# On (expected polynomial) runtime in cryptography

Michael Klooß*

We revisit the definition of *efficient* algorithms and argue, that the standard runtime classes, strict probabilistic polynomial time (PPT) and expected probabilistic polynomial time (EPT) are "unnatural" from a cryptographic perspective. They are not closed under indistinguishability.

Applied to EPT, this suggests *computationally* expected polynomial time (CEPT), the class of runtimes which are (computationally) indistinguishable from EPT. We analyse the behaviour of CEPT for zero-knowledge proofs and *designated adversaries* in the setting of *uniform complexity* (following Goldreich (JC'93)). A designated adversary is (only) efficient in the protocol it is designed to attack. This security notion, first proposed in Feige's thesis [Fei90], is very natural, but there are obstructions to achieving it.

Prior work on handling (designated) EPT adversaries by Katz and Lindell (TCC'05) requires superpolynomial hardness assumptions, whereas the work of Goldreich (TCC'07) requires "nice" adversarial behaviour under rewinding. We 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.* For example, $\mathcal{P}$ can convince $\mathcal{V}$ that a (deterministic) program $F$ outputs $x = F(w)$ for some *witness* $w$, without revealing anything about $w$. Thus, zero-knowledge proofs can guarantee *honest behaviour* (soundness) while *preserving privacy* (zero-knowledge), making them a foundational tool in cryptography.

The formulation of zero-knowledge relies on the simulation paradigm: It stipulates that, for every (malicious) verifier $\mathcal{V}^*$, there is a simulator $\mathsf{Sim}$ which, given only the inputs $x, \mathit{aux}$ of $\mathcal{V}^*$, can produce a *simulated* output (or *view*[1]) $\mathit{out} = \mathsf{Sim}(x, \mathit{aux})$, 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* $\mathsf{Sim}$ *and* $\mathcal{V}^*$ *lie in the same complexity class.*

There are two widespread notions of zero-knowledge: PPT simulation for PPT adversaries, and EPT simulation for PPT adversaries. The former satisfies the "promise of zero-knowledge", but comes at a price. Barak and Lindell [BL04] show that it is impossible to construct *constant* round proof system with *black-box* simulation (and negligible soundness error in the plain model). Since *constant round black-box*

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

[1] We use view and output synonymous in the introduction. The standard definition of a view consists of input, randomness, and received messages. With EPT, this choice is debatable, see Remark 5.7.

zero-knowledge is attractive for many reasons, the relaxation of zero-knowledge to *EPT* simulation for PPT adversaries is used. However, this asymmetry breaks the "promise of zero-knowledge". The adversary *cannot* execute $\mathsf{Sim}$, 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 to modularly construct 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 several sensible definitions of EPT adversaries, but the arguably most natural choice are *designated* EPT adversaries. That is, adversaries which need (only) be *EPT when interacting with the protocol they are designed to attack*. Feige [Fei90] first considered this setting, and demonstrates significant technical obstacles.

The problems of EPT (and designated adversaries) are not limited to zero-knowledge, and extend to the simulation paradigm and multi-party computation in general. However, in this work, we focus solely on zero-knowledge as a proxy.

**Preliminary conventions.** Throughout this work, $\kappa$ denotes the security parameter, and we generally consider objects which are families (of objects) parameterised by $\kappa$. We abbreviate *systems of (interactive) machines (or algorithms)* by *system*. A system is *closed*, if $\kappa$ is its only "input", i.e. if there are no "open interfaces" (except the output interface). 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 (almost closed) system are generated efficiently by so-called *input generating machines*. Interaction of algorithms $\mathsf{A}$, $\mathsf{B}$ is denoted $\langle \mathsf{A}, \mathsf{B} \rangle$, the time spent in $\mathsf{A}$ is denoted $\mathsf{time}_{\mathsf{A}}(\langle \mathsf{A}, \mathsf{B} \rangle)$, and similarly for time spent in $\mathsf{B}$ or $\mathsf{A} + \mathsf{B}$. Oracle access to $\mathcal{O}$ is written $\mathsf{A}^{\mathcal{O}}$.

## 1.1. Obstacles

Before diving into our approach, we 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 $T_\kappa$ over $\mathbb{N}$, where $\mathbb{P}(T_\kappa = 2^\kappa) = 2^{-\kappa}$ and $T$ is 0 otherwise. Then $T_\kappa$ 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 breaks the non-black-box simulation of Barak [Bar01] (which suffers from a quadratic growth in runtime), but also machine model independence of EPT as an efficiency notion.

**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 $\mathsf{A}$ 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 $\mathsf{A}$ 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

We demonstrate the problems and the develop the idea underlying our approach on a concrete example, the constant-round black-box zero-knowledge proof of Goldreich and Kahan [GK96].

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 D.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 $\psi$ be a 3-colouring of $G$. The prover is given $G, \psi$ and the verifier is given $G$.

**(P0)** The prover 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 randomised colourings for each of the $N$ parallel repetitions of the standard graph 3-colouring proof system, and sends the $\mathsf{Com}^{(\mathrm{H})}$-committed randomised 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 $\kappa$), 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.

$\mathsf{G}_0$ This is the real G3C protocol. The output is the real view.
$\mathsf{G}_1$ 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) (wheres the black-box has fixed randomness).
$\mathsf{G}_2$ If the two openings in (V1) differ, return `ambig`, indicating ambiguity of the commitment. Otherwise, proceed unchanged.
$\mathsf{G}_3$ 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 $\mathsf{G}_3$ outputs purported view independent of the witness. Thus, the simulator is defined as in $\mathsf{G}_3$: 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)), $\mathsf{Sim}$ aborts (as an honest prover would) and outputs the respective view. Otherwise, $\mathsf{Sim}$ rewinds the verifier to Step 2 and sends a pseudo-colouring (w.r.t. the previously revealed challenge) instead. $\mathsf{Sim}$ 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.

**$\mathsf{G}_0$ to $\mathsf{G}_1$.** The expected number of rewinds is at most 1. Namely, if $\mathcal{V}^*$ opens in (V1) with probability $\varepsilon$, then an expected number of $\frac{1}{\varepsilon}$ rewinds are required. Consequently, the expected runtime is polynomial (and $\mathsf{G}_1$ is EPT). The output distribution of the games is identical.

**$\mathsf{G}_1$ to $\mathsf{G}_2$.** It is easy to obtain an adversary against the binding property of $\mathsf{Com}^{(\mathrm{H})}$ which succeeds with the same probability that $\mathsf{G}_2$ outputs `ambig`. Thus, this probability is negligible.

**$\mathsf{G}_2$ to $\mathsf{G}_3$.** 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 (in all challenges). Thus, by security of the commitment scheme, $\mathsf{G}_2$ and $\mathsf{G}_3$ are indistinguishable.[2]

**A closer look.** The above proof is clear and simple. But the described simulator is not EPT! While $\mathsf{G}_2$ and $\mathsf{G}_3$ 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 random a random colouring, just like Sim, and a challenge edge $e$ which reveals that $c$ is not a 3-colouring. 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 its challenge only for $c$, but in that case the challenge edge $e$ exposes the simulation.

As described, $\mathcal{V}^*$ is a priori PPT, and indeed, the simulator in [GK96] uses a "normalisation 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. Indeed, this approach shows that designated *PPT* $\mathcal{A}$ are as simple to handle as a priori PPT adversaries.[3] However, there are EPT adversaries, or more concretely runtime distributions, where *any strict polynomial truncation* affects the output in the real protocol *noticeably*.[4] 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: Katz and Lindell [KL08] use *superpolynomial* truncation; Goldreich [Gol10] uses a restriction so that runtime does not explode under rewinding. See Section 1.6.)

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

[2] The commitments which are to be opened are known beforehand (since $\mathcal{V}$ committed to the challenge), or we abort by outputting `ambig`. Also, we rely on security of binding and hiding against *expected time* adversaries, which easily follows from PPT-security.

[3] However, such a simulator is not fully black-box anymore, as it depends on $\mathcal{A}$.

[4] There exist distributions $T$ over $\mathbb{N}_0$ such that $\mathbb{E}(T) < \infty$, but $\mathbb{E}(T^2) = \infty$: 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.

of runtime. If one could *distinguish* the runtime of $\mathsf{G}_2$ and $\mathsf{G}_3$, 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 an important detail: We solved the problem with the very strategy we claim to fix — different runtime classes for $\mathsf{Sim}$ and $\mathcal{V}^*$! Fortunately, $\mathsf{Sim}$ also handles CEPT adversaries in CEPT.

## 1.3. Computationally expected polynomial time

With sufficient motivation, we turn to a basic treatment of CEPT.

*Definition* (Oversimplified Definition 3.5). A runtime $S_\kappa$ (i.e. a family of random variables with values in $\mathbb{N}_0$) is **computationally expected polynomial time (CEPT)**, if there exists a runtime $T_\kappa$ which is (perfectly) expected polynomial time (i.e. EPT), such that $T \overset{c}{\approx} S$, i.e. any a priori PPT distinguisher has negligible distinguishing advantage for the distributions $T$ and $S$. The class of CEPT runtimes is denoted $\mathcal{CEPT}$. An analogous definition for PPT yields CPPT.

