal{O}_2$ *as initial input. Now,* $\mathcal{O}$ *continues to run* $\mathcal{O}_2(y)$*. The results of Lemma 3.12 hold for* $\mathcal{O}$*, where* $\Omega_{\mathcal{O}} = \Omega_{\mathcal{O}_1} \times \Omega_{\mathcal{O}_2}$*.*

Moreover the probability ε *for* `timeout` *decomposes as follows: Let event* $\mathcal{E}_{\texttt{timeout},1}$ *be the event for* `timeout` *while running* $\mathcal{O}_1$*. Let event* $\mathcal{E}_{\texttt{timeout},2}(y,t_1)$ *be the event for* `timeout` *while running* $\mathcal{O}_2$ *where* $\mathcal{O}_1$ *took* t_1 *steps to output* y*. Let* Y *and* T_1 *be the random variables for the output and number of steps of* $\mathcal{O}_1$*. Let* $\varepsilon_1 = \mathbb{P}(\mathcal{E}_{\texttt{timeout},1})$ *and let* $\varepsilon_2(y,t_1) = \mathbb{P}(\mathcal{E}_{\texttt{timeout},2}(y,t_1) \mid (Y,T_1) = (y,t))$*. Then*

$$\begin{aligned}
\varepsilon &= \mathbb{P}(\mathcal{E}_{\texttt{timeout}}) \\
&= \mathbb{P}(\mathcal{E}_{\texttt{timeout},1}) + \sum_{(y,t_1)} \mathbb{P}(\mathcal{E}_{\texttt{timeout},2}(y,t_1) \wedge (Y,T_1) = (y,t_1)) \\
&= \varepsilon_1 + \sum_{(y,t_1)} \varepsilon_2(y,t_1)\, \mathbb{P}((Y,T_1) = (y,t_1))
\end{aligned}$$

Lemma E.7 follows essentially from Lemma 3.12 and the fact that the runtimes of $\mathcal{O}_1$ and $\mathcal{O}_2$ sum to that of $\mathcal{O}$. For the decomposition, one argues as in the proof of Lemma 3.12, and uses that knowledge of (y,t_1) is good enough for the truncation construction, i.e. $\mathcal{O}_2'$ only needs to know the elapsed time in $\mathcal{O}_1'$ to "continue" the truncation by incorporating the steps of $\mathcal{O}_1'$. The proof is left to the reader.

Remark E.8. In the setting of Lemma E.7, it is also possible to "separate" $\mathcal{O}_1$ and $\mathcal{O}_2$ instead of treating them as one entity. That is, one can modify them separately, without telling $\mathcal{O}_2$ the runtime t_1 spent in $\mathcal{O}_1$. (Implicit access to t_1 is the only additional knowledge used in Lemma E.7.) Concretely, assuming total virtuality ε, one can apply Lemma 3.12 for an ε-quantile cutoff to $T_1 = \mathsf{time}_{\mathcal{O}_1}(\mathsf{A}^{\mathcal{O}_1,\mathcal{O}_2})$ to obtain $\mathcal{O}_1'$, and then to $T_2 = \mathsf{time}_{\mathcal{O}_2}(\mathsf{A}^{\mathcal{O}_1',\mathcal{O}_2})$ to obtain $\mathcal{O}_2'$. (For this, note that the virtualities of T_1 and T_2 are certainly bounded by ε.) Together, $(\mathcal{O}_1', \mathcal{O}_2')$ have overall timeout probability of (at most) 2ε and the expected runtime is $t + O(1)$. Unfortunately, the timeout probability of this construction is larger than ε. Except, if y fixes t_1, i.e. if there is a function f such that $f(y) = t_1$, then $\mathcal{O}_1'$ and $\mathcal{O}_2'$ are "separated" by construction (also in Lemma E.7).

E.3. Section 4

E.3.1. Precision-tightness tradeoff

Most of our results, are very precise in handling the runtime of algorithms, approximating them, and often show that the *distribution* of the runtime only changes negligibly. For example, we proceeded like this in Lemmas 3.6, 3.8, 4.1 and 4.7 and Corollaries 3.9 and 3.10. However, this precision is often unnecessary. Yet, for the sake of simplicity and self-containedness, we always reduced to polynomial support by truncation, and we used a very naive closeness test (see Remark E.9 below).

Observe that approximation becomes more expensive (and less tight), the larger the *support* of the distribution is. To improve the state of affairs, one can "coarsen" the time-steps in consideration. Concretely, let $f(x) = 2^{\lceil \log_2(x) \rceil}$, that is, $f(x)$ rounds x to the next power of 2 (and $0 \mapsto 0$). Let X by some positive-valued random variable X (e.g. a runtime). Then we have:

- $\mathbb{E}(X) \leq \mathbb{E}(f(X)) \leq 2 \cdot \mathbb{E}(X)$, since $x \leq f(x) < 2x$.
- $\mathsf{card}(\mathsf{supp}(f(X))) \leq \log_2(\mathsf{card}(\mathsf{supp}(X)))$

In particular, consider an EPT or CEPT runtime T, and assume that $T_\kappa \leq 2^\kappa$. Then $\mathsf{card}(\mathsf{supp}(f(T))) \leq \kappa$, whereas $\mathsf{card}(\mathsf{supp}(T^{\leq \mathsf{poly}})) = \mathsf{poly}(\kappa)$, for any polynomial poly. Thus, the coarser $f(T)$ is, the more efficient approximation (and closeness testing) becomes. However, precision is lost in the time-domain. Generally, this is irrelevant, since efficiency is not affected at all by this uncertainty.

Remark E.9 (Closeness testing). Our analysis was not geared towards optimal tightness and precision to begin with. And, for simplicity, we actually never made explicit, that we often merely require *closeness tests* for distributions (see Appendix C.4 for a reminder). We used a naive approximation of distributions as our closeness test, but there are much better alternatives. However, even for the (optimal) closeness tester of [Cha+14], its precision depends on the size of the support. Hence, using [Cha+14] further improves in tightness, is overall is still very loose.

E.3.2. Constructive reductions

While we prove all of our results in a uniform complexity setting, hence avoid non-uniformity, almost none of our proofs are constructive. In fact, almost all reductions depend on polyomial bounds, which exist but are not computable in general. We will use following rough notion of constructive reductions.

Remark E.10 (Constructive reduction). A reduction is *constructive*, if there is a (universal) efficient algorithm, which is given the code of an adversary, and produces (the code of) an adversary against some underlying assumption.

For simple results, such as the standard truncation argument, one can give constructive proofs in restricted cases. We sketch how these can be obtained.

Remark E.11. A weakening of the standard reduction (Lemma 4.1 and Corollary 4.2) can be proven constructively. We sketch how to transform the distinguisher $\mathcal{D}$ into a distinguisher $\mathcal{A}$ (where $\mathcal{D}$, $\mathcal{O}_0$, $\mathcal{O}_1$ are as the standard reduction). For the constructive $\mathcal{D}$, we have to merge the two distinguishers $\mathcal{A}_1$

(which uses `timeout`) and $\mathcal{A}_2$ (which uses the output of $\mathcal{D}$). One problem is, that the use of "unsigned" advantage (i.e. absolute values instead of the (signed) differences) does not work well constructively, since we cannot "know" the signs. Thus, we use standard sign-correction techniques. Consider some distinguisher $\mathcal{B}$. To sign-correct, the reduction runs $\mathcal{B}^{\mathcal{O}_0}$ and $\mathcal{B}^{\mathcal{O}_1}$, emulating $\mathcal{O}_0$ and $\mathcal{O}_1$, to get output (and runtime statistics). Using this, one corrects the output of $\mathcal{B}^{\mathcal{O}^*}$, so that $\mathbb{P}(\mathcal{B}^{\mathcal{O}_b} = b) \geq \frac{1}{2}$ for $b = 0, 1$. This gives us a first restriction: We require that emulating $\mathcal{B}^{\mathcal{O}_b}$ is efficient for $\mathcal{O}_0$ and $\mathcal{O}_1$. In particular, $\mathcal{O}_0$ and $\mathcal{O}_1$ will have to be efficient in some sense.

Consider some efficiently samplable runtime distribution S (to be chosen later). Our distinguisher $\mathcal{A}$ works as follows:

- Pick $s \leftarrow S$.
- Set $\mathcal{B} := \mathcal{D}^{\leq s}$.
- Emulate $\mathcal{B}^{\mathcal{O}_b}$ to obtain runtimes t_b and outputs out_b for $b = 0, 1$. (We assume $out \in \{0, 1\}$.)
- Emulate $\mathcal{B}^{\mathcal{O}^*}$ and obtain (t^*, out^*) and output a guess as follows:
  - If $t_b = \texttt{timeout} \neq t_{1-b}$: If $t^* = \texttt{timeout}$ return b, else return $1 - b$.
  - If $t_0, t_1 \neq \texttt{timeout}$ and $out_0 \neq out_1$: Return $out^* \oplus (out_0 \oplus out_1 \oplus 1)$.
  - Else: Return 0.

By construction, $\mathcal{A}$ is a merge of a distinguisher based on `timeout` probabilites and a distinguisher based on the output of $\mathcal{D}$. Both are sign-corrected, so the advantages actually add up (and don't cancel out). To guarantee non-negligible advantage, when $\mathcal{D}$ has non-negligible advantage, we must ensure that the truncation $\mathcal{B}$ of $\mathcal{D}$ at s gives $\mathcal{B}$ enough runtime with polynomial probability. For this, one can use any distribution S which is EPT and has fat tails, e.g. the distribution obtained from normalizing $\sum_{n=1}^{\infty} n^{-3}$. However, for $\mathcal{A}$ to be efficient overall, we additionally require the time spent to emulate $\mathcal{O}_0$ and $\mathcal{O}_1$ does not make $\mathcal{A}$ inefficient (for the varying $s \leftarrow S$). One possibility is to restrict to a priori PPT $\mathcal{O}_0$, $\mathcal{O}_1$, but less strict choices are possible.[84] All in all, this yields a weaker, less tight, more restricted, but constructive, form of the standard reduction.

E.3.3. Relative efficiency for mappings of systems

The definition of weak relative efficiently (Definition 4.5) does not strictly capture our actual application. Indeed, a simulator takes as *input* an adversary, i.e. a system/oracle (or an algorithm/code), and *acts* as (or *outputs*) a new system. Hence, (the existence of) a simulator is actually a mapping from admissible adversaries to simulators. This is quite obvious for universal (resp. bb-rw) simulation, where the code (resp. bb-rw access) are clear "inputs". Since the simulator is independent of the input generation or the distinguisher, i.e. of the "distinguishing environment", it is also evident that $\mathsf{Sim}(\mathsf{code}(\mathcal{V}^*))$ has an input and output interface. The input is $(x, w, \textit{aux})$, the output is the output of $\mathcal{V}^*$. While Sim discards w, it is necessary so that $\mathsf{Sim}(\mathcal{V}^*)$ and $\langle \mathcal{P}, \mathcal{V}^* \rangle$ offer the same interface to the environment. We sketch a general definition of the above.

Definition E.12. A **mapping** of systems (or algorithms) to systems (or algorithms) is a function $\mathrm{F}\colon C_I \to D_J$ with maps system (or algorithms) with interface I to systems (or algorithms) with interface J.

With this, we can define when a mapping G is weakly efficient relative to a mapping F. This generalizes Definition 4.5, which can be recovered from the constant mappings $\mathrm{F} = \mathsf{A}$, $\mathrm{G} = \mathsf{B}$.

Definition E.13. Let $\mathrm{F}, \mathrm{G}\colon C_I \to D_J$ be mappings. We say G is **weakly $(\mathcal{T}, \mathcal{S})$-efficient relative to** F *w.r.t. (implicit) runtime classes $\mathcal{T}$, $\mathcal{S}$*, if for all distinguishing environments $\mathcal{E}$,

$$\forall \mathcal{A} \in C_I\colon \quad \mathsf{time}_{\mathcal{E}+\mathrm{F}(\mathcal{A})}(\langle \mathcal{E}, \mathrm{F}(\mathcal{A}) \rangle) \in \mathcal{T} \implies \mathsf{time}_{\mathcal{E}+\mathrm{G}(\mathcal{A})}(\langle \mathcal{E}, \mathrm{G}(\mathcal{A}) \rangle) \in \mathcal{S}$$

[84] As seen with the runtime squaring problem and expected polynomial size inputs, relatively stringent requirements on $\mathcal{O}_0$, $\mathcal{O}_1$ (or size-guarding, c.f. Appendix E.4.5) seem necessary in general.

Unfortunately, Definition E.13 is not strong enough to be used in reductions, which is why we reserved the specification "*weakly*" for Definitions 4.5 and E.13 (More precisely, we cannot prove or refute that it is (not) strong enough.) Therefore, we rely on following strengthening, where, the runtime classes $\mathcal{T}$ and $\mathcal{S}$ are from $\{\mathcal{PPT}, \mathcal{EPT}, \mathcal{CPPT}, \mathcal{CEPT}\}$, and they decide whether strict or expected time is measured.[85]

Definition E.14. Let F, G, etc. be as in Definition E.13. We say that G is **$(\mathcal{T}, \mathcal{S})$-efficient relative** to F with **runtime tightness** $(\mathsf{poly}_{\mathsf{time}}, \mathsf{poly}_{\mathsf{virt}})$, if: For *all timeful* environments $\mathcal{E}$ and all $\mathcal{A} \in C_I$, if $\mathsf{time}_{\mathrm{F}(\mathcal{A})}(\langle \mathcal{E}, \mathrm{F}(\mathcal{A})\rangle)$ is virtually strict/expected (t_0, ε_0)-time, then $\mathsf{time}_{\mathrm{F}(\mathcal{A})}(\langle \mathcal{E}, \mathrm{G}(\mathcal{A})\rangle)$ is virtually strict/expected (t_1, ε_1)-time, with $t_1(\kappa) \leq \mathsf{poly}_{\mathsf{time}}(\kappa) t_0(\kappa)$ with $\varepsilon_1(\kappa) \leq \mathsf{poly}_{\mathsf{virt}}(\kappa)\varepsilon_0(\kappa)$ (for all κ).

We stress that Definition E.14 is unconditional w.r.t. the environment, i.e. uses timeful environments, and that the tightness bounds depend *only on* κ. Mixing strict and expected time (i.e. $\|\cdot\|_\infty$ and $\|\cdot\|_1$) in Definition E.14 is possible and useful. For example when strict PPT protocols and adversaries are handled by EPT simulators.

(Weak) Relative efficiency is transitive in the obvious sense. Lastly, we mention that there are obvious variations of relative efficiency, e.g. relative efficiency w.r.t. environments in a class $\mathfrak{E}$ of admissible environments with restriction beyond runtime.

E.4. Section 5

In this section, we discuss some technical asides. It should be skipped on a first reading.

E.4.1. Definitional choices

Remark E.15 (The adversary's view). We did not use the *view* of the adversary to define zero-knowledge for a reason. The usual definition of a view consists of input, randomness, and received messages. This conflates different complexities, e.g. randomness and space, and thus prevents *strict polynomial space simulation*, which may be of interest. For example, the simulator for $\mathrm{G3C}_{\mathrm{GK}}$ uses expected polynomial randomness and space, even if $\mathcal{V}^*$ requires only strict polynomial space (since $\mathsf{bbrw}(\mathcal{V}^*)$ chooses and fixes (i.e. remembers) the random coins) A slightly "improved" simulation requires only strict polynomial space. For more discussion on interaction of (bb-rw) emulation with strict versus expected space and randomness complexity, see Remark A.6.

Remark E.16. By a standard reduction to PPT (Corollary 4.2), we can assume that $\mathcal{D}$ is a priori PPT in Definition 5.2. A formulation of zero-knowledge via indistinguishable ensembles,

$$\begin{aligned} &\{(x, \mathit{aux}, \mathit{out}, \mathit{state}) \mid (x, w, \mathit{aux}, \mathit{state}) \leftarrow \mathcal{I}(\kappa); \mathit{out} \leftarrow \mathsf{out}_{\mathcal{V}^*}\langle \mathcal{P}(w), \mathcal{V}^*(\mathit{aux})\rangle(x)\}_\kappa \\ \overset{c}{\approx} &\{(x, \mathit{aux}, \mathit{out}, \mathit{state}) \mid (x, w, \mathit{aux}, \mathit{state}) \leftarrow \mathcal{I}(\kappa); \mathit{out} \leftarrow \mathsf{Sim}(\mathsf{code}(\mathcal{V}^*), x, \mathit{aux})\}_\kappa \end{aligned}$$

is then equivalent to Definition 5.2.

Remark E.17. There are other formulations of zero-knowledge, which can be obtained by swapping the order of the quantifiers. To recover the "usual notions" (that is universal quantification over the inputs), $\mathcal{I}$ should be instantiated by a non-uniform machine which regurgitates its advice.

(Timed) Black-box simulator: Timed bb-rw access to $\mathcal{V}^*$. Most common form of simulation.
Universal simulator: $\exists\mathsf{Sim}\forall\mathcal{V}^*\forall\mathcal{I}\forall\mathcal{D}$. Typical form of non-black-box simulation, e.g. in [Bar01].
Existential simulator: $\forall\mathcal{V}^*\exists\mathsf{Sim}\forall\mathcal{I}\forall\mathcal{D}$. Typical definition of zero-knowledge, e.g. in [Gol01].

We see in Appendix E.4.4 below that existential simulation and universal simulation are equivalent for *auxiliary input* zero-knowledge for *a posteriori time*.

There are also less common, weaker notions, such as *distributional simulation* (roughly "$\forall\mathcal{V}^*\forall\mathcal{I}\exists\mathsf{Sim}\forall\mathcal{D}$"), *weak simulation* (roughly "$\forall\mathcal{V}^*\forall\mathcal{D}\exists\mathsf{Sim}\forall\mathcal{I}$"), *weak distributional simulation* (roughly "$\forall\mathcal{V}^*\forall\mathcal{D}\forall\mathcal{I}\exists\mathsf{Sim}$"),

[85] This generalizes to other norms besides $\|\cdot\|_1$ and $\|\cdot\|_\infty$ as measures of efficiency.

see [Dwo+03; CLP15]. Likewise, there are stronger notions, such as precise zero-knowledge [MP06; DG12] where simulation and real execution must have similar runtime per execution. We have not pursued an adaption to CEPT for these notions.

Remark E.18 (Non-uniform zero-knowledge). When considering non-uniformity, there are different options. In any case, $\mathcal{I}$ and $\mathcal{D}$ (equivalently $\mathcal{E}$) should be non-uniform. Now, $\mathcal{V}^*$ can be uniform or non-uniform. For Sim, there are two similar options. One is to insists on uniform simulation, in the sense that the advice Sim is given the same as the advice of $\mathcal{V}^*$. The other is to allow an existential non-uniform Sim, whose advice may arbitrarily depend on $\mathcal{V}^*$ and $adv_{\mathcal{V}^*}$. Goldreich [Gol01] calls the latter a "fully non-uniform" formulation of zero-knowledge, and argues that the former is preferable. These choices do not affect our results based on black-box simulation. However, the existence of a universal $\mathcal{V}_{\text{univ}}$ is unclear, if $\mathcal{V}^*$ has access to non-uniform advice.

Remark E.19 (Effects of "(non-)environmental" distinguishing). Let us consider the effect of "non-environmental" environments $(\mathcal{I}, \mathcal{A}, \mathcal{D})$, i.e. quantification over $\mathcal{I}$ which output no *state* (i.e *state* $= \bot$ always). In this case case, Sim has the same information that is available to $\mathcal{D}$, whereas in Definition 5.2 $\mathcal{I}$ can pass information to $\mathcal{D}$ directly.

Since a bb-rw simulator has no access to *aux*, *aux* can be used to pass "direct" messages to $\mathcal{D}$, thus both notions coincide in this case. It seems plausible, that simulators which do not "reverse-engineer" $\mathcal{V}^*$ and *aux* and are "non-environmentally" secure are also "environmentally" secure, i.e. we know of no counterexamples even for non-black-box simulators. Intuitively, $\mathcal{E}$ and $\mathcal{D}$ may share a "key" (e.g. as non-uniform advice or hardwired), and use a one-time pad to "encrypt" messages which are passed from $\mathcal{E}$ to $\mathcal{D}$. It is easy to see that, if Sim is not secure, then there is a (sequence of) "keys", such that the advantage of a suitable non-environmental $(\mathcal{E}', \mathcal{D})$ is at least that of $\mathcal{E}$ (infinitely often), if the "key" is long enough to one-time pad "encrypt" the state of $\mathcal{E}$ passed between input generation and distinguishing. (Summing the advantage over hardwired or non-uniform key k for $(\mathcal{E}'_k, \mathcal{D}_k)$ over all possible $\mathsf{poly}(\kappa)$-bit keys, weighted by $2^{-\mathsf{poly}(\kappa)}$ is exactly the advantage of $\mathcal{E}$. The claim follows.) Thus, in a uniform model, constant size messages can be passed to $\mathcal{D}$ without affecting security, and in a non-uniform model, polynomial size messages can be passed.

Remark E.20 ("Environmental" distinguishing and non-uniformity). The additional output *state* of $\mathcal{I}$ in Definition 5.2 essentially makes $(\mathcal{I}, \mathcal{D})$ into a stateful distinguishing "environment". This is a visible difference from the direct translation of non-uniform zero-knowledge and [Gol93]

In a non-uniform CEPT setting, Definition 5.2 does coincide with the standard definition, if $\mathcal{I}$ and $\mathcal{D}$ have non-uniform advice. To see this, observe that we can assume that successful $\mathcal{I}$ and $\mathcal{D}$ are a priori PPT, and by coin-fixing (i.e. fixing the optimal coins for $\mathcal{I}$ and $\mathcal{D}$ in the advice), they are deterministic. Now, *state* can simply be included in the non-uniform advice of $\mathcal{D}$ as well. Hence, in this non-uniform PPT setting, the notions are equivalent.