**Characterising CEPT.** At a first glimpse, CEPT looks like a hideous monster. It mixes computational assumptions into its very definition, mingling efficiency and indistinguishability; it seems far removed from typical settings. Fortunately, this is a mirage. We have following characterisation of CEPT.

**Proposition** (Informal Corollary 3.8)**.** *Let $T$ be a runtime. Then $T \in \mathcal{CEPT}$ if following equivalent conditions hold:*

*(1) $\exists S \in \mathcal{EPT}$ which is* computationally *PPT-indistinguishable from $T$.*
*(2) $\exists S \in \mathcal{EPT}$ which is* statistically *PPT-indistinguishable from $T$.*
*(3) There is a family of sets of good events $\mathcal{G}_\kappa$ with $\mathbb{P}(\mathcal{G}) \geq 1 - \varepsilon$ such that $\mathbb{E}(T_\kappa \mid \mathcal{G}_\kappa) = t$ (for the conditional expectation), where $\varepsilon$ is negligible and $t$ polynomially bounded.*

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

Thanks to this characterisation, 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.

**A beauty in disguise?** The characterisation reveals that CEPT is rather tame after all. However, it gets better (for some notion of better): CEPT distinguishers are no more powerful than PPT distinguishers, hence we could define CEPT using CEPT-indistinguishability instead! Thus, the runtime class $\mathcal{CEPT}$ is "closed under indistinguishability": Any runtime $S$ which is CEPT-indistinguishable from some $T \in \mathcal{CEPT}$ lies in $\mathcal{CEPT}$. Thus, CEPT has a nice intrinsic property, which PPT and EPT do not share.

*Example* 1.2 (Perfect and statistical properties). Let $\mathsf{A}$ be an algorithm which outputs 42 in $10^{10}$ steps. Modify it to $\mathsf{A}'$ which acts identical to $\mathsf{A}$, except with probability $2^{-\kappa}$, in which case it runs $2^{2\kappa}$ steps. Then $\mathsf{A}'$ is neither PPT nor EPT. Yet, no matter how much a (polynomially bounded) distinguisher tries, it will not succeed in distinguishing $\mathsf{A}$ and $\mathsf{A}'$ when given *timed black-box access.* That is, by observing the output and runtime of the black-box $\mathcal{O}$, it is not possible to tell $\mathsf{A}$ and $\mathsf{A}'$ apart. This holds even when $\mathcal{O}$ may be called repeatedly and adaptively! Thus, it is rather unexpected that $\mathsf{A}'$ is considered inefficient.

Indeed, a variation of the example w.r.t. *correctness* would lead to calling $\mathsf{A}'$ *statistically* correct. Our relaxation of EPT and PPT is such a relaxation from perfect to statistical properties. Viewing PPT and EPT as perfect notions of runtime, corresponding to perfect correctness, perfect hiding, perfect binding, perfect zero-knowledge, perfect soundness, etc., CPPT resp. CEPT are their statistical counterparts.

## 1.4. Technical overview and results

We give a short overview of our techniques, definitions, and results. Recall that we restrict our attention to runtimes of closed systems. 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 $T_\kappa$ over $\mathbb{N}_0$. We use *runtime* and *runtime distribution* synonymously.

### 1.4.1. The basic tools

In Appendix C we give an abstract treatment of runtime which is meant for the inclined reader. Even though we tried to give general definitions, the most important results are only proven for "algebra-tailed runtime classes"; this is a straightforward generalisation of the polynomial runtime setting. Therefore, we restrict to the polynomial setting in the following. For completeness, we sketch one general definition.

*Definition* (Informal). Let $\mathcal{T}$ be a runtime class. Then $\mathcal{T}$ is **(distinguishing-)closed**, if any runtime $S$ which is indistinguishable from a runtime $T \in \mathcal{T}$ by $\mathcal{T}$-time distinguishers lies in $\mathcal{T}$.

Note that, a priori, *computational* and *statistical* closedness need not coincide.

**Statistical vs. computational indistinguishability.** The *equivalence of statistical and computational indistinguishability* for distributions with *"small" support* is a simple, but central, tool in Appendix C. For polynomial time, the setting and proof is as follows: For any distribution with a polynomial support, say $\{0, \dots, \mathsf{poly}_1(\kappa)\}$, one can compute the empirical probabilities of all elements with precision $1/\mathsf{poly}$.[5] From this, it is easy to see that *computational* and *statistical* indistinguishability coincide for such distributions. By truncation arguments, we can maneuver into this setting. Indeed, let $(T_\kappa)$ be a runtime distribution (a sequence of distributions on $\mathbb{N}_0$). Since (by assumption) for any $\mathsf{poly}_0$ there is a $\mathsf{poly}_1$ such that $\mathbb{P}(T_\kappa > \mathsf{poly}_1(\kappa)) \leq \frac{1}{\mathsf{poly}_0(\kappa)}$, we can reduce distinguishability to the strictly polynomial support while preserving non-negligible statistical distance. For expected polynomial time, this assumed property follows from Markov bounds.

**Standard cutoff argument.** Another simple, yet central, tool is the *standard cutoff argument* (Section 4.2). It is the core tool to obtain *efficiency from indistinguishability*.

**Lemma 1.3** (Standard reduction to PPT)**.** *Let $\mathcal{D}$ be a distinguisher for two oracles $\mathcal{O}_0$, $\mathcal{O}_1$ (which may be distributions, or model an IND-CPA game, or ...). Suppose $\mathcal{D}$ has advantage at least $\varepsilon \geq \frac{1}{\mathsf{poly}_0}$ (infinitely often). Suppose furthermore that $\mathcal{D}^{\mathcal{O}_0}$ is CEPT with virtually expected time $(\mathsf{poly}_1, \delta)$. Then there is an a priori PPT distinguisher $\mathcal{A}$ with advantage at least $\frac{\varepsilon}{4} - \delta$ (infinitely often).*

We stress that there are *no runtime guarantees* for $\mathcal{D}^{\mathcal{O}_1}$ — it may never halt for all we know. For the proof, define $N = 4\mathsf{poly}_0 \cdot \mathsf{poly}_1$ 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 $\frac{\varepsilon}{4} + \delta$ close. For $\mathcal{A}^{\mathcal{O}_1}$ and $\mathcal{D}^{\mathcal{O}_1}$ this may be false. However, if the runtime for $\mathcal{D}^{\mathcal{O}_1}$ exceeds $N$ 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. Consequently, $\mathcal{A}$ has advantage $\frac{\varepsilon}{4} - \delta$.

**Working with CEPT.** The characterisation of CEPT now follows essentially from the computational-to-statistical reduction and a variant of Lemma 1.3. As is, the characterisation is not that useful, because the randomness of different parties, say the adversary and prover (resp. simulator) are "entangled". As an example, consider a prover which sends a random string $r$, and an adversary $\mathcal{A}$ which picks a random $s$ runs forever if $r = s$. Now, the "bad behaviour" cannot easily be attributed to $\mathcal{A}$. Unsurprisingly,

[5] Here, repeated sampling is central, which is for example satisfied if one can efficiently sample.

one can "disentangle" in the sense that bad behaviour only depends on messages exchanged, and not internal coin tosses of honest parties.

Lemma 4.21 essentially formalises the "disentanglement", and roughly states that for interacting algorithms $\langle \mathsf{A}, \mathsf{B} \rangle$, there is an "imaginary" modification $\mathsf{B}'$ (which need not be efficiently computable), which 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. Moreover, the time spent in $\mathsf{B}'$, i.e. $\mathsf{time}_{\mathsf{B}'}(\langle \mathsf{A}, \mathsf{B}' \rangle)$, is EPT. We stress that $\mathsf{B}'$ is "imaginary" and will be used via oracle-access only. It is merely a convenient tool to track the evolution of virtuality under actions such as rewinding.

### 1.4.2. Definitions and tools for zero-knowledge

We define auxiliary input zero-knowledge for efficiently generated input $(x, w, \mathit{aux}) \leftarrow \mathcal{G}(\kappa)$, and designated adversaries. That is, for an adversary $(\mathcal{G}, \mathcal{V}^*)$, $\mathsf{time}_{\mathcal{G}+\mathcal{P}+\mathcal{V}^*}(\langle \mathcal{P}(x,w), \mathcal{V}^*(x, \mathit{aux}) \rangle_{(x,w,\mathit{aux})\leftarrow\mathcal{G}(\kappa)})$ must be efficient, e.g. CEPT. We make no restrictions on $(\mathcal{G}, \mathcal{V}^*)$ beyond that.

**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: We introduce all rewinding steps as in $\mathsf{G}_1$. Here, virtually expected runtime and virtuality at most doubles. To show this, we use Lemma 4.21 to replace $\mathcal{V}^*$ with an "imaginary" $\mathcal{V}'$ which yields an EPT execution and outputs `timeout` for "bad" queries. Since Game $\mathsf{G}_1$ 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 expected runtime also (at most) doubles.

Step 2: We apply an indistinguishability transition, which reduces to hiding and binding properties of the commitment. From this, we obtain output quality of $\mathsf{Sim}$, as well as efficiency of $\mathsf{Sim}$. Concretely, this follows by an application of the standard reduction (to PPT).