More generally, if a successful $\mathcal{I}$ only chooses (x, aux) of strictly polynomially bounded size, then there is an optimal choice and adaptions of the coin-fixing argument work. By Example E.26, we know that for covering simulation efficiency, we cannot restrict to such strictly polynomially bounded $(x, w, aux, state)$. Perhaps, this can be salvaged this somewhat. However, we find such "non-uniformity hacks" very unsatisfactory (even if they work). They result in "less robust" definitions (which is the reason we had to adapt the definitions in the first place).

Remark E.21 (Simulation tightness and inefficient provers). In Definition 5.2, we compare the runtime of a simulator with the runtime of $\mathcal{V}^*$ *and* $\mathcal{P}$. We do so, because efficiency (and tightness) of a simulation should be related to efficiency of the real execution.[86] Alternatively, we could compare $\mathsf{time}_{\mathcal{V}^*+\mathcal{P}}(\ldots)$ and $\mathsf{time}_{\mathsf{Sim}}(\ldots)$. This is equivalent, if the completed system with $\mathcal{I}, \mathcal{D}$ was efficient.[87]

[86] This entails some technical artefacts, e.g. a prover may be badly behaved for invalid inputs, e.g. not halting. The complexity class for "good protocols" should be robust and prevent such behaviour.

[87] However, $\mathsf{time}_{\mathcal{V}^*+\mathcal{P}}(\ldots)$ and $\mathsf{time}_{\mathsf{Sim}}(\ldots)$ are more suitable for analyzing the tightness of simulation, which we did not define (since we do not have a perfectly convincing definition).

By viewing $\mathcal{P}$ as timeful and setting its runtime to the length of messages sent by it (which is the minimal consistent choice for time), Definition 5.2 extends to inefficient provers. However, technical artefacts occur, e.g. if a simulation must run in quadratic time, then inputs which are *expected* polynomial size, can cause runtime explosions, and therefore the proof system is not zero-knowledge. If such problems occur, they can usually be circumvented by resorted to size-guarded security, which ensures that inputs are strictly polynomially bounded.[88]

Remark E.22 (Precomputation and different complexity classes). Non-uniform advice is often motivated as a means to strengthen attacks and allow arbitrary (even uncomputable) "precomputation". However, the practical meaning of non-uniformity is questionable [KM13]. Fortunately, precomputation neatly fits into our model by using different complexity classes for input generators (and possibly distinguishers or environments). This can be applied to our definitions of (sequential) zero-knowledge, but was omitted for the sake simplicity.

E.4.2. Motivating sequential zero-knowledge

Often, e.g. in [Gol93], only sequential repetition is considered for zero-knowledge. That is, the same pair $(x, w) \in \mathcal{R}$ is used in multiple rounds of the interactive argument. The (stateful) adversary $\mathcal{V}^*$ engages in these multiple rounds until it produces some output. To obtain zero-knowledge for such a sequential repetition, the core property is the *auxiliary input*. Construct from $\mathcal{V}^*$ a new $\mathcal{V}'$ which simply executes the code of $\mathcal{V}^*$ on *aux*. (More precisely, *aux* encodes either a state of $\mathcal{V}^*$ or, in the first invocation of $\mathcal{V}'$, it is the actual auxiliary input for $\mathcal{V}^*$.) Thus, each protocol now runs with the same adversary $\mathcal{V}'$, but different auxiliary inputs. Applying auxiliary input zero-knowledge and a hybrid argument, one sees that each interaction can be simulated.

In our setting, the universal adversary $\mathcal{V}_{\text{univ}}$ could be used directly (due to a posteriori time). Specifying and proving security of sequential repetition for *varying* statements is also possible along these lines. However, for *adaptive choices* of varying statements, it's not clear how to allow it with a "single" adversary $\mathcal{V}^*$. If $\mathcal{V}^*$ provides (x, w) then simulation becomes meaningless. Thus, we introduce an "environment", which models the use of the protocol. The environment $\mathcal{E}$ can make repeated calls to the protocol and choose the inputs (for both parties) and an adversary $\mathcal{V}^*$. Then $\mathcal{E}$ obtains the output of $\mathcal{V}^*$ (since $\mathcal{P}$ has no output). Finally, $\mathcal{E}$ produces some output. The "environment" does not participate in the computation. It is essentially a distinguisher for sequential real or ideal protocol executions.

Remark E.23. The weaker notion of sequential repetition, as sketched above, follows from auxiliary input zero-knowledge, even if the "environmental" $\mathcal{I}$ is not allowed not output *state*, e.g. if always *state* $= \bot$. This is in line with [Gol93], and uses that everything the adversary "knows" the simulator "knows". However, for two (or more) adaptive statements, the "environment" has "knowledge" which the simulator has not. In the classical definition (with classical PPT), the power of non-uniformity still allows to prove sequential composition. The non-uniform advice of $\mathcal{I}$ and $\mathcal{D}$ effectively establishes the "shared *state*" between $\mathcal{I}$ and $\mathcal{D}$.

E.4.3. Size-guards and size-guarded security

In our definition of zero-knowledge, due to fat tailed input distributions, a simulator is allowed almost no runtime overhead in $|x|$ compared with $\mathcal{P}$, i.e. if a prover is linear-time in $|x|$ the simulator must also be. In [KL08; Gol10], the simplification $\kappa = |x|$ is used, which alleviates this issue somewhat. We mirror that by explicitly *size-guarding* a protocol. This means that prover (and verifier) reject inputs which exceed the length of a (polynomial) size-guard $\mathsf{gd}(\kappa)$. Size-guards "decouple" efficiency of simulator and prover w.r.t. $|x|$, simplify efficiency arguments, but slightly weaken security.

Definition E.24 (Size-guarded zero-knowledge). We define **(uniform) zero-knowledge w.r.t. (input) size-guarded** security as follows: For any (monotone) polynomial bound gd (called **size-guard**), the

[88] An alternative "fix" is to prevent too efficient verifiers, e.g. using timelock puzzles.

derived protocol, where prover and verifier abort with $\texttt{gderr}$ on inputs (x, w) if $|x| > \mathsf{gd}(\kappa)$, is zero-knowledge (in the above sense).

The definition of **non-uniform** (size-guarded) zero-knowledge is analogous to the above, but $\mathcal{G}(\kappa)$ has access to an advice via an additional input interface.

Definition E.25. We define **(non-)uniform sequential zero-knowledge w.r.t. (input) size-guarded** security (see Definition E.24) as follows: For any polynomial size-guard gd, the derived protocol, where prover and verifier abort with $\texttt{gderr}$ on inputs (x, w) where $|x| > \mathsf{gd}(\kappa)$, is sequential zero-knowledge (in the above sense).

The use of (input) size-guarded security is meant to address certain situations, which we may want to consider secure, but cannot due to runtime artefacts.

Example E.26. Namely, suppose the simulator has quadratic runtime in the instance size $|x|$, whereas the prover's runtime is linear. Then, a problematic fat-tailed input distribution renders simulation inefficient. Consequently, without size-guarding, simulation must be "tight" in $|x|$. One technical artefact, partially mitigated by size-guards, is that very efficient provers make simulation harder. That is, by making a prover slower, e.g. adding a quadratic overhead, simulation becomes easier.

These problems may be of practical relevance: Given succinct argument systems, extraction comes with an overhead which is often superlinear. Such argument systems can be incompatible with CEPT under adversarial input distributions.

There are other means than size-guarding for solving the above problem. For example, one may quantify only over admissible adversaries. Indeed, adversaries which only send strictly polynomial size inputs are equivalent to size-guarded security.

See Appendix E.4.5 for more on size-guards.

E.4.4. The universal adversary $\mathcal{V}_{\text{univ}}$

The **universal adversary** $\mathcal{V}_{\text{univ}}$ is basically a virtual machine emulating some adversary, i.e. the input to $\mathcal{V}_{\text{univ}}$ is of the form $(\textit{code}, \textit{state}, \textit{aux})$, and $\mathcal{V}_{\text{univ}}$ continues execution of the code *code* in state *state*. The universal adversary $\mathcal{V}_{\text{univ}}$ contains the core hardness of simulation. An **existential** simulator is a simulator which may depend arbitrarily $\mathcal{V}^*$. The universal adversary shows that this arbitrary "existential" dependency on $\mathcal{V}^*$ does not weaken the notion of zero-knowledge. Thus, in Definition 5.2, we do not give up any power.

Lemma E.27 (Equivalence of existential and universal simulation). *Let $(\mathcal{P}, \mathcal{V})$ be an interactive argument system. If this argument system is zero-knowledge against $\mathcal{T}$-time designated adversaries w.r.t. to $\mathcal{S}$-time existential simulation, then it is zero-knowledge w.r.t. the universal simulator* Sim *defined as follows.*

Let $\mathcal{V}_{\text{univ}}$ be the universal adversary and $\mathsf{Sim}_{\text{univ}}$ be the existential simulator for $\mathcal{V}_{\text{univ}}$. Here, Sim *is defined by* $\mathsf{Sim}(\mathsf{code}(\mathcal{V}^*), x, \textit{aux})$ *emulating* $\mathsf{Sim}_{\text{univ}}(\mathsf{code}(\mathcal{V}_{\text{univ}}), x, (\mathsf{code}(\mathcal{V}^*), \textit{state}, \textit{aux}))$*, where state is the initial state of $\mathcal{V}^*$.*

Proof. First we define $\mathcal{G}_{\mathcal{V}^*}$, which samples $(x, w, \textit{aux}) \leftarrow \mathcal{G}$, and returns $(x, w, (\mathsf{code}(\mathcal{V}^*)), \textit{state}, \textit{aux})$, where *state* is the initial state of $\mathcal{V}^*$. Recall that for $(\mathcal{G}, \mathcal{V}^*)$, the simulator $\mathsf{Sim}(\mathsf{code}(\mathcal{V}^*), x, \textit{aux})$ runs $\mathsf{Sim}_{\text{univ}}(\mathsf{code}(\mathcal{V}_{\text{univ}}), x, (\mathsf{code}(\mathcal{V}^*), \textit{state}, \textit{aux}))$, which corresponds to $(\mathcal{G}_{\mathcal{V}^*}, \mathcal{V}_{\text{univ}})$. Moreover the real executions $\mathsf{Real}_{\mathcal{G},\mathcal{V}^*}$, and $\mathsf{Real}_{\mathcal{G}_{\mathcal{V}^*},\mathcal{V}_{\text{univ}}}$ are identical. Thus

$$\mathsf{Ideal}_{\mathcal{G},\mathsf{Sim}(\mathcal{V}^*)} \overset{d}{\equiv} \mathsf{Ideal}_{\mathcal{G}_{\mathcal{V}^*},\mathsf{Sim}_{\text{univ}}} \overset{c}{\approx} \mathsf{Real}_{\mathcal{G}_{\mathcal{V}^*},\mathcal{V}_{\text{univ}}} \overset{d}{\equiv} \mathsf{Real}_{\mathcal{G},\mathcal{V}^*}.$$

□

The upshot of the proof is, that an existential simulator cannot truly leverage its arbitrary dependency on $\mathcal{V}^*$. All the hardness of $\mathcal{V}^*$ might be in the *auxiliary input*, which Sim cannot depend upon. Lemma E.27 extends to the non-uniform setting and to size-guarding.