We abstract this proof strategy to cover a large class of zero-knowledge proofs. We split a simulator into two phases, a rewinding phase and an indistinguishability phase. 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).[6]

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

Firstly, a normal rewinding strategy outputs an adversarial view which is identically distributed to the real execution. Secondly, there is some $\mathsf{poly}$ so that for any adversary

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

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

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

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

[6] We significantly deviate from [KL08], to obtain easier to quantify and generally simpler reductions. For completeness, we demonstrate in Appendix E that an approach based on "query indistinguishability" is feasible.

This follows because a bad query in the interaction $\langle \mathcal{P}, \mathcal{V}^* \rangle$ (a `timeout` of the "imaginary" $\mathcal{V}'$ from Lemma 4.21) is asked by $\mathsf{RWS}^{\mathcal{V}^*}$ with probability at most poly-fold higher.

**Lemma** (Informal)**.** *Fix and suppress some input distribution (generated by an input generating machine). Let* RWS *be a normal rewinding strategy for* $(\mathcal{P}, \mathcal{V})$ *with runtime and probability tightness* poly*. If* $\langle \mathcal{P}, \mathcal{V}^* \rangle$ *is CEPT with virtually expected time* $(t, \varepsilon)$*, then* $\mathsf{RWS}(\mathcal{V}^*)$ *is CEPT with virtually expected time* $(\mathsf{poly} \cdot t, \mathsf{poly} \cdot \varepsilon)$*. Moreover,* $\mathsf{out}_{\mathcal{V}^*}\langle \mathcal{P}, \mathcal{V}^* \rangle \equiv \mathsf{RWS}(\mathcal{V}^*)$*, i.e. the output is identically distributed.*

**A hidden requirement (Relative efficiency).** A property which is hardly visible in the concrete setting, but necessary for our abstract results, is *relative efficiency* of (oracle) algorithms. Roughly 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 $(t, \varepsilon)$-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. Note the similarities with the lemma for normal rewinding strategies. Indeed, runtime tightness of RWS corresponds to $\mathsf{poly}_{\mathsf{time}}$ and probability tightness to $\mathsf{poly}_{\mathsf{virt}}$.

**Abstracting Step 2 (Simple assumptions).** To move from a rewinding strategy RWS to a bb-rw simulator Sim, we argue by a reduction to a **simple assumption**. Roughly, 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. For example, hiding resp. binding for a commitment scheme are simple assumptions. In fact, a merge of hiding and binding (as used in Step 2) is also a simple assumption.

The indistinguishability of $\mathsf{RWS}^{\mathcal{V}^*}$ and $\mathsf{Sim}^{\mathcal{V}^*}$ is reduced 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 $\mathrm{R}^{\mathcal{C}_1}(\mathcal{V}^*)$. The proof for $\mathsf{out}_{\mathcal{V}^*}\langle \mathcal{P}, \mathcal{V}^* \rangle \overset{c}{\approx} \mathsf{Sim}^{\mathcal{V}^*}$ is as follows: 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 $\mathrm{R}^{\mathcal{C}_1}(\mathcal{V}^*)$, $\mathsf{Sim}^{\mathcal{V}^*}$ is CEPT and the output is indistinguishable from $\mathsf{RWS}^{\mathcal{V}^*}$, which is distributed as $\mathsf{out}_{\mathcal{V}^*}\langle \mathcal{P}, \mathcal{V}^* \rangle$ by normality of RWS.

**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.

**Lemma** (Informal Lemma 5.35)**.** *Proof systems with* benign *simulators are* auxiliary-input zero-knowledge *against CEPT adversaries. In particular, benign simulators handle CEPT adversaries in CEPT.*

*Proof summary.* The outlined strategy above can be summarised symbolically:

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

An important point is that the right algorithm in any "$\equiv$" above is efficient relative to the left one, and that $\mathrm{R}^{\mathcal{C}_b}(\mathcal{V}^*)$ are both CEPT if one is, since CEPT is preserved under indistinguishability transitions. □

**Sequential zero-knowledge.** The observant reader may wonder why we require polynomial tightness bounds all over the place. Indeed, they are not necessary for auxiliary input zero-knowledge. However, due to a posteriori runtime and designated adversaries, we were unable to prove sequential composition results without having some means to bound the runtime *over multiple invocations* of a simulator. Consider a sequential repetition of a zero-knowledge proof. Suppose we only know that $\mathsf{Sim}^{\mathcal{V}^*}$ is CEPT if the real execution $\langle \mathcal{P}, \mathcal{V}^* \rangle$ is. Then we do not know how to prove (or disprove) that the runtime does not explode under sequential repetition of Sim. More concretely, suppose poly-fold repetition is not CEPT for some adversary. We would like to derive a contradiction (to output quality or efficiency of CEPT without repetition). Since we see no other means, we try a hybrid argument. Intuitively, at some point, there should be a "transition from efficiency to inefficiency" which we could

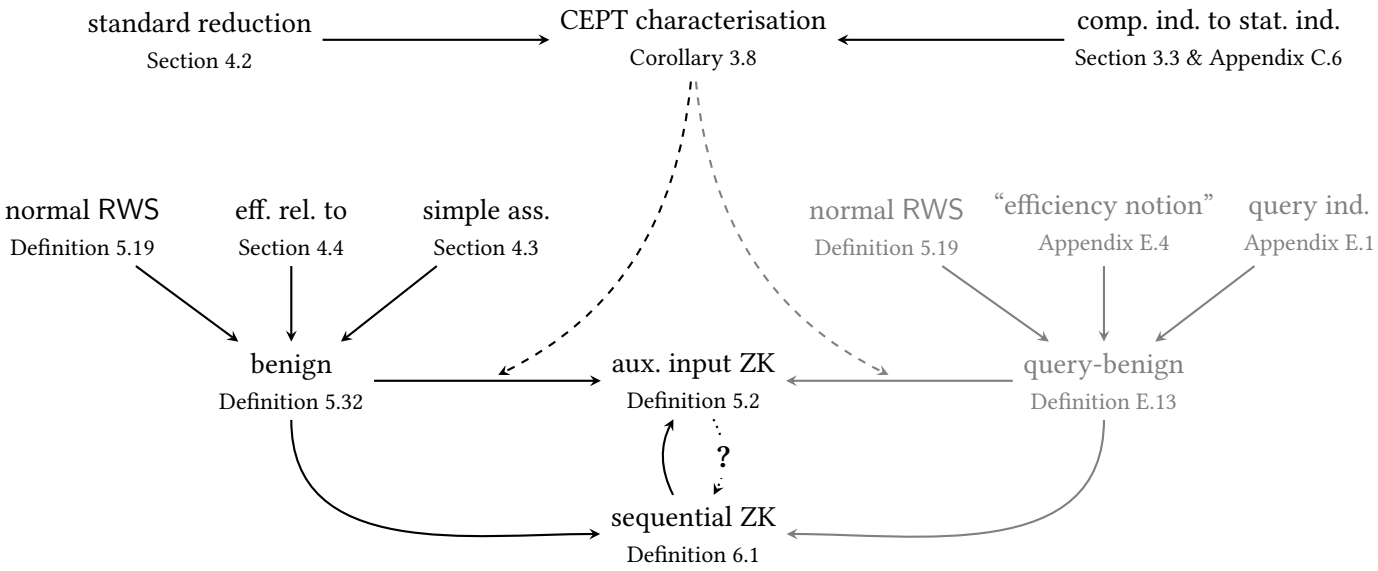


Figure 1: A rough overview of dependencies of results and definitions. The top line is implicitly present almost everywhere. Many intermediate non-core results are left out, e.g. Lemma 4.21 and size-guarding. The greyed out approach follows [KL08] more closely.

catch and distinguish. However, there are two big problems: We do not know when such a transition happens (or if there is a boundary at all); a naive hybrid distinguisher may thus be *inefficient*. Perhaps worse, while the CEPT characterisation ensures that indistinguishable from CEPT remains CEPT, but distinguishable from some CEPT runtime does not imply a runtime is not CEPT! (Case in point: Constant runtimes are easily distinguished, yet CEPT.)

The above shows that arguing efficiency of Sim under sequential composition is surprisingly non-trivial. Thus, we provide an explicit definition of sequential zero-knowledge, and prove that benign simulators do compose sequentially. This follows because, normal RWS and relative efficiency compose sequentially. Moreover, simple assumptions satisfy indistinguishability under "repeated trials" which translates to sequential composition of benign.

**Lemma** (Informal Lemma 6.4)**.** *Proof systems with* benign *simulators are* sequential zero-knowledge *against CEPT adversaries.*

**Size-guarded security.** Lastly, we mention a possible relaxation. In our definition of zero-knowledge, a simulator's runtime must be closely related to the prover's and adversary's runtime. In particular, Sim is allowed almost no overhead in $|x|$ (compared with $\mathcal{P}$), because fat tailed input distributions would render it inefficient. This can relaxed by *size-guarding* the protocol. This means that prover (and verifier) reject inputs which exceed the length of a (polynomial) size-guard gd. Size-guards "decouple" efficiency of simulator and prover w.r.t. $|x|$, simplify efficiency arguments, but slightly weaken security.

## 1.5. Contribution

Our main contribution is the reexamination of the notion of runtime in cryptography. We offer a novel, and arguably natural, 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, and give a convenient characterisation.

- To the best of our knowledge, this is the first work which embraces *uniform*[7] complexity, *a posteriori* efficiency, *expected* time, and *designated* adversaries and consequently develops tools for zero-knowledge in this setting. Our tools include, normal rewinding strategies, relative efficiency, simple assumptions,[8] and benign simulators.
- Easy-to-check criteria show that many (all known?) black-box zero-knowledge proof systems from standard assumptions in the plain model[9] have CEPT simulators which handle designated CEPT adversaries. Consequently, security against designated adversaries is natural.
- We impose no (non-essential) restrictions on the adversary, nor do we need additional (hardness) assumptions. See Section 1.6 for a comparison with prior work.

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.

Overall, this work proposes a new notion of efficiency and demonstrates its viability for zero-knowledge. It stops short of covering the full real-ideal paradigm, let alone environmental security. It also leaves open important compatibility questions with superpolynomial hardness assumptions. See Section 7, where we discuss some open problems. Tackling these (more complex) settings now seems premature and detrimental for comprehensibility, which already suffers from the technical issues we face.

**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.[10] 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 $\kappa$. 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 $\mathsf{Sim}$ 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).

Due to our focus, we give no (general) definitions of these complexity classes in the real-ideal setting, but only their specialisations to zero-knowledge, Definitions 5.2 and 6.1.

## 1.6. Related work

We are aware of three (lines of) related works: 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 $\kappa = |x|$. Our notion of size-guarding mirrors this weakened security (as it prevents problems of *expected* polynomial size inputs).

**Comparison with [KL08].** The work of Katz and Lindell [KL08] attacks 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

[7] Our results are applicable to a minor generalisation of the non-uniform setting as well, namely non-uniformly generated input *distributions*, see Appendix F.7.

[8] Simple assumptions are essentially falsifiable assumptions in the sense of [GW10]. However, our setting is not an impossibility setting, and one can allow broader classes of assumptions. We do not pursue that, since all of our examples only rely on simple assumptions.

[9] Unfortunately, problems might arise with superpolynomial hardness assumptions, see Section 7.

[10] A sensible complexity class for protocols should compose well in any situation, hence be far more restrictive than the class of feasible attacks. Indeed, even the machine model may be different. Protocols should be *uniform (classical)* computations, even when assuming *non-uniform* hardness (or *quantum* adversaries).

function $\alpha$ and have to assume security of primitives w.r.t. (strict) $\alpha$-time adversaries. Squinting hard enough, their approach as 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".

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.[11] [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]. Our approach of sequential security may close this gap, as protocols can be replaced by simulation en bloc, which gives a new CEPT adversary per protocol replacement.

**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 $\kappa$ and all $C \geq 0$. Equivalently, $\mathbb{E}(T_\kappa^\gamma)$ is polynomially bounded (in $\kappa$) for some $\gamma > 0$. Introducing negligible "errors" relaxes the notion further. This definition fixes the composition problems of expected polynomial time. But arguably, this stretches what is considered efficient far beyond what one may be willing to accept. Indeed, runtimes whose expectation is "very infinite" are considered efficient.[12] 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.

(For completeness, we note that we are not aware of work on designated adversaries in this setting.)

**More related work.** Hofheinz, Unruh, and Müller-Quade [HUM13] define *PPT with overwhelming probability (w.o.p.)* — essentially CPPT — but treat good *compositional* properties in the setting of universal composability. Goldreich [Gol11a] defines *typical efficiency* similar to CPPT (resp. PPT w.o.p.), although in the setting of complexity theory. As the relaxations for strict bounds is very straightforward, we suspect more works using CPPT variations for a variety of reasons.

Halevi and Micali [HM98] define a notion of efficiency for extractors, which may be viewed as a (special case of) relative efficiency.

## 1.7. Structure of the paper

In Section 2, we clarify preliminaries, such as (non-)standard (notational) conventions and terminology. We also state some basic concepts and results. The definition of CEPT and its characterisation is in Section 3. In Section 4, we build some first tools for working with CEPT. In Section 5, we finally apply CEPT to zero-knowledge. We define (uniform complexity auxiliary input) zero-knowledge, rewinding strategies, query indistinguishability, benign simulation. Lastly, in Section 6, we define sequential zero-knowledge and prove that benign simulators satisfy it as well. We conclude in Section 7.

In Appendix A, we give a detailed discussion on the effect of machine models and their (in)compatibility with expected time. Appendix B merely contains some simple but useful results for our

[11] 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.

[12] For example, the distribution $X = X_\kappa$ in Footnote 4 with $c = 2$ and $\gamma = 3$ has expectation $\sum_n n$, but is considered efficient. (The limit $-\frac{1}{12}$ is not applicable here.)

general discussion of runtime classes in Appendix C. Appendix D contains supplementary definitions for commitment schemes. For completeness, we show in Appendix E 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. Lastly, in Appendix F, we discuss many points of lesser importance, give examples or fill in some details.

See page 78 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 $\kappa$; it is often suppressed. Similarly, we often speak of an object $X$, instead of a family of objects $(X_\kappa)_\kappa$ parameterised by $\kappa$. By $\mathsf{Dists}(X)$ we denote the space of probability distributions on $X$.[13] We write $X \sim Y$ if a random variable $X$ is distributed as $Y$. For random variables $X, Y$ over a (partially) ordered set $(A, \leq)$ we write $X \leq Y$ if $\mathbb{P}(X \leq a) \geq \mathbb{P}(Y \leq a)$ for all $a \in A$ and say $Y$ *dominates* $X$ (or is greater than $X$ in distribution). We use the same notation for families of random variables, i.e. we write $X \leq Y$ and mean $X_\kappa \leq Y_\kappa$ for all $\kappa$. 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. $X|_{\bot \mapsto 0}$ or $X|_{S \mapsto 0}$ or $X|_{\geq N \mapsto N}$.

We define statistical distances $\Delta_p(\rho, \sigma)$ of distributions (i.e. measures) $\rho, \sigma$ over a countable set $\Omega$ as $\Delta_p(\rho, \sigma) = \frac{1}{2}(\sum_{x \in \Omega} |\rho(x) - \sigma(x)|^p)^{1/p}$, where $\rho(x) := \rho(\{x\})$ is the probability for $x$ under $\rho$ and likewise for $\sigma$. For $p = \infty$ this is $\frac{1}{2} \sup_{x \in \Omega} |\rho(x) - \sigma(x)|$. 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 $\rho$ over $\sigma$; $\rho$ and $\sigma$ may be arbitrary non-negative functions.

With $\mathsf{poly}$, $\mathsf{polylog}$, and $\mathsf{negl}$ we denote polynomial, polylogarithmic and negligible functions (in $\kappa$) respectively. Usually, we (implicitly) assume that $\mathsf{poly}$, $\mathsf{polylog}$, and $\mathsf{negl}$ are *monontone*. A function $\mathsf{negl}$ is (polynomially) negligible if $\lim_{\kappa \to \infty} \mathsf{poly}(\kappa)\mathsf{negl}(\kappa) = 0$ for every polynomial $\mathsf{poly}$. In many definitions, we assume the existence of a negligible bound $\mathsf{negl}$ 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 "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.[14]

## 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; this ensures 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. More precisely, EPT also needs a suitable strict runtime bound, e.g. $2^\kappa$, but CEPT not (due to virtuality).

[13] We assume that $X$ has some (obvious) associated $\sigma$-algebra and consider only probability distributions w.r.t. this $\sigma$-algebra.

[14] 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 $\mathsf{negl}$, c.f. [Bel02]. This is false for $\leq$. Indeed, $\leq_{\text{ev}}$ behaves unintuitive. Consider a sum of a growing number (in $\kappa$) of negligible functions $\nu_i$. It is well-known that $\mu(\kappa) := \sum_{i=1}^{\kappa} \nu_i(\kappa)$ need not be negligible, even if all $\nu_i$ are negligible. But if all $\nu_i$ are "strictly dominated" by some $\nu$, i.e. $\nu_i \leq \nu$, then $\mu(\kappa) \leq \kappa\nu(\kappa)$ hence $\mu$ is negligible. However, if all $\nu_i$ 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 $\mu$ need not be negligible. Concretely, $\nu = 0$ *eventually* dominates all $\nu_i$, yet $\mu(n) = 1 > 0 = n\nu(n)$. In conclusion, $\leq_{\text{ev}}$ has a rather counterintuitive behaviour, and should be used with care.

**Systems, algorithms and oracles.** We always consider (induced) systems, which offer **interfaces** for (message-based) communication. Input and output are modelled as interfaces as well, the security parameter is an implicit input interface of (almost) every system. A **system** is a "mathematical" object, which defines (probabilistic) behaviour of the offered interfaces. An **algorithm** is given by *code*, a *finite*[15] 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 emphasise 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 is a system, for which some notion of purported elapsed runtime is defined. 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 primarily as a convenient abstraction for defining unconditional properties, e.g. EPT in any interaction. Timeful oracles can also be timed in the obvious manner.