Caution E.28. We crucially relied on our *a posteriori* runtime notion. For other notions of runtime, Lemma E.27 may not hold! For example, if we assume *a priori* PPT algorithms, then $\mathcal{V}_{\text{univ}}$ cannot emulate every adversary $\mathcal{V}^*$, since $\mathcal{V}_{\text{univ}}$ must not exceed poly steps, for some *fixed* poly, whereas $\mathcal{V}^*$ may run (much) longer, say poly$+1$ steps. (There is a family $\mathcal{V}^n_{\text{univ}}$ with runtime bounds $\text{poly}_n(\kappa) = n\kappa^n$, so a morally equivalent result does hold.)

E.4.5. Size-guards

We recall the need for size-guards, discuss two approaches to generalizing size-guarding, and mention complexity classes for which size-guarding is superfluous, Then, we identify some problems with size-guards, which may complicate their use. For we generality, consider a generic real-ideal setting, and use zero-knowledge as an example. It is easy to see that both proposed notions of size-guarding are efficient transformations (in any sensible machine model).[89]

A case for size-guards. Recall that adversarial input *distributions*, which exploit expected polynomial size via fat-tailed distributions, may yield simulators which are not CEPT, because the have a, say quadratic, dependency on input length, whereas the real protocol (e.g. the prover) has a linear dependency, see Example E.26. Bounding input length, or even message length, which honest parties accept hardly affects the usefulness of a protocol. Indeed, these bounds are fixed *a posteriori*, i.e. after the full system is built from its parts. We have no good example for a setting, where there is no suitable polynomial bound one the input (or message) length of honest parties. So we expect that such a posteriori restrictions do not affect real applications.

Size-guarding inputs. The most natural approach to size-guarding is arguably to size-guard inputs to ideal functionalities, i.e. messages sent to the interface of real protocol or their ideal equivalent. Size-guarding a functionality yields a new functionality, which aborts upon receiving inputs which exceed the length allowed by the size-guard. (Adversarial parties should be allowed to ignore size-guard restrictions. Also, other parties should be notified of such an abort.) As explained above, we know no good example where a functionality cannot be replaced in such a way.

This simplistic sketch of size-guarding may be ill-defined, and lead to problems, as in pointed out in a later paragraph.

Size-guarding communication. Instead of size-guarding only inputs, one may want to size-guard all communication of honest parties. This may also be viewed as size-guarding all interfaces and the communication channel. (We should not impose size-guards on adversarial communication, as there is no justification for limiting their communication.) This kind of size-guarding is formally stronger, but we expect that for most (all?) interesting protocols, it is equivalent with size-guarding inputs. However, size-guarding communication affects everything, not just functionalities. Thus, we find the more local notion of size-guarding inputs preferable, and less likely to lead to unpleasant surprises.

Strict polynomial space. An algorithm A has a priori *strict (probabilistic) polynomial space* (SPS) (in analogy to PPT) if there exists a polynomial $\text{poly}(\kappa)$ which bounds maximal used space/memory. We count outgoing (but not incoming) message queues as part of an algorithms space/memory. With this, any size-guard larger than poly does not affect the behaviour of A at all. Thus, for a priori SPS adversaries, size-guarded security and normal security are equivalent.

For "classical" SPS, the space of A may depend on the input size, i.e. $\text{poly}(\kappa, |x|)$. All protocols of interest satisfy SPS. We find (classical, a posteriori and a priori) EPT SPS algorithms an appealing

[89] The effect of size-guarding on runtime is minor. If a lazy size-guard implementation is used, instead of eagerly checking the size, then up to emulation overhead, the runtime doubles at most. (Eager implementations may blow up runtime if the time for writing the message is not accounted for, e.g. because of huge messages from (inefficient) oracles.)

complexity class. Of course, the "negligible slack" of CEPT and CPPT is motivated for and applies to this setting as well. Unfortunately, deterministic bb-rw oracles for EPT adversaries are not compatible with SPS, hence our simulators are not PPT either. This can likely be fixed, see Remark A.6.

Composability and definitional issues. One major drawback of size-guarded security, is that it *changes the ideal functionality*. This may break properties, such as correctness, of protocols using such subprotocols. As mentioned before, size-guards should be chosen after a system is composed, so that such problems do not occur. Since a protocol may call a subprotocol with squared input length, for composition, one needs to keep track of size-guards, and be aware that they may not be identical for all protocols. That is, a protocol which is built from subprotocols imposes different size-guards on the subprotocols than the size-guard which was imposed on itself.

Another problem of size-guards is, that it may be convenient or relevant to have different or more fine-grained size-guards for different interfaces. E.g. for zero-knowledge, we left the witness unguarded. A more flexible approach than merely limiting the input length may be useful in a larger setting.

An alternative to size-guards. The problems noted above seem to disappear if instead of size-guarding inputs and changing protocol behaviour, one restricts to "admissible adversaries", as mentioned in Example E.26. The drawback is that now, one needs to specify admissibility variations for all notions, e.g. rewinding strategies, relative efficiency, and so on. We also caution that, similar problems as for size-guards may appear, just hidden deeper in security proofs.

E.5. Section 6

A simulator Sim is **benign under size-guarding**, if it is benign (Definition 6.20) whenever a polynomial size-guard is imposed on the protocol. Analogous claims for Lemmas 6.23 and 6.26 hold w.r.t. size-guarded zero-knowledge.

E.5.1. Connection between runtime and probability tightness

Following example illustrates, that normality is not automatic.

Example E.29 (Bad RWS). Consider a proof system with a (useless) preamble, where the prover sends a random string $s \leftarrow \{0,1\}^\kappa$, the verifier acknowledges it, and the actual protocol begins. A rewinding strategy RWS could send 0^κ as its first query, and then rewind. Against classical PPT adversaries, this is no problem at all. However, this essentially notifies the adversary of being in a simulation. Indeed, the probability tightness of RWS is 2^κ. Similarly, a rewinding strategy RWS, which "prefers" to output lexicographically smaller transcripts, typically has (very) noticeable output skew.

For EPT adversaries, runtime tightness implies probability tightness asymptotically.

Remark E.30 (Necessity of probability tightness). Let RWS be a rewinding strategy for $(\mathcal{P}, \mathcal{V})$. Let $\mathcal{V}^*$ be a deterministic malicious verifier. Suppose there is a (sequence of) logical queries $query = query(\kappa)$ such that $\mathsf{pr}_{\mathrm{rws}}(query) > \frac{1}{\mathsf{negl}} \cdot \mathsf{pr}_{\mathrm{real}}(query)$ infinitely often. By modifying $\mathcal{V}^*$ to run an extra $\frac{1}{\mathsf{pr}_{\mathrm{real}}(query)}$ steps if queried with *query*, we obtain a new deterministic verifier $\mathcal{V}^{**}$ whose expected runtime increases by 1. But RWS incurs a superpolynomial runtime growth, as it cannot see the "trap". Thus, runtime tightness implies probability tightness.

The "attack idea" on RWS in Remark E.30 may also be viewed as an indication that almost(?) all rewinding strategies in the literature are normal. More generally, even an a priori PPT adversary can exploit a large probability tightness and give "bad" answers in such cases. So, even for a priori PPT adversaries which cannot cause runtime explosion, there is no incentive to have large probability tightness, because it is unclear how that could be usefully exploited.

Remark E.31 (Probability tightness does not imply runtime tightness due to stupid reasons). Let RWS be some normal rewinding strategy. Construct RWS′ from RWS by running for 2^κ steps and then emulate RWS. Clearly, RWS′ and RWS are equivalent systems, but runtime tightness of RWS′ is exponential.

Nevertheless, for typical classes of well-behaved protocols and rewinding strategies, runtime tightness, "query tightness", and probability tightness are closely related.

E.6. ⋆ Absolute notions of relative efficiency

In Section 4.2, we work with "relative notions of (relative) efficiency", that is, we compare the performance of two algorithms. A scrapped approach used "absolute notions of relative efficiency", which have no comparison point. Absolute relative efficiency ensures, that whenever the communication partner of A is efficient, so is A. In other words, it allow us to "blame" a party for running too long. While this is easier to describe than relative efficiency, the need to be absolute makes the notion brittle, as we see at the end of this section.

We use the name *absolute relative efficiency* mostly due to a lack of a better name.

Definition E.32 (Weak absolute relative efficiency). Let $\mathcal{T}$ be a runtime class and A be an algorithm. Then A is **weakly absolutely relatively efficient (ar-eff)** (w.r.t. $\mathcal{T}$) if: For any timeful oracle $\mathcal{O}$, $\mathsf{time}_{\mathcal{O}}(\langle \mathsf{A}, \mathcal{O}\rangle) \in \mathcal{T}$ implies $\mathsf{time}_{\mathsf{A}+\mathcal{O}}(\langle \mathsf{A}, \mathcal{O}\rangle) \in \mathcal{T}$.

Definition E.33 (Absolute relative efficiency). Let A be an algorithm and $\mathcal{O}$ an oracle. Then A is **absolutely relatively efficient (ar-eff)** w.r.t. $\|\cdot\|_q$ with **rel-eff ratio** $\mathsf{poly}_{\mathsf{arr}}(\kappa)$ if: For any timeful oracle $\mathcal{O}$, we have $\|\mathsf{time}_{\mathsf{A}+\mathcal{O}}(\langle \mathsf{A}, \mathcal{O}\rangle)\|_q \leq \mathsf{poly}_{\mathsf{arr}}(\kappa) \cdot \|\mathsf{time}_{\mathcal{O}}(\langle \mathsf{A}, \mathcal{O}\rangle)\|_q$.

If q is not specified, we mean $q = \infty$, i.e. ar-eff w.r.t. to strict time. To $q = 1$, we say ar-eff **w.r.t. expectation**.

Importantly, the notion of (weak) absolute relative efficiency is *unconditional and amortized* since $\mathcal{O}$ is timeful and can abort at any time. In particular, if an algorithm is *PPT (resp. EPT) per activation*, it is (weakly) ar-eff.

As a rule of thumb, the *non-adversarial parties* (e.g. challengers) should be ar-eff, so that runtime problems can be traced back to the adversary. With this, one can exploit runtime explosions to break hardness assumptions.

Remark E.34 (Relation to EPT in any interaction). At fist glance, ar-eff (w.r.t. expectation) seems to be closely related to *EPT in any interaction (EPTiai)* [KL08; Gol10]. However, EPTiai has a different flavour. It is a property imposed on the (ideal) adversary, so that a simulator's runtime does not explode. Katz and Lindell state in[KL08, Sec. 4.2] that they could not show that the simulator obtained by modular sequential composition again satisfies EPTiai. This "prevents" further composition of this type. Conversely, ar-eff is a property imposed "honest parties", e.g. challenger in a security game.