**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 $\mathsf{A}_1$ and $\mathsf{A}_2$, 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 $\mathsf{A}_i$ for $i = 1, 2$. We write $\mathsf{A}^{\mathcal{O}}$ for an algorithm (or system) $\mathsf{A}$, with access to an oracle $\mathcal{O}$ (where $\mathcal{O}$ may be a subroutine, e.g. a commitment scheme). This notation emphasises, that the output of the system is that of $\mathsf{A}$. Otherwise, the system is equivalent to $\langle \mathsf{A}, \mathcal{O}\rangle$, or even $\mathcal{O}^{\mathsf{A}}$. In particular, we view interaction, oracle, and subroutine calls as essentially identical and use the notation interchangeably if no confusion arises. **Black-box rewinding (bb-rw) access** to an algorithm $\mathsf{A}$ (or timeful system) means access to an oracle emulating $\mathsf{A}$ with fresh but fixed randomness, which allows to feed $\mathsf{A}$ messages and rewind it to any visited state. For simplicity, we model this as a $\mathsf{NextMsg}_{\mathsf{A}}$ function, which upon a query $(m_1, \ldots, m_n)$ returns the result of $\mathsf{A}$ when given $m_i$ as its $i$-th message.[16] A **timed** bb-rw oracle truncates and returns the runtime of its emulated program.

## 2.3. Preliminary remarks on runtime

The full discussion of runtime is in Appendix C, and meant for the inclined reader. 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 $\mathsf{A}$, we write $\mathsf{time}_{\mathsf{A}}(\mathsf{A}^{\mathcal{O}})$ for the time spent in $\mathsf{A}$ (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 other systems built from interacting machines. If not all randomness is fixed, a runtime $T$, such as $T = \mathsf{time}_{\mathsf{A}}(\mathsf{A}^{\mathcal{O}})$, is a *random variable*, or more precisely, a sequence of random variables $T_\kappa$ parameterised by $\kappa$. We assume that an oracle call is a single step and 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* 2.1 (Preliminary definitions). A **runtime (distribution)** $T$ is a family of random variables (resp. distributions) over $\mathbb{N}_0$ parameterised by the security parameter $\kappa$. We (only) view a runtime as a random variable $T_\kappa \colon \Omega_\kappa \to \mathbb{N}_0$, when stochastic dependency is relevant.

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

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

[16] There are technical problems with the efficiency of this approach, which we ignore here. They can be solved in a straightforward way, see Appendix A.

The default notion of runtime for an oracle algorithm $\mathsf{A}^{\mathcal{O}}$ is oracle-excluded-time, but we usually specify exactly what is considered.

Our definition of runtime does not take inputs into account. Thus, efficiency depends only on $\kappa$. In particular, we do not assign a stand-alone runtime to a non-closed system, e.g. an algorithm $\mathsf{A}$ which needs inputs (resp. oracle access, resp. communication partners). The exception to the rule are *a priori PPT resp. EPT* algorithms $\mathsf{A}$, for which there is a bound $\mathsf{poly}$ 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 (resp. oracles, parties).

*Remark* 2.2. Since our notion of efficiency is asymptotic in the security parameter alone, we do not pass around $1^\kappa$ as "fuel" for algorithms. Still every algorithm is given $\kappa$ as (implicit) input.

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

*Definition* 2.3 (Runtime truncation). Let $\mathsf{A}$ be an algorithm. We define $\mathsf{A}^{\leq N}$ as the algorithm which executes $\mathsf{A}$ up to $N$ steps, and then returns $\mathsf{A}$'s output. If $\mathsf{A}$ did not finish in time, return `timeout`.

Lastly, we offer a warning.

*Remark* 2.4. We warn the reader to be wary about the interaction of machine models and runtime notions. Technical problems pop up easily, yet seem to be just that — technical. The most prominent example is the failure of the "next-message" definition of black-box rewinding access. In Appendix A, such problems are discussed in more detail.

## 2.4. Probability theoretic conventions

The underlying probability space is usually denoted by $\Omega$. We neglect measurability questions because they do not pose any problems and are merely trivial technical overhead, see Appendix F.9 for a brief discussion.

We allow product extension of $\Omega$ 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 $\Omega$ are lifted implicitly and we again write $\Omega$ instead of $\Omega'$. Let $\mathbb{N}_0 \cup \{\infty, \mathtt{timeout}\}$ be totally ordered via $n < \infty < \mathtt{timeout}$ for all $n \in \mathbb{N}_0$.

*Definition* 2.5 ($\nu$-quantile cutoff). Let $T$ be a distribution on $\mathbb{N}_0 \cup \{\infty\}$ and $\nu > 0$. Suppose that $\mathbb{P}(T = \infty) \leq \nu$.[17] The (exact) **$\nu$-quantile (cutoff)** $T^\nu$ is following distribution on $\mathbb{N}_0 \cup \mathtt{timeout}$. Let $\mathsf{CDF}_T(\,\cdot\,)\colon \mathbb{N}_0 \cup \{\infty\} \to [0,1]$ be the CDF of $T$. Then $\mathsf{CDF}_{T^\nu}(\,\cdot\,)\colon \mathbb{N}_0 \cup \mathtt{timeout} \to [0,1]$ is defined by $\mathsf{CDF}_{T^\nu}(n) = \max\{1-\nu, \mathsf{CDF}_T(n)\}$ for $n \in \mathbb{N}$, and $\mathsf{CDF}_{T^\nu}(\infty) = \lim_{n\to\infty} \max\{1-\nu, \mathsf{CDF}_T(n)\}$, hence $\mathbb{P}(T^\nu = \infty) = 0$, and $\mathsf{CDF}_{T^\nu}(\mathtt{timeout}) = 1$,

An exact $\nu$-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 $\nu$, let $T^\nu \coloneqq T|_{>N \mapsto \mathtt{timeout}}$. Else, pick a (measurable) subset of $A = \{\omega \in \Omega \mid T(\omega) = N\}$ of probability $\nu - \nu'$, and let $T^\nu \coloneqq T|_{A \mapsto \mathtt{timeout}}$. If necessary, modify $\Omega$. So we assume w.l.o.g. that there is such a set of events. An *approximate $\nu$-quantile cutoff* with *error* $\delta$ is an exact $\nu'$-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 `timeout`. By modifying $\Omega$ 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* 2.6 (Equal-unless). If $X, Y\colon \Omega \to \mathcal{S}$ are random variables over $\Omega$ and coincide (as functions), except for an event $\mathcal{E} \subseteq \Omega$, then $X$ and $Y$ are **(pointwise) equal unless** $\mathcal{E}$. Typically, $\mathcal{E} = \{\omega \mid Y(\omega) = \mathtt{bad}$ (for some symbol `bad`)), and we say $X$ equals $Y$ unless `bad` happens. We also say $X$ and $Y$ *coincide unless* (or *agree except*) if `bad` happens. The definition extends to oracles in the obvious manner.

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

The relaxed notion of (distributionally) **equal unless** is defined as follows: Let $X$ and $Y$ be two random variables and let $\mathtt{bad}$ be a symbol that only $Y$ outputs. We say $X$ and $Y$ are equal unless $\mathtt{bad}$, if $\Delta(X,Y) \leq \mathbb{P}(Y = \mathtt{bad})$. That is, $Y$ can be changed into $X$ by modifying the distribution only on $\mathtt{bad}$. In other words: We can view $X$ and $Y$ as generated by a random variables $Z$, $Z_X$, $Z_Y$, as follows: With probability $1-\varepsilon = \mathbb{P}(Y \neq \mathtt{bad})$ output $Z$, else output $Z_X$; this is distributed like $X$. If instead we output $Z_Y$, it is distributed like $Y$.

Let $\mathcal{O}$ and $\mathcal{O}'$ be two oracles and let $\mathtt{bad}$ be a symbol that only $\mathcal{O}'$ outputs. Suppose that, if $\mathcal{O}'$ output $\mathtt{bad}$, it halts (and the answer to any follow-up query is $\mathtt{bad}$ by definition). Denote by $X(\vec{m})$ the $k$-th output of $\mathcal{O}$ given $m_0, ..., m_k$ as queries. Denote by $Y(\vec{m})$ the same for $\mathcal{O}'$. We say $\mathcal{O}$ and $\mathcal{O}'$ are **equal unless** $\mathtt{bad}$, if for all query sequences $\vec{m}$, $X(\vec{m})$ and $Y(\vec{m})$ are equal unless $\mathtt{bad}$.

### 2.4.1. 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* 2.7 (Tail bounds). Let $X$ be some distribution on $\mathbb{R}_{\geq 0}$. We call a 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 $\mathsf{bnd}$ of some sequence $(x_n)_n$ a tail bound, i.e. $x_n \leq \mathsf{bnd}(n)$ for all $n$.

Tail bounds generalise 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 $\mathsf{CDF}_X$ is the cumulative distribution function of $X$. We use $\mathsf{tail}^\dagger(\alpha)$ to conveniently denote the minimal $n_\alpha$ with $\mathsf{tail}(n_\alpha) \leq \alpha$, which exists due to continuity of $\mathsf{tail}$.

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

**Lemma 2.8** (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 $\mathcal{L}_p$-norms, i.e. $\|X\|_p = (\mathbb{E}(X^p))^{1/p} \leq t$, 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 B.2.

## 2.5. Oracle-indistinguishability

The (in)distinguishability of oracles (or systems) is a folklore abstraction. "Bit-guessing" experiments, or more generally game-based security notions can be straightforwardly rephrased as an oracle pair, see Appendix F.2. Depending on the oracles (or systems) and their interfaces, distinguishing can encompass (adversarial) input generation, protocol runs, and more. For example, an oracle present an IND-CPA game for public key encryption, or it may present the distinguisher with a concurrent zero-knowledge setting.

*Definition* 2.9 (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 systems $\mathcal{D}^{\mathcal{O}_b}$. The **(one-shot) 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 $\mathcal{T} \in \{\mathcal{PPT}, \mathcal{CPPT}, \mathcal{EPT}, \mathcal{CEPT}\}$. Then $\mathcal{O}_0$ and $\mathcal{O}_1$ are **computationally (one-shot) 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 $b = 0, 1$)[18] there is some negligible $\mathsf{negl}$ such that $\mathrm{Adv}^{\mathrm{dist}}_{\mathcal{D},\mathcal{O}}(\kappa) \leq \mathsf{negl}$. We define *statistical* indistinguishability by counting only oracle-queries as runtime.

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

*Definition* 2.10. Oracles $\mathcal{O}_0$, $\mathcal{O}_1$ (or systems, or algorithms), for which any (statistical unbounded) distinguisher has advantage 0 are called **perfectly indistinguishable**. We also write $\mathcal{O}_0 \equiv \mathcal{O}_1$ to emphasise this.

### 2.6. Query-sequences

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

*Definition* 2.11 (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 $\mathsf{A}$ 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 C for polynomial runtime classes, and have a first glimpse of the behaviour of CEPT. The inclined reader may wish to continue with Appendix C instead; it deals with runtime classes in more generality.

Before defining CEPT, we take a brief look at a handy formalisation.

### 3.1. Virtually expected time

We are interested in properties, which need not hold in any event. It is sufficient that these properties hold with overwhelming probability. We formalise 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 $\varepsilon$-**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 $\varepsilon$-**virtually** $t$-**time** if $T$ has $\varepsilon$-virtual expectation bounded by $t$. We abbreviate this as **virtually expected** $(t, \varepsilon)$-**time** and call $\varepsilon$ the **virtuality** of time $(t, \varepsilon)$. Finally, if we do not specify $\varepsilon$, then $\varepsilon$ is a negligible function, that is, **virtually expected** $t$-**time** means $\mathsf{negl}$-virtually $t$-time for some $\mathsf{negl}$.

The definition of virtual properties has a "probably approximately" flavour. It is closely related to "$\varepsilon$-smooth properties", such as $\varepsilon$-smooth min-entropy, which smudge over statistically close random variables (instead of conditioning).[19] 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, as long as we condition on *overwhelming* events $\mathcal{G}$, polynomially bounded expectation $\mathbb{E}(X)$ is preserved. In fact, it suffices that $\mathcal{G}$ is noticeable. Thus, consecutive restrictions of $\Omega$ are unproblematic.

[18] This is equivalent to being efficient in the respective distinguishing experiment, see Definition C.13 or Appendix F.2.
[19] We borrowed the terminology of virtual properties from group theory.

*Remark* 3.3 (Virtual properties of distributions). A distribution $\nu$ with density $\rho \leq \alpha^{-1}$ w.r.t. $t$ is a **subdistribution** of $t$ of **weight** $\alpha$. (This abstracts conditional probability distributions.) A distribution $t$ on $\mathbb{R}_{\geq 0}$ has $\varepsilon$**-virtual expectation (bounded by)** $t$ if there is a subdistribution $t'$ of weight $1-\varepsilon$ with expectation bounded by $t$.

## 3.2. A brief recap

We briefly recap the essentials of Appendix C. There, we rely on *triple-oracle indistinguishability*, instead of one-shot indistinguishability of runtime distributions. This abstracts technical details and prevents technical problems.

*Definition* 3.4 (Informal). 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**, if any $\mathcal{T}$-time distinguisher has advantage $o(1)$. If $\mathcal{T}$ contains $\mathcal{PPT}$, then (by amplification) any distinguisher has negligible advantage. We write $T \overset{c}{\approx}_{\mathcal{T}} S$ for **computationally triple-oracle $\mathcal{T}$-time indistinguishable** runtimes. For statistical triple-oracle indistinguishability, we only count oracle queries as a step.

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 $S \in \mathcal{T}$.

In the definition, we sketched our approach for general runtime classes (namely requiring $o(1)$ advantage bound). From now on, we specialise 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 $X_0, X_1$. From a PPT distinguisher $\mathcal{D}'$ with non-negligible advantage, one can construct, by a standard hybrid argument, a PPT distinguisher $\mathcal{D}$ which needs only a single challenge sample. If the distributions $X_0$ and $X_1$ are *efficiently* samplable, then, since $\mathcal{D}$ can sample $X_0, X_1$ itsself, we see that triple-oracle indistinguishability is equivalent to standard *one-shot* indistinguishability (where the distinguisher is given one sample of $X_0$ or $X_1$ as input, see Section 2.5).

Applying the above reduction concretely to runtimes $T_0$, $T_1$ which are induced by (emulation of) an algorithm, reveals a problem: If $T_0$ or $T_1$ were *not efficient*, e.g. superpolynomial, then they cannot be efficiently sampled by emulation! Thus, the reduction from triple-oracle to one-shot indistinguishability does not work trivially.

Fortunately, given an algorithm $\mathsf{A}$, its induced runtime $T = \mathsf{time}_{\mathsf{A}}(\mathsf{A})$ can be sampled "continuously", i.e. emulation of $N$ steps incurs only $N$ steps plus emulation overhead. By an appropriate runtime truncation, we can preserve efficiency. We detail this below for CEPT.

## 3.3. Characterising CEPT

We begin with the fundamental definition of this section.

*Definition* 3.5 (CEPT and CPPT). The runtime class $\mathcal{CEPT}$ of **computationally expected polynomial time** contains all runtimes which are (triple-oracle) $\mathcal{PPT}$-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 PPT-indistinguishable.

The runtime class $\mathcal{CPPT}$ of **computationally (strict) probabilistic polynomial time** is defined analogously.

The use of triple-oracle indistinguishability in Definition 3.5 is required for consistency with our general treatment of runtime, see Appendix C. For concrete applications, we want to get rid of it. We do this in a sequence of lemmata.

**Lemma 3.6.** *Suppose $S$ and $T$ are runtimes and $T \in \mathcal{CEPT}$. Then statistical and computational triple-oracle indistinguishability coincide. Moreover, a priori PPT distinguishers are sufficient.*

*Proof sketch.* 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* 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 B.3. Consequently, we can construct a (computational) PPT distinguisher if $T$ and $\widetilde{T}$ are not statistically 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 $\Delta(T^{\leq N}, S^{\leq N}) \geq \frac{\delta}{4}$.[20] 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. □

We say that $\mathcal{PPT}$ is *distinguishing-dense* (*d-dense*) in $\mathcal{CEPT}$ for runtime distributions, which means that if (runtime) distributions can be (one-shot) distinguished in CEPT then they can be (one-shot) distinguished in PPT.

**Lemma 3.7.** *Suppose $T$ and $S$ are runtimes induced by algorithms* A, B. *Moreover, suppose $T \in \mathcal{CEPT}$. Then triple-oracle and one-shot PPT-indistinguishability coincide.*

*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 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.[21]

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

**Corollary 3.8** (Characterisation of CEPT)**.** *Let $T$ be a runtime. The following conditions are equivalent:*

*(0) $T$ is in $\mathcal{CEPT}$.*
*(1) $T$ is ($\mathcal{PPT}$-time) computationally indistinguishable from some $\widetilde{T} \in \mathcal{EPT}$.*
*(2) $T$ is ($\mathcal{PPT}$-time) 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 $\mathbb{E}(T_\kappa \mid \mathcal{G}) \leq \mathsf{poly}(\kappa)$.*

[20] 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 B.3.

[21] 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.

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

$$\mathsf{tail}_{T_\kappa}(N) \leq \frac{\mathsf{poly}(\kappa)}{N} + \mathsf{negl}(\kappa)$$

*for* poly *and* negl *as in (3). Consequently, $\mathcal{PPT}$ is d-dense in $\mathcal{CEPT}$ and one can use $\mathcal{CEPT}$-time distinguishers in the above. Thus, $\mathcal{CEPT}$ is a closed runtime class and in fact 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, triple-oracle indistinguishability and standard one-shot indistinguishability coincide.*

*The analogous characterisation and properties hold for CPPT.*

The essence of Corollary 3.8 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 3.8.* 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. Say the statistical distance is $\delta$. Let $T^\nu$ be the respective $\nu$-quantile of $T$. Clearly, $T^\varepsilon|_{\texttt{timeout}\mapsto 0}$ minimises 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 $\delta$-close EPT $S$. Hence, we have $\mathbb{E}(T^\delta|_{\texttt{timeout}\mapsto 0}) \leq$ poly. Consequently $\mathbb{E}(T^\delta \mid \neg\texttt{timeout}) \leq \frac{1}{(1-\delta)}$poly, and the claim follows.