Example E.35 (G3C is not ar-eff). The prover, verifier and simulator for $\mathrm{G3C}_{\mathrm{GK}}$ (Section 1.2) are ar-eff under size-guarding, but *not* unguarded, assuming $|(V, E)| \approx \mathsf{card}(V) + \mathsf{card}(E)$. The problem is that the $\mathcal{P}$ makes $\kappa \cdot \mathsf{card}(E) \cdot \mathsf{card}(V)$ commitments, whereas the verifier only makes $\mathsf{card}(E)$ commitments. The factor $\mathsf{card}(V)$ is not bounded by $\mathsf{poly}(\kappa)$, thus, there is no $\mathsf{poly}_{\mathsf{arr}}$ which depends only on κ and the prover is not ar-eff.[90]

This problem is mitigated by size-guards. For a variation of $\mathrm{G3C}_{\mathrm{GK}}$ with graph hamiltonicity, this problem would not occur as κ parallel repetitions suffice, independent of G. (Modulo technical complications.)

[90] This can be seen as a technical artefact from not counting the commitments sent by the prover towards the runtime of the verifier. An honest verifier would read the commitments, hence requiring roughly the same amount of "computation" as the prover. A dishonest or timeful verifier is not bound by that. If we would count incoming messages towards the runtime of the oracle, stupid problems like "message length doubling attacks" could appear. By sending a message m, A gets $\mathsf{poly}_{\mathsf{arr}}(\kappa) \cdot |m|$ more time from $\mathcal{O}$. If $\mathcal{O}$ discards messages which are too long, then $\mathcal{O}$ can remain efficient, whereas $\mathcal{A}$ increases its runtime exponentially. Arguably, we do not want to view such an A as efficient in any sense.

All in all, Example E.35 demonstrates how brittle *unguarded* ar-eff is. We see that not even the prover of $\mathrm{G3C}_{\mathrm{GK}}$ satisfies ar-eff without size-guards. This is the core reason to replace ar-eff with the arguably more complex notion of "efficiency relative to" another algorithm.

E.7. ⋆ The necessity of $\|\cdot\|_1$

One may hope that there is a notion more stringent than expected time, which still allows rewinding-based arguments of 3-move proofs of knowledge (based on special soundness), or 5-move zero-knowledge such as [GK96], with black-box proofs of security. For example, one might hope for $\|\cdot\|_2$ instead of $\|\cdot\|_1$, i.e. expected polynomial time and variation. Unfortunately, it is unlikely that a satisfying solution exists, at least along this line of arguments, unless one allows a larger (constant) number of rounds.

Concretely, consider the setting of 3-move proofs of knowledge. There, one can assume an adversary which plays honestly, but aborts with probability $1 - p \in [0, 1]$. Suppose the 3-move proof of knowledge is special sound and has large challenge space, so that it the soundness error is negligible. Consider the typical extractor for special soundness: If the adversarial prover convinces the verifier, it rewinds and uses honestly sampled challenges until the adversary produces a second convincing answer. With overwhelming probability, the first and second challenge are distinct, and by special soundness a witness can be computed.

It is evident that the number of rewinds for this extractor follows a geometric distribution. Indeed, with probability p, the first challenge is answered convincingly, in which case the extractor needs $R \sim \mathsf{Geo}(p)$ rewinds to obtains a second convincing answer. Then the expectation of R is

$$\|R\|_1 = (1-p) \cdot 0 + p\frac{1-p}{p} \leq 1.$$

If we consider $\|R\|_2$, then we find the

$$\|R\|_2^2 \geq p\frac{1-p}{p^2} \geq \frac{1}{p}.$$

Thus, if p negligible, the $\|R\|_2$ is superpolynomial. A first attempt is to exploit virtuality: If p is negligible, then in fact, R^2 is virtually expected polynomial. Conversely, if p is bounded below by $\frac{1}{\mathsf{poly}}$, then R^2 is expected polynomial. However, things fall apart if $p = p(\kappa)$ is not negligible, yet not bounded below by any polynomial. For this, define $p(\kappa)$ as follows: $p(\kappa) = \kappa^{-2^{f(\kappa)}}$, where $f(\kappa) = 1$ for all $2 \nmid \kappa$, $f(\kappa) = 2$ for all $2 \mid \kappa \wedge 4 \nmid \kappa$, $f(\kappa) = 3$ for all $4 \mid \kappa \wedge 8 \nmid \kappa$, and so on. (Let $f(0) = 0$.) That is,

$$(f(\kappa))_\kappa = (0, 1, 2, 1, 3, 1, 2, 1, 4, 1, 2, \ldots)$$

Thus, p contains infinitely many terms of κ^{-2^c} for any $c \in \mathbb{N}$. It is easy to see that p is not negligible. However, for any polynomial poly, we have $p < \frac{1}{\mathsf{poly}}$ infinitely often, i.e. p is not polynomially bounded away from 0. Thus, $\|R\|_2 > \mathsf{poly}$ infinitely often. In other words, there is no polynomial which bounds $\|R\|_2$. Allowing negligible virtuality does not help either. Thus, this choice of p results in an adversary which cannot be extracted in virtually expected polynomial $\|\cdot\|_2$-time.

Repetitions can be used to "bring down exponents", and hence, for any $q \in \mathbb{N}$, there should exist a (constant) number C of repetitions (and modified extractors) such that $\|R\|_q < \mathsf{poly}$, namely $C = q$. This may be interpreted as an intermediate result between proofs of knowledge with EPT extraction (i.e. $q = 1$), and "strong proofs of knowledge" with PPT extraction (i.e. $q = \infty$). We also refer to [Pas06] where this is discussed in the context of precise zero knowledge proofs (of knowledge).

E.8. ⋆ Measurability

In this section, we discuss questions of measurability, which we ignored elsewhere. Since all of our constructions are simple and make no use of the axiom of choice, there is little reason to doubt that all

are measurable. Admittedly, we have not formally verified this for every construction, and merely spot-checked some. We do note that some properties, e.g. "uniqueness" of events, were used in simplified explanations. They are not used in actual constructions.

Stochastic processes and timeful systems. The evolution of a computation or interaction for closed systems should be viewed as a stochastic process. The random variables of interest are the exchanged messages, and the purported elapsed time,[91] namely the sequence of random variables $(Z_0, Z_1, \ldots)$ describing the progress of the computation. Concretely, Z_i consists of $(m_0, t_0, \ldots, m_i, t_i)$, which is transcript up to the i-th message exchange, plus the elapsed runtime t_j for computing m_j. One may augment this with other (purported) values, such as memory usage, etc. Obviously, we also require $\mathsf{proj}_{1,\ldots,j}(Z_i) = Z_j$ for all $i < j$ in $\mathbb{N}_0$. (And this implies $\sigma(Z_i) \subseteq \sigma(Z_{i+1})$ for the σ-algebras.)

The image of Z_i lies is in a countable space, which we equip with the discrete σ-algebra. The sample space of process $Z(i, \omega) = Z_i(\omega)$ is given the induced σ-algebra, but is not countable anymore. (Recall that the σ-algebra on $\mathbb{N}^\infty$ is constructed from the finite steps $\mathbb{N}^k$, $k \in \mathbb{N}$.)

We have ignored inputs and non-closed systems, but these are easily defined as *functions*, which take a sequence of input messages and return output messages. (Technically, these may be partial functions, since some input sequences may correspond to impossible executions. E.g. inputs for a system which halted.) Letting two such systems A, B interact by connecting interfaces yields a new system, defined in the obvious way. The resulting system $\langle \mathsf{A}, \mathsf{B} \rangle$ has an associated random process (which lives in the product space of the random processes associated with A, B.)

An alternative description. One may alternatively describe the random process of individual systems via "conditional transition probabilities", roughly, $p_i(\vec{y}) = \mathbb{P}(Z_i = \vec{y} \mid Z_{i-1} = \mathsf{proj}_{1,\ldots,i-1}(\vec{y}))$. This approach always specifies independent processes. Dependency is (only) introduced by interaction. While this would probably be sufficient, "extending" the probability space (as we did to achieve exact ν-quantile cutoffs) is not immediately possible. One can introduce some "irrelevant action", e.g. a zeroth message, which has the desired distribution. We find this to be just as inconvenient as working with underlying probability spaces explicitly. Moreover, for systems induced by algorithms and machine models, the underlying probability space is usually explicit anyway.

Algorithms and machine models. Unlike (timeful) systems, algorithms and machine models have an "explicit randomness-providing interface". Thus, the underlying probability space for such systems is simple to describe, usually $\{0,1\}^\mathbb{N}$ with "uniform" distribution (i.e. the "limit" of $\{0,1\}^k$, $k \in \mathbb{N}$, with uniform distribution). Standard definitions of machine models (via transition functions) then evidently imply that any algorithm yields systems which are very well behaved, in particular every typical function of interest is measurable (e.g. messages, runtime, memory trace, ...).

Measurability of our constructions. Most constructions merely relied on runtime statistics, and can be defined on the process Z_i by (a consistent family of) measurable functions (for each i). Indeed, if the domain of Z_i has the discrete σ-algebra, any function is measurable. Since these statistics are measurable by assumption, and our functions are measurable as well, we therefore find that our constructions are measurable. (More concretely, they are measurable for every i, and hence the resulting process is measurable.)

E.9. ⋆ Musings on runtime classes

Instead of dealing with runtime *distributions* only, a runtime class could deal with *random variables*. For this, fix some (family of) universal probabilistic space(s) Ω_κ and redefine runtime classes as follows:

[91] In particular, a *timeful* system must have measurable purported runtime.

Definition E.36. A runtime class $\mathcal{T}$ is a set of (families of) random variables $T\colon \Omega \to \mathbb{N}_0 \cup \{\infty\}$ with following property:[92] If T and S have the same distribution, then either both or none lie in $\mathcal{T}$. In other words, membership in $\mathcal{T}$ only depends on the distribution.

Only the distribution matters for membership. But operations, such as sums, of distributions and random variables differ — random variables are closer to "practical" usage of runtime, e.g. simulation with 3-fold overhead or sums of dependent runtimes.

Given such a "better" definition, we want to impose additional constraints on what should be considered a *runtime* class. We derive these from properties of bound algebras.

Example E.37. A "good" runtime class $\mathcal{T}$ should satisfy following properties:

Constants: The constant 0 and constant 1 runtime are in $\mathcal{T}$.[93]

Closed under domination: For any $T \in \mathcal{T}$, all its dominated runtimes S are contained in $\mathcal{T}$ as well, i.e. $\forall S \forall T \in \mathcal{T}\colon S \overset{d}{\leq} T \implies S \in \mathcal{T}$. Recall that $S \overset{d}{\leq} T$ is defined pointwise w.r.t. κ, i.e. $\forall \kappa\colon S_\kappa \overset{d}{\leq} T_\kappa$, and recall that $X \leq Y$ means Y dominates X (in distribution).

Closed under addition: For any $T, S \in \mathcal{T}$, also $T + S \in \mathcal{T}$. Note that this is the sum of *random variables*, not distributions.

Asymptotically monotone: Let $T \in \mathcal{T}$ and let $S_\kappa \coloneqq \max\{T_1, \dots, T_\kappa\}$. Then $S \in \mathcal{T}$. This statement is of nonsensical if $\Omega_i \neq \Omega_j$. Therefore, it is a statement about distributions.

Remark E.38. The closedness under domination says that no "inefficient" algorithm (i.e. runtime outside $\mathcal{T}$) can be made efficient by doing *more* steps. Closedness under addition is a (weak) abstraction for saying that (finite) sequential composition of $\mathcal{T}$-time algorithms is again a $\mathcal{T}$-time algorithm. It also models that constant multiples of a runtime remain efficient. In particular, any constant runtime lies in $\mathcal{T}$. Asymptotic monotonicity should ensure that increasing κ only increases admitted runtimes, e.g. efficiency of constant runtimes can be tested for $\kappa = 1$.

Remark E.39 (Weak composition). We lack a generalization of "composability" of runtimes, which mirrors "oracle composition" in a weak form. For bound algebras, this was multiplicative closedness. There is the obvious candidate of letting $T * S$ be the product of random variables. This is most likely not what we need. Instead, considering a T-fold sum of independently drawn S's is more plausible (but non-commutative). Such a "**weak composition**" models "independent sampling access" for a runtime distribution, which seems sufficient in our triple-oracle distinguishing setting.[94]

Remark E.40. The item on "asymptotic monotonicity" is the most questionable one, and we are not sure if it is the right point of view. Many alternative approaches exist. One advantage of our choice is, that it is easy to see that arbitrary intersections of good runtime classes are again "good" runtime classes. This is a first step for analysing whether the intersections of all closed "good" runtime classes is again closed, and whether it is equal to our definition of closure.

A nice property of bound algebras, which we did not add to Example E.37, is the existence of a countable "monotone generating sets". We do not know whether or not this is a good addition. Unlike sequences in $\mathbb{N}_0$, sequences of distributions behave very differently. In particular, since domination of distributions is not a total order.

[92] Perhaps even further, a runtime class should be a function, mapping a probability space Ω to a runtime class $\mathcal{T}(\Omega)$ with suitable compatibility rules.

[93] We are not certain whether or not this property is absolutely necessary. However, the class of expected $O(1/\kappa)$ time is both strange and behaves badly. For example, inverting the output (which takes a constant number of steps) cannot be done. Arguably, such pathological behaviour is best avoided.

[94] Using (effectively) a *distribution* S here, but not in closedness under addition seems questionable. Allowing dependent S instances may be possible, but combining it with dependent T leads to disaster (since abstract EPT explosion examples can now be modelled). Perhaps, closedness under addition should be weakened, and is therefore not necessary at all (because it is subsumed by "weak composition")? At least for the abstract theory, this may be a valid choice. In fact, that decision would allow to define runtime classes as sets of distributions again.

Another interesting question is that of a canonical d-dense subclass in $\mathcal{T}$, or a lack thereof. A very large canonical subclass is that of finite runtimes (i.e. where $\mathbb{P}(T_\kappa \leq N_\kappa) = 1$). But is there a general analogon to d-density of $\mathsf{RTC}_\infty(\mathcal{B})$ for $\mathcal{B}$-tailed runtime classes? If $\mathcal{T}$ satisfies strong guarantees, we have a plausible candidate.

Example E.41 (A candidate for $\mathsf{RTC}_\infty(\mathcal{T})$). In the proof of Corollary D.31, it was central to consider $\mathsf{tail}^\dagger_{T_\kappa}(\alpha)$. This leads us to our candidate definition of $\mathsf{RTC}_\infty(\mathcal{T})$ as $\mathcal{T}_{\mathsf{tail}} \subseteq \mathcal{T}$. The runtime class $\mathcal{T}_{\mathsf{tail}}$ is generated by $\mathsf{tail}^\dagger_{T_\kappa}(\alpha)$ for every constant $\alpha > 0$ and every $T \in \mathcal{T}$. Moreover, assuming "weak composability" then $\mathcal{T}_{\mathsf{tail}}$ actually defines a *bounds algebra*. The proof of Corollary D.31 also requires "weak composability" (in the sense of Remark E.39), and further properties such as "smallness" of runtimes.

Interestingly, $\mathcal{T}_{\mathsf{tail}}$ equals strict $\mathsf{RTC}_\infty(\mathcal{B})$ for $\mathcal{B}$-tailed runtime classes. The definition of $\mathcal{T}_{\mathsf{tail}}$ also gives rise to a bound algebra $\mathcal{B}_{\mathsf{tail}}$. This gives some hope that, perhaps, our restricted treatment of algebra-tailed runtime classes was not too restrictive after all. Indeed, if we could show that $\mathcal{T}$ is $\mathcal{B}_{\mathsf{tail}}$-tailed, we're done. This property is closely related to the "smallness" condition in Corollary D.31, and to "equivalence of statistical and computational indistinguishability". Indeed, the relation of these three properties appears to be of central importance. Characterizing the "equivalence of statistical and computational indistinguishability" for general runtime classes would be a core tool for working with them. For example, can quasi-linear time be distinguished from non-quasi-linear time in quasi-linear time? Is "weak composability" really necessary, or is it just a convenient property?

To summarize, we gave some best guesses for candidate definitions and properties for "good" runtime classes. But we lack suitable theoretical evidence towards the usefulness of their "good" nature. Indeed, there are many open questions of abstract interest, for which we have no answers.

F. ⋆ Extendability from indistinguishable queries

Our definition of benign simulators relies on structure of the proof of security for (PPT) simulation, and, although it covers many examples, is therefore somewhat limited. In this section, we give a different approach to benign simulation. Intuitively, we require that an "eavesdropping" environment cannot distinguish the bb-rw interaction of a rewinding strategy or a simulator with $\mathcal{V}^*$. This corresponds to the properties of query-indistinguishability and zero-knowledge.

The upside of this approach is its apparent greater generality. The downside is, that using query-indistinguishability is more technical, and requires a separate treatment of efficiency and indistinguishability. Perhaps a better, general approach exists — yet we know none.

F.1. Query-sequences indistinguishability

Our notion of "indistinguishable queries" for simulators is similar in spirit to [KL08].

Definition F.1 (Query-indistinguishability). Let A and B be oracle algorithms. The distinguishing advantage $\mathrm{Adv}^{\mathrm{qseq}}_{(\mathcal{I},\mathcal{O},\mathcal{D}),\mathsf{A},\mathsf{B}}(\kappa)$ for queries *including output* of A, B by an adversary $(\mathcal{I}, \mathcal{O}, \mathcal{D})$ is defined as the distinguishing advantage $\mathrm{Adv}^{\mathrm{dist}}_{\mathcal{D},X,Y}(\kappa)$ for the distributions

$$X \coloneqq \{(x, y, r, \mathsf{A}^{\mathcal{O}(y;r)}(x; r_{\mathsf{A}}), \mathsf{qseq}_{\mathcal{O}}(\mathsf{A}^{\mathcal{O}(y;r)}(x; r_{\mathsf{A}}))) \mid (x, y) \leftarrow \mathcal{I}(\kappa)\}_\kappa$$
$$Y \coloneqq \{(x, y, r, \mathsf{B}^{\mathcal{O}(y;r)}(x; r_{\mathsf{B}}), \mathsf{qseq}_{\mathcal{O}}(\mathsf{B}^{\mathcal{O}(y;r)}(x; r_{\mathsf{B}}))) \mid (x, y) \leftarrow \mathcal{I}(\kappa)\}_\kappa$$

Here r denotes the *accessed* randomness of $\mathcal{O}$.[95] (We make explicit the randomness r_{A} and r_{B} only to make it evident, that the output and query sequence refer to the same run.)

[95] Typical machine models offer an infinite pool of (independent) randomness, e.g. a random tape. Thus, we "restrict" to accessed randomness.

We say that A and B satisfy **($\mathcal{T}$-time) query-indistinguishability (Q-IND)**, if for all adversaries $(\mathcal{I}, \mathcal{O}, \mathcal{D})$ such that $\mathsf{time}_{\mathcal{I}+\mathcal{O}+\mathcal{D}}(\mathsf{A}^{\mathcal{O}}) \in \mathcal{T}$ and $\mathsf{time}_{\mathcal{I}+\mathcal{O}+\mathcal{D}}(\mathsf{B}^{\mathcal{O}}) \in \mathcal{T}$ the advantage $\mathrm{Adv}^{\mathrm{qseq}}_{(\mathcal{I},\mathcal{O},\mathcal{D}),\mathsf{A},\mathsf{B}}(\kappa)$ is negligible.

Size-guarded query-indistinguishability is defined by size-guarding A and B (as non-adversarial parties), i.e. A and B reject inputs of length larger than their size-guard.

Definition F.1 requires *jointly* indistinguishable *queries* and *outputs*. This may not be strictly necessary, but greatly simplifies sequential composition of Q-IND. All bb-rw zero-knowledge simulators we are aware of satisfy this joint indistinguishability. Indeed, typically the last query induces the (purported) view of the adversary.

We stress that the distinguisher $\mathcal{D}$ learns the oracle randomness. This allows $\mathcal{D}$ to replay the execution of $\mathcal{O}$, recover the complete transcript of the execution, and compute the runtime spent in $\mathcal{O}$.

Remark F.2. For CEPT and CPPT, it suffices in Definition F.1 to require that $T = \mathsf{time}_{\mathcal{I}+\mathcal{O}+\mathcal{D}}(\mathsf{A}^{\mathcal{O}}) \in \mathcal{T}$. If $\mathsf{time}_{\mathcal{I}+\mathcal{O}+\mathcal{D}}(\mathsf{B}^{\mathcal{O}}) \notin \mathcal{T}$, then this is a distinguishing statistic. Indeed, by a standard reduction to PPT, any distinguisher $(\mathcal{I}, \mathcal{O}, \mathcal{D})$ with advantage at least $\varepsilon = \mathsf{poly}^{-1}$ (infinitely often) can be truncated to an *a priori* PPT distinguisher with advantage $\frac{\varepsilon}{4}$ (infinitely often). (Just interpret $\mathcal{D}' \mathrel{\widehat{=}} (\mathcal{I}, \mathcal{O}, \mathcal{D})$ as interacting with oracles A or B, and apply Corollary 4.2.)

Remark F.3 ("Universal" adversary, environments, sequential security). Using a universal machine for $\mathcal{O}$ (and even $\mathcal{D}$) gives a universal adversary similar to zero-knowledge. Moreover, one can rephrase Definition F.1 in terms of an "environment" $\mathcal{E}$ which encompasses $\mathcal{I}$ and $\mathcal{D}$; $\mathcal{E}$ sends inputs (x, y), and then gets access to randomness, output and query sequence. A sequential security version of Q-IND is defined by this approach, following the definition of sequential zero-knowledge (i.e. $\mathcal{E}$ is given adaptive repeated trials).