The converse is trivial: If $\mathbb{E}(T \mid \mathcal{G}) \leq$ poly for an event $\mathcal{G}$ of overwhelming probability $1 -$ negl, then $\widetilde{T} = T|_{\bar{\mathcal{G}}\mapsto 0}$ 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 $\Delta(T,\widetilde{T}) \leq$ negl. Thus, the tail bound follows immediately from Markov's bound (Lemma 2.8) applied to $\widetilde{T}$ and statistical distance of negl. That $\mathcal{PPT}$ is d-dense in $\mathcal{CEPT}$ is straightforward given the tail bound. Item (1) and d-density of $\mathcal{PPT}$ imply that $\mathcal{CEPT}$ is closed.

Finally, Lemma 3.7 demonstrates the equivalence of triple-oracle and one-shot distinguishing. □

# 4. Towards applications

Before applying CEPT, we make some global conventions, clarify our setting of uniform complexity with input generating machines, point out some (ir)relevant choices and standard techniques, and develop some basic tools to deal with CEPT and interactive algorithms in general.

We recommend reading Sections 4.1 and 4.2, and skipping the remaining sections until they are used in Section 5 or later.

## 4.1. Conventions in our setting

In the rest of this work, $\mathcal{T}$ always denotes a runtime class $\mathcal{T} \in \{\mathcal{PPT}, \mathcal{CPPT}, \mathcal{EPT}, \mathcal{CEPT}\}$.

### 4.1.1. Input generation and (non-)uniformity

For protocols, such as zero-knowledge, specification of inputs (in security definitions) is usually done via universal quantification over inputs. In the uniform complexity setting [Gol93], we specify an *efficiently samplable input distribution* instead. The machine sampling the input distribution is usually denoted $\mathcal{G}$ and called the **input-generating machine**. For non-uniform security, we use the same definition, but give $\mathcal{G}$ (and *only* $\mathcal{G}$) an interface for tape-like access to an (unbounded) non-uniform advice string $adv c_\kappa$. All other algorithms are uniform algorithms. This deviates from standard definitions [Gol01]

only by allowing input *distributions*. Non-uniformity is typically, but not always, unrelated to runtime problems.[22] See also Appendix A.5.

*Notation* 4.1. Let $(\mathcal{P}, \mathcal{V})$ be a two-party protocol and let $\mathcal{G}$ be an input generation machine (or distribution). We use the shorthand notation $\langle \mathcal{P}, \mathcal{V} \rangle_{\mathcal{G}}$ for the system resp. interaction of $\langle \mathcal{P}, \mathcal{V} \rangle$ completed with $\mathcal{G}$, e.g. $\langle \mathcal{P}(x), \mathcal{V}(y) \rangle$ with inputs distributed as $(x, y) \leftarrow \mathcal{G}$.

*Remark* 4.2 (Environmental distinguishing light). Having an input-generating machine $\mathcal{G}$ in real-ideal settings begs the question whether it can "cooperate" with a distinguisher $\mathcal{D}$ in some way, e.g. whether $\mathcal{G}$ and $\mathcal{D}$ should be considered as one machine (with shared state), a distinguishing "environment". For sequential zero-knowledge (Definition 6.1) we use a similar approach, see also Remark 6.2.

### 4.1.2. A posteriori time, a priori time, and designated adversaries

Our view on runtime of algorithms, or rather systems of machines, is mostly *a posteriori*. Let $\mathsf{A}$ be an algorithm and $\mathcal{E}$ be an environment such that $\langle \mathcal{E}, \mathsf{A} \rangle$ is a closed system. We say $\mathsf{A}$ is (**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, ...). Applying this to security notions leads to **designated adversaries**, which need only be *efficient for the protocol they are designed to attack*, see [Fei90] or [KL08; Gol10].

An algorithm $\mathsf{A}$ is **a priori PPT** if there is some $\mathsf{poly}$, so that $\mathsf{time}_{\mathsf{A}}(\langle \mathcal{E}, \mathsf{A} \rangle)$ is strictly bounded by $\mathsf{poly}$, for any system $\mathcal{E}$ so that $\langle \mathcal{E}, \mathsf{A} \rangle$ is closed. We define **a priori EPT** analogously. Note that, 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. Also note that "classical" PPT algorithms are not a priori PPT in our sense, since their runtime bound depends on the input size. We can mitigate this by size-guarding (see Definition 5.2).

For PPT (and CPPT), the distinction of *a priori PPT* and (a posteriori) PPT is often insignificant. For example, any CPPT algorithm $\mathsf{A}$ with virtual runtime bound $\mathsf{poly}$ can be truncated to $\mathsf{poly}$ steps, giving an statistically indistinguishable a priori PPT algorithm $\mathsf{A}'$. Thus, we can usually assume *a priori* PPT for PPT adversaries.[23]

### 4.1.3. Linearity of expectation (and subadditivity)

We often consider the runtime of a subsystem, and not the whole system. We illustrate some choices and their (ir)relevance in following example.

Suppose $(\mathcal{P}, \mathcal{V})$ is a protocol, for sake of concreteness a zero-knowledge proof system. As our example, we consider $\mathcal{D}(\mathsf{out}_{\mathcal{V}^*}\langle \mathcal{P}, \mathcal{V}^* \rangle_{\mathcal{G}})$, and think of $(\mathcal{G}, \mathcal{V}^*, \mathcal{D})$ as the adversary. The total system time is $T_{\text{total}} := \mathsf{time}_{\mathcal{G}+\mathcal{P}+\mathcal{V}^*+\mathcal{D}}(\dots)$. The time spent in the protocol is $T_{\text{proto}} := \mathsf{time}_{\mathcal{P}+\mathcal{V}^*}(\dots)$. The time spent pre- and post-protocol (i.e. in the distinguishing environment) is $T_{\text{env}} := \mathsf{time}_{\mathcal{G}+\mathcal{D}}(\dots)$

Generally, we consider adversaries $\mathcal{T}$-time if the *total* system time is $\mathcal{T}$-time. In particular, we include the runtime of honest parties.[24] Fortunately, we can usually apply the following rules of thumb: If $T_{\text{total}}$ is efficient, say CEPT, then by truncation argument, there is an *a priori PPT* $\mathcal{D}'$ with advantage at least half that of $\mathcal{D}'$. Thus, we may assume *a priori PPT* $\mathcal{D}$, making its runtime contribution "irrelevant". Moreover, by linearity of expectation, $\mathbb{E}(\mathsf{time}_{\mathcal{G}+\mathcal{P}+\mathcal{V}^*}(\dots)) = \mathbb{E}(\mathsf{time}_{\mathcal{G}}(\dots)) + \mathbb{E}(\mathsf{time}_{\mathcal{P}+\mathcal{V}^*}(\dots))$. Hence, we can consider efficiency of $\mathbb{E}(\mathsf{time}_{\mathcal{G}}(\dots))$ and $\mathbb{E}(\mathsf{time}_{\mathcal{P}+\mathcal{V}^*}(\dots))$ in isolation as well. (Virtualities interfere mildly, see Sections 4.2 and 4.5.) For indistinguishability purposes, we can typically (by

[22] For example, this fails if non-uniformity breaks hardness assumptions: 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 may very well 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).

[23] The argumentation relies on the possibilistic nature of PPT. It does not apply to other runtime classes, such as EPT

[24] This is a tradeoff, and has some consequences. In particular, we do not require honest parties to be "robust"; they may never halt upon receiving an ill-formed message. Of course, from actual protocols, we want strong(er) robustness guarantees.

truncation arguments) assume $\mathcal{G}$ to be *a priori PPT* as well. For efficiency questions, $\mathcal{G}$ can usually not be truncated (except in the presence of size-guards).

Instead of expectation, any subadditive "measure of efficiency" may be used, e.g. sup instead of $\mathbb{E}$. The total runtime is then *at most* the sum of its parts, i.e. we have an *inequality*.

## 4.2. Standard reductions and truncation techniques

In this section, we give some semi-abstract reduction and truncation techniques, which are a central work-horse for dealing with designated CEPT adversaries.

**Lemma 4.3** (Reduction to a priori runtime)**.** *Let $\mathcal{O}_0$ and $\mathcal{O}_1$ be two oracles. Suppose $\mathcal{D}$ is a distinguisher with advantage $\varepsilon := \mathrm{Adv}^{dist}_{\mathcal{D},\mathcal{O}_0,\mathcal{O}_1}$ and suppose $\mathsf{time}_{\mathcal{D}}(\mathcal{D}^{\mathcal{O}_0})$ is bounded by $(t_0, \nu_0)$ (with expectation $t_0$ and virtuality $\nu_0$). Then there is a (one-shot) distinguisher $\mathcal{A}$ with runtime strictly bounded by $t = 4t_0$ (up to emulation overhead), and advantage at least $\frac{\varepsilon}{4} - \nu_0$. More concretely, $\mathcal{A}$ is a runtime truncation of $\mathcal{D}$ after $t$ steps, hence the runtime distribution of $\mathcal{A}$ and $\mathcal{D}$ are closely related. Indeed, $\mathbb{E}(\mathsf{time}_{\mathcal{A}}(\mathcal{A}^{\mathcal{O}_0})) \lessapprox (1-\nu_0)t_0 + 4\frac{\nu_0}{\varepsilon}t_0$ (up to emulation overhead).*

*Proof sketch.* By assumption, there is a set of good events $\mathcal{G}$ so that $\mathbb{E}(\mathsf{time}_{\mathcal{D}}(\mathcal{D}^{\mathcal{O}_0}) \mid \mathcal{G}) \leq t_0$ and $\mathbb{P}(\neg\mathcal{G}) \leq \nu_0$. Let $\mathcal{A}$ be $\mathcal{D}$ truncated to $4\varepsilon^{-1}t_0$ steps. Let $\mathcal{A}$ return a random guess on `timeout`. The outputs of $\mathcal{D}^{\mathcal{O}_0}$ and $\mathcal{A}^{\mathcal{O}_0}$ have statistical distance at most $\frac{\varepsilon}{4} + \nu_0$.