As in Remark F.2, we may assume $\mathcal{E}$ is a priori PPT. Also, (one-guess) "environmental" security is equivalent to Definition F.1, because the state of $\mathcal{E}$ can be encoded as part of y.

F.2. Adapting the result of Katz–Lindell

As a warm-up, we adapt the result of Katz and Lindell [KL08]. For that, we rely on bb-rw simulators which are EPT for *any* adversary (not counting the adversary's steps). In other words, we rely on simulators which are normal in the sense of Goldreich [Gol10]. That covers most simulators in the literature, but not our naive simulator for $\mathrm{G3C}_{\mathrm{GK}}$. Moreover, the definition is not compatible with expected polynomial input sizes, and thus restricted to size-guarded security.[96] (For size-guards, see Appendices E.4 and E.4.3) After this motivation, we generalize the result to our setting.

Definition F.4 (Goldreich-normal [Gol10]). A bb-rw simulator is **normal in the sense of Goldreich**, short **Goldreich-normal**, if for any (not necessarily computable) timeful $\mathcal{V}^*$ and any input (x, w, aux) (with $(x, w) \in \mathcal{R}$) there is a polynomial poly such that $\mathbb{E}(\mathsf{time}_{\mathsf{Sim}}(\mathsf{Sim}(x, \mathcal{V}^*(\mathit{aux})))) \leq \mathsf{poly}_{\mathsf{Sim}}(|x|)$,

There is no requirement of $x \in \mathcal{L}$ in [Gol10, Definition 6]. Since zero-knowledge only quantifies over such statements, we have adapted the definition to fit.

Lemma F.5 (Auxiliary input zero-knowledge). *Let $(\mathcal{P}, \mathcal{V})$ be an argument system. Let* Sim *be a (timed) bb-rw simulator with associated rewinding strategy* RWS*. Suppose that* RWS *is normal,* Sim *is Goldreich-normal,* RWS *and* Sim *have indistinguishable queries, and* Sim *handles PPT adversaries in EPT.*

Then Sim *handles CEPT adversaries in CEPT* under size-guarding*, and $(\mathcal{P}, \mathcal{V})$ is size-guarded zero-knowledge.*

Proof sketch. By a standard reduction, the output quality of Sim can be tested by an a priori PPT adversary. By assumption, such output is indistinguishable from the real protocol. Thus, we only need to ensure efficiency of Sim under size-guarding.

[96] These problems do no surface in [KL08; Gol10] since they define $\kappa = |x|$.

Due to size-guarding, we can assume that $\mathcal{I}$ outputs x with $|x| \leq \mathsf{poly}_{\mathcal{I}}(\kappa)$. Therefore, by assumption, Sim is *a priori* EPT with bound $\mathsf{poly}_{\mathsf{Sim}}(\mathsf{poly}_{\mathcal{I}}(\kappa))$, *excluding* the time spent in the bb-rw oracle $\mathcal{V}^*$. Thus, it is sufficient to bound $S_{\mathcal{V}^*} = \mathsf{time}_{\mathcal{V}^*}(\mathsf{Sim}(x, \mathcal{V}^*(\mathit{aux})))$, where $(x, w, \mathit{aux}) \leftarrow \mathcal{I}$. By normality of RWS and query-indistinguishability, recomputing the time spent in $\mathcal{V}^*$ by emulation (using inputs, queries and randomness) is possible in CEPT for any CEPT adversary. Consequently, by query-indistinguishability, switching from RWS to Sim results in an indistinguishable distribution of t. Hence, $S_{\mathcal{V}^*}$ is CEPT and the claim follows. □

We also demonstrate that sequential composition, i.e. sequential zero-knowledge, holds for this type of simulator.

Lemma F.6 (Sequential zero-knowledge). *Let $(\mathcal{P}, \mathcal{V})$ be a size-guarded zero-knowledge argument, with a simulator satisfying the conditions in Lemma F.5. Then $(\mathcal{P}, \mathcal{V})$ is sequential size-guarded zero-knowledge.*

Proof sketch. Again, the main question is efficiency. Namely, if there is a distinguishing adversary for zero-knowledge, then there is an a priori PPT adversary. This contradicts our assumptions, because "classical" sequential composition against a priori PPT adversaries holds.

To prove efficiency, we prove, essentially, that query-indistinguishability composes sequentially. As in Lemma F.5, this then implies that $\mathcal{O}_{\mathsf{Sim}}$ is efficient because $\mathcal{O}_{\mathsf{RWS}}$ is.

Suppose the contrary, i.e. suppose query-indistinguishability does not hold for Sim. By Remarks F.2 and F.3, we know that there is an a priori PPT distinguisher $(\mathcal{E}, \mathcal{V}^*)$ breaking "sequential query-indistinguishability". We leave the definition of sequential Q-IND, sketched in Remark F.3, to the reader.

Since Sim is Goldreich-normal, $\mathcal{O}_{\mathsf{Sim}} = \mathsf{rep}(\mathsf{Sim}(\cdot))$ handles PPT adversaries in EPT. (This is "classical" sequential composition.) Sequential Q-IND of Sim and RWS for PPT distinguishers reduces, by a hybrid argument, to standard Q-IND. The hybrid distinguisher is efficient, because Sim is Goldreich-normal (and RWS normal). Consequently, Sim and RWS cannot be Q-IND. A contradiction. Hence, sequential Q-IND holds for RWS and Sim. In particular, $\mathsf{rep}(\mathsf{Sim})$ satisfies all conditions in Lemma F.5 lifted to the sequential setting, and the proof lifts as well. □

This warm-up demonstrates two things: First, with size-guarding, many arguments get simplified and reduce to standard a priori PPT arguments. Second, the main difficulty for relaxations will be to demonstrate efficiency. By the nature of CEPT, efficiency and indistinguishability are somewhat entangled. We use unconditional guarantees, similar to Goldreich-normal in the above, to partially disentangle that.

Proving a "full-fledged" CEPT simulation, i.e. getting rid of size-guarding and weakening Goldreich-normal is surprisingly cumbersome. We do so by introducing two properties. The first property, *runtime estimators*, allows us to link together the runtime of RWS and Sim, *assuming Q-IND holds*. The second property ensures efficiency if one truncates after polynomially many queries. This replaces Goldreich-normal, and enables the hybrid argument which shows that Q-IND must hold under sequential composition. Taken together, we find that the runtime of Sim cannot be too far from RWS, and thus Sim is efficient whenever RWS is. This generalizes the proof of Lemma F.6.

F.3. Runtime estimation

In the following, we give a definition of a "runtime estimator", which allows to lower- and upper-bound the expected runtime of an algorithm depending on oracle queries, or more precisely, on the information available to a Q-IND adversary. The algorithms of interest are RWS and Sim. Typically, their runtime is closely related, since both emulate the honest prover (with minor modifications). Consequently, their runtime per activation is easy to lower- and upper-bound (if the prover's runtime per activation is).

Definition F.7 (Runtime estimation). Let $\theta\colon \mathbb{N}_0 \times D \times \Omega_\theta \to \mathbb{N}_0$, be a probabilistic algorithm with randomness space Ω_θ, and where D is the input space of a query distinguisher (as in Definition F.1).

Let A be an algorithm and $\mathcal{O}$ some oracle. Let $z \in D$ and recall that $z = (x, y, r, \mathit{out}, \mathit{qs})$, where x (resp. y) is input to A (resp. $\mathcal{O}$), r is the oracle randomness, out is the output of $\mathsf{A}^{\mathcal{O}(y;r)}(x)$, and qs is the sequence of queries. Define

- $t_\theta(\kappa, z) \coloneqq \mathbb{E}(\mathsf{time}_\theta(\theta(\kappa, z)))$, the expected runtime of θ given z.
- $t_{\mathsf{A}}(\kappa, z) \coloneqq \mathbb{E}(\mathsf{time}_{\mathsf{A}}(\mathsf{A}^{\mathcal{O}(y;r)}(x)) \mid \mathsf{A}^{\mathcal{O}(y;r)}(x) = \mathit{out} \wedge \mathsf{qseq}_{\mathcal{O}}(\mathsf{A}^{\mathcal{O}(y;r)}(x)) = \mathit{qs})$, the expected runtime of A conditioned on z.
- $t_{\mathsf{A}+\mathcal{O}}(\kappa, z)$ like t_{A}, but using $\mathsf{time}_{\mathsf{A}+\mathcal{O}}(\ldots)$.

We say that θ is a **runtime estimator** if it satisfies **efficiency**, i.e. there exists some $\mathsf{poly}(\kappa)$ such that for all $z \in D$ and all κ: $t_\theta(\kappa, z) \leq \mathsf{poly}(\kappa) \cdot t_{\mathsf{A}+\mathcal{O}}(\kappa, z)$. Moreover, θ is a **lower bound estimator** if there exists some poly such that $\mathbb{E}(\theta(\kappa, z)) \leq \mathsf{poly}(\kappa) \cdot t_{\mathsf{A}}(\kappa, z)$ for all $z \in D$ and $\kappa \in \mathbb{N}_0$. Analogously, θ is a **upper bound estimator** if there exists some poly such that $t_{\mathsf{A}}(\kappa, z) \leq \mathsf{poly}(\kappa) \cdot \mathbb{E}(\theta(\kappa, z))$ for all $z \in D$ and $\kappa \in \mathbb{N}_0$.

Note that estimators are "unconditional" constructions; we quantify over all $z \in D$.

Remark F.8 (Sketched application of runtime estimators). Consider a simulator Sim and its rewinding strategy RWS. If $T = \mathsf{time}_{\mathsf{RWS}}(\mathsf{RWS}^{\mathcal{V}^*})$ is CEPT, then the (output of the) runtime estimate θ is CEPT if it lower-bounds T. If θ upper-bounds $S = \mathsf{time}_{\mathsf{Sim}}(\mathsf{RWS}^{\mathcal{V}^*})$, then S is CEPT if (the output of) θ is. Since θ only depends on the information available to a Q-IND adversary, assuming RWS and Sim are Q-IND, the runtime bound provided by θ only changes negligibly, hence if T is CEPT, so is S. This provides a central link between the runtime RWS and Sim.

Remark F.9 (Convenience of size-guards). Arguing via runtime estimates requires that the algorithms runtime per activation behave somewhat regularly (which is fortunately typical). Most convenient are "essentially constant-time" algorithms (where runtime only depends on query/message length). With size-guards this is usually immediate, as every round has an a priori polynomial upper bound for the (expected) number of steps taken, both in RWS and Sim (not counting the black-box $\mathcal{V}^*$). Hence θ is as simple as the total number of queries. Without size-guards, the behaviour is more fickle.

F.4. Efficiency from query-truncation

We already saw in Lemmas F.5 and F.6 that Q-IND ensures that the time spent in $\mathcal{V}^*$ only changes negligibly between RWS and Sim. However, we cannot reuse the arguments to show that Q-IND composes sequentially. The problem lies within efficiency of the hybrid distinguisher. As seen in Lemma F.6, once we obtain an a priori setting, this problem "disappears". Hence this is our solution. We define what it means to be "Goldreich-normal for any polynomial query cutoff" of the interaction. Intuitively, it means that any "polynomial prefix" of the interaction is Goldreich-normal.

Definition F.10. Let A be an oracle-algorithm. Let $\mathsf{B}^{\mathcal{O}}(x, q)$ be the oracle-algorithm which emulates $\mathsf{A}^{\mathcal{O}}(x)$ until the q-th query of A to $\mathcal{O}$. After that, B returns `timeout` (otherwise B returns whatever A returns). We say A is **Goldreich-normal for any polynomial query cutoff** (and input space $\mathcal{X}_\kappa$), if for any polynomial poly_0 there is a polynomial poly_1, such that for any oracle $\mathcal{O}$, and any inputs $(x, y) \in \mathcal{X}_\kappa$ $\mathbb{E}(\mathsf{time}_{\mathsf{B}}(\mathsf{B}^{\mathcal{O}(y)}(x, \mathsf{poly}_0(\kappa)))) \leq \mathsf{poly}_1(|x|, \kappa)$. In other words, $\mathsf{B}(\cdot, \mathsf{poly}_0)$ is Goldreich-normal for any poly_0. For zero-knowledge, the input space is $\mathcal{R} \times \{0,1\}^*$.

Example F.11. Our rewinding strategy and simulator of $\mathrm{G3C}_{\mathrm{GK}}$ are Goldreich-normal for any polynomial query cutoff. Indeed, they are even PPT for any polynomial query cutoff. As a matter of fact, we cannot point out any (natural) bb-rw simulator which does not satisfy this property.

Remark F.12 (Goldreich-normal for any polynomial query cutoff does not imply efficiency). Similarly to size-guarding, restricting to a polynomial number of queries makes simulations efficient which would otherwise not be. For example, a simulator which is a a priori PPT per activation, but never halts, is Goldreich-normal for any polynomial query cutoff.

F.5. Query-benign simulators

Now, we bring together our definitions to define an alternative of benign, which we call query-benign.

Definition F.13 (Query-benign simulator). Let $(\mathcal{P}, \mathcal{V})$ be an argument system. Let Sim be a *(timed) bb-rw* simulator with **associated rewinding strategy** RWS. Then Sim is **query-benign** if

(1) RWS is a *normal* and has a runtime estimator θ;
(2) for all a priori PPT adversaries $(\mathcal{I}, \mathcal{V}^*)$, RWS and Sim satisfy Q-IND;
(3) Sim is Goldreich-normal for any polynomial query cutoff.

Query-benign under size-guard gd is as usual (i.e. by query-benign w.r.t. the size-guarded prover).

Recall that Q-IND (i.e. condition item (2)) implies that RWS and Sim have indistinguishable outputs by definition, i.e Q-IND implies zero-knowledge. In (2) we use a priori PPT adversaries, since the security is equivalent to CEPT anyway.

Now, we put our definitions to use. Since our arguments are very close to Lemma F.6, we directly show sequential zero-knowledge.

Lemma F.14 (Query-benign implies sequential zero-knowledge). *Suppose $(\mathcal{P}, \mathcal{V})$ is an argument system. Let* Sim *be a query-benign simulator. Then $(\mathcal{P}, \mathcal{V})$ is sequential zero-knowledge. In particular,* Sim *handles CEPT adversaries in CEPT. The analogous claim holds under size-guarding.*

Our proof is only a sketch and somewhat hand-wavy. in particular, we leave sequential security definitions, like "sequential Q-IND" and "sequential runtime estimators", and many straightforward arguments to the reader.

Proof sketch. As usual, the proof consists of two parts. First, we prove that using Sim instead of $\mathcal{P}$ is still CEPT. Then, by standard arguments, zero-knowledge follows. For simplicity, we argue assuming the real execution halts with probability 1.

Step 1 (Replacing RWS): As in Lemma 6.26, using $\mathsf{rep}(\mathsf{RWS}(\cdot))$ instead of $\mathsf{rep}(\langle \mathcal{P}, \cdot \rangle)$ in the sequential zero-knowledge experiment is still CEPT and the output distribution is unchanged.[97]

Step 2 (Goldreich-normal for any polynomial query cutoff composes sequentially): It is straightforward to verify that if an algorithm B is Goldreich-normal for any polynomial query cutoff, so is its "repetition" $\mathsf{rep}(\mathsf{B})$. For this compare, the poly-query truncation of $\mathsf{rep}(\mathsf{B})$ with $\mathsf{rep}(\mathsf{B}_0)$, where B_0 is the poly-query truncation of B. Since B_0 is Goldreich-normal, so is $\mathsf{rep}(\mathsf{B}_0)$. Consequently, $\mathsf{rep}(\mathsf{B})$ is Goldreich-normal for any polynomial truncation.

Step 3 (Q-IND holds for $\mathsf{rep}(\mathsf{RWS})$ and $\mathsf{rep}(\mathsf{Sim})$): Now, consider the "sequential Q-IND" experiment, i.e. consider Q-IND of $\mathsf{rep}(\mathsf{RWS})$ and $\mathsf{rep}(\mathsf{Sim})$. More concretely, the distinguishing environment $\mathcal{E}$ that can repeatedly invoke $\mathsf{RWS}^{\mathcal{V}^*}$ resp. $\mathsf{Sim}^{\mathcal{V}^*}$, and obtains the output of an invocation, including the query sequence and randomness of (that invocation of) $\mathcal{V}^*$, as noted in Remark F.3. Note that $\mathcal{E}$ can adaptively choose inputs to RWS resp. Sim and $\mathcal{V}^*$.

W.l.o.g., we may assume that $\mathcal{E}$ is a priori PPT, say $\mathcal{E}$ makes at most $\mathsf{poly}_{\mathcal{E}}$ steps. Moreover, we may assume that $\mathcal{E}$ *linearly reads* the outputs of each invocation. In particular, $\mathcal{E}$ cannot skip (parts) of the outputs, and read only the final queries.[98] Importantly, $\mathcal{E}$ only reads a strict polynomial prefix of the full (sequential) query sequence.

If we replace Sim by a truncation Sim_0, which stops after $\mathsf{poly}_{\mathcal{E}}$ queries, we know that Sim_0 is EPT with expected runtime bounded by some $\mathsf{poly}_{\mathsf{Sim}_0}$ (due to Sim being Goldreich-normal for any polynomial query truncation, see also Step 2).

[97] In case of non-halting executions, argue as in Lemma 6.23.

[98] This is a technical requirement. Depending on the machine model, $\mathcal{E}$ may have random access to the output. That would make our later argument incomplete. To see that we can assume that $\mathcal{E}$ completely reads the outputs, just use the output length as a distinguishing statistic. That is, if there is a PPT distinguisher $\mathcal{E}$ which skips parts of the output, then the variation which reads *all of the output* is still CEPT for RWS. By standard truncation arguments, an a priori PPT truncation of $\mathcal{E}'$ retains non-negligible advantage. And $\mathcal{E}'$ is a distinguisher of the kind we are interested in.

By construction, from the perspective of $\mathcal{E}$, $\mathsf{rep}(\mathsf{Sim}_0)$ and $\mathsf{rep}(\mathsf{Sim})$ behave identically. Indeed, since $\mathcal{E}$ only reads at most a prefix of length $\mathsf{poly}_{\mathcal{E}}$ of the (total) query sequence, $\mathcal{E}$ never encounters the difference of Sim_0 and Sim. For symmetry, let RWS_0 be defined analogously to Sim_0. (Formally, we could use RWS, since there are no efficiency problems with RWS.)

Now, we can use that Sim_0 is Goldreich-normal, to show via a hybrid argument as in Lemma F.6 that if $\mathcal{E}$ can distinguish RWS_0 and Sim_0 for "sequential Q-IND", there is a Q-IND distinguisher $\mathcal{D}$ for RWS_0 and Sim_0. And hence, there is a Q-IND distinguisher for RWS and Sim (since the constructed hybrid distinguisher $\mathcal{D}$ also sees no difference between Sim_0 and Sim (resp. RWS_0 and RWS)). Thus, "sequential Q-IND" holds.

Step 4 (Sim **is CEPT if** RWS **is)**: Now we make use of the runtime estimator θ. More precisely, we extend θ to the sequential setting by applying the underlying θ for each invocation separately, and taking the sum of the estimates. It is easy to see that this preserves efficiency, lower-bounding and upper-bounding.

Let $(\mathcal{E}, \mathcal{V}^*)$ be a CEPT adversary. Since $\mathsf{time}_{\mathsf{RWS}}(\langle \mathcal{E}, \mathsf{rep}(\mathsf{RWS}^{\mathcal{V}^*})\rangle)$ is CEPT, so is θ (by lower-bounding of $\mathsf{RWS}^{\mathcal{V}^*}$). Since $\theta(z_{\mathsf{RWS}})$ is CEPT for $z_{\mathsf{RWS}} = \mathsf{qseq}_{\mathcal{V}^*}(\mathsf{rep}(\mathsf{RWS}^{\mathcal{V}^*}))$ and since z_{RWS} and $z_{\mathsf{Sim}} = \mathsf{qseq}_{\mathcal{V}^*}(\mathsf{rep}(\mathsf{Sim}^{\mathcal{V}^*}))$ are indistinguishable w.r.t. $\mathcal{E}$ (by "sequential Q-IND"), also $\theta(z_{\mathsf{Sim}})$ is CEPT. Since θ upper-bounds the runtime of $\mathsf{time}_{\mathsf{Sim}}(\langle \mathcal{E}, \mathsf{rep}(\mathsf{Sim}^{\mathcal{V}^*})\rangle)$, $\mathsf{time}_{\mathsf{Sim}}(\langle \mathcal{E}, \mathsf{rep}(\mathsf{Sim}^{\mathcal{V}^*})\rangle)$ is CEPT.

Finally, since the time spent in $\mathcal{V}^*$ can be easily reconstructed from z (by emulating the execution), $\mathsf{time}_{\mathcal{V}^*}(\langle \mathcal{E}, \mathsf{rep}(\mathsf{RWS}^{\mathcal{V}^*})\rangle) \overset{c}{\approx} \mathsf{time}_{\mathcal{V}^*}(\langle \mathcal{E}, \mathsf{rep}(\mathsf{Sim}^{\mathcal{V}^*})\rangle)$ due to Q-IND.

All in all, replacing $\mathsf{rep}(\langle \mathcal{P}, \cdot\rangle)$ with $\mathsf{rep}(\mathsf{Sim}(\cdot))$ preserves CEPT.

Step 5 (Output quality): Our definition of Q-IND included the outputs, so zero-knowledge follows. □

The proof sketch should be interpreted as follows: Step 3 shows that Q-IND for A and B composes sequentially if B is Goldreich-normal for any polynomial query cutoff. (It uses Step 2, although somewhat indirectly.) Step 4 shows that runtime estimators compose sequentially. Taken together, query-benign composes sequentially. Lastly, (sequential) query-benign implies (sequential) zero-knowledge.

We remark that to prove Q-IND, for all of our examples, one essentially proves benignness as well.

Contents