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

In the setting of polynomial runtime, we get the following.

**Corollary 4.4** (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 $\varepsilon := \frac{1}{\mathsf{poly}}$ infinitely often, and $\mathsf{time}_{\mathcal{D}}(\mathcal{D}^{\mathcal{O}_0}) \in \mathcal{CEPT}$. Then there is an* a priori PPT *(one-shot) distinguisher $\mathcal{A}$ with advantage $\geq \frac{\varepsilon}{4} - \mathsf{negl}$ infinitely often. More concretely, $\mathcal{A}$ is a truncation of $\mathcal{D}$ with roughly the same runtime.*

Note that $\mathcal{D}$ need only be efficient for $\mathcal{O}_0$. Corollary 4.4 follows immediately from Lemma 4.3 using $\frac{1}{\varepsilon} \leq \mathsf{poly}$.

*Remark* 4.5 (Standard cutoff argument). The strategy in the proof of Lemma 4.3 and Corollary 4.4 is the *standard cutoff argument*. It works with minor variations in many situations.

*Notation* 4.6. We often sloppy write $\overset{c}{\approx}$ instead of $\overset{c}{\approx}_{\mathcal{T}}$ when specifying indistinguishability. Corollary 4.4 justifies this (for the runtime classes of interest).

*Caution* 4.7. A posteriori efficiency and hybrid arguments can behave unexpectedly. Even if the first and last hybrids are efficient, that does *not trivially* imply the intermediate hybrids are! In particular, the hybrid distinguisher need not be efficient. As a matter of fact, we run into such a problem for sequential composition of zero-knowledge proofs, and require "benign" simulators to work around it.

Sometimes, one needs to truncate oracle executions, and argue about the statistical distance of output distributions.

**Lemma 4.8** (Truncation of timed oracles)**.** *Let $\mathcal{G}$ be an input generating machine, $\mathsf{A}$ be an oracle-algorithm and $\mathcal{O}$ be a (probabilistic) timed oracle. Let $T = \mathsf{time}_{\mathcal{O}}(\mathsf{A}^{\mathcal{O}(y)}(x))$, where $(x, y) \leftarrow \mathcal{G}$. Suppose $\mathsf{bnd}$ is a bound for $T$, and let $\varepsilon$ be such that $\mathbb{P}(T_\kappa > \mathsf{bnd}(\kappa)) \leq \varepsilon(\kappa)$. Let $\mathcal{O}'$ be the truncation of $\mathcal{O}$ after $\mathsf{bnd}$ steps. Then*

$$\mathbb{P}(\mathcal{O}' \textit{ returns } \texttt{timeout} \textit{ in } \langle \mathsf{A}, \mathcal{O}' \rangle_{\mathcal{G}}) \leq \varepsilon.$$

*In particular, executions using $(\mathcal{G}, \mathcal{O})$ resp. $(\mathcal{G}, \mathcal{O}')$ with $\mathsf{A}$ have statistical distance $\varepsilon$.*

Applications of Lemma 4.8 need efficienctly computable cutoffs, in our case $\mathsf{bnd} = \mathsf{poly}$. Typically, $\varepsilon$ (and $\mathsf{bnd}$) is found via standard tail bounds (e.g. Corollary 3.8), and we additionally truncate $\mathcal{G}$ in order to get an priori PPT setting.

*Proof.* In the setting of Lemma 4.8, $\mathcal{O}'$ and $\mathcal{O}$ only differ if a `timeout` occurs. Thus, the claim follows, if we prove the bound $\varepsilon$ on the `timeout` probability. This follows immediately from the definition of $\mathsf{bnd}$ and $\varepsilon$. □

## 4.3. 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* 4.9 ($\mathsf{PPTpa}$). A timeful oracle $\mathcal{O}$ is **a priori PPT per activation ($\mathsf{PPTpa}$)**, if there is a polynomial $\mathsf{poly}$ such that if every invocation of $\mathcal{O}$ has runtime bounded by $\mathsf{poly}(\kappa)$.

Weaker relative notions of efficiency exists, and are sufficient for most purposes. The relevant property for is that, if a distinguisher yields an inefficient system, then the oracle is never to blame.

*Definition* 4.10 (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) is a **simple assumption**. We also say $C_0$ and $C_1$ form a simple assumption.

*Example* 4.11. 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 not simple as well. Note that assumptions which can be reduced to simple assumptions need not be simple.[25]

By definition, simple assumptions are essentially falsifiable assumptions 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.[26] Since our primary setting is the plain model, simple assumptions are a natural, but we stress that our techniques work for a much broader class of game-based assumptions.

*Definition* 4.12 (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 $1, \dots, q$.)

Note that $\mathsf{rep}(\mathcal{O}) = \mathsf{rep}_\infty(\mathcal{O})$.

*Definition* 4.13 (Indistinguishability under repeated trials). Let $\mathcal{O}_0$ and $\mathcal{O}_1$ be two oracles. Let $\mathcal{D}$ be a distinguisher. The **distinguishing advantage under $q$-repeated trials** is

$$\mathrm{Adv}^{q\text{-rt-dist}}_{\mathcal{D},\mathcal{O}_0,\mathcal{O}_1}(\kappa) = \mathrm{Adv}^{\mathrm{dist}}_{\mathcal{D},\mathsf{rep}_q(\mathcal{O}_0),\mathsf{rep}_q(\mathcal{O}_1)}(\kappa),$$

[25] For example, "soundness" of smooth projective hash functions [CS02] and derived non-interactive zero-knowledge argument systems [KW15] are no simple assumptions. (Without extractability, i.e. proof of knowledge, it is hard to check whether a "proof" was simulated, i.e. the experiment is inefficient.)

[26] Indeed, typical 1-more assumptions have a meaningful notion of security under repeated trials as well, but Definition 4.12 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.

where $\mathsf{rep}_q(\mathcal{O})$ is the $q$-fold repeated access oracle of Definition 4.12. In other words, the distinguisher has access to (at most) $q$ independent instances of $\mathcal{O}_b$ (for $b = 0, 1$).

Let $\mathcal{T} \in \{\mathcal{PPT}, \mathcal{CPPT}, \mathcal{EPT}, \mathcal{CEPT}\}$. We say $\mathcal{O}_0$ and $\mathcal{O}_1$ are *$\mathcal{T}$-time indistinguishable under $q$ (repeated) trials*, if $\mathrm{Adv}^{q\text{-rt-dist}}_{\mathcal{D},\mathcal{O}}(\kappa) = \mathsf{negl}$ for any $\mathcal{T}$-time distinguisher $\mathcal{D}$ which creates at most $q$ queries instances of $\mathcal{O}_b$. 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.

As usual, we define *statistical* indistinguishability by counting only oracle-queries as runtime.

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

**Lemma 4.14** (Hybrid lemma for simple assumptions)**.** *Let $\mathcal{C}_0$ and $\mathcal{C}_1$ be two oracles forming a simple assumption. Suppose $\mathcal{D}$ is a CEPT distinguisher with $q$ trials and advantage $|\mathrm{Adv}^{dist}_{\mathcal{D},\mathcal{C}}| \geq \varepsilon = 1/\mathsf{poly}$ infinitely often. Suppose $\mathsf{time}_{\mathcal{D}}(\mathcal{D}^{\mathcal{C}_0})$ is bounded by $(t_0, \nu_0)$ (with expectation $t_0$ and virtuality $\nu_0$ not necessarily polynomial). 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 one-shot distinguisher $\mathcal{A}$ with advantage at least $\frac{1}{M}(\frac{\varepsilon}{4} - \nu_0)$ infinitely often. More concretely, $\mathcal{A}$ is the hybrid distinguisher which is truncated according to Lemma 4.3.*

We remark that $\varepsilon = \frac{1}{\mathsf{poly}}$ and $t_0$ polynomial implies that $M$ is polynomial and $\frac{1}{M}(\frac{\varepsilon}{4} - \nu_0) \geq \frac{1}{\mathsf{poly}'}$ for some $\mathsf{poly}'$.

**Corollary 4.15.** *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 4.14.* First apply Corollary 4.4 to get an a priori PPT distinguisher $\mathcal{A}'$. Note that we treat distinguishing under repeated trials as one-shot 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 $(t_0, \nu_0)$ is virtually expected time of $\mathcal{D}$ as in Lemma 4.3. 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. That is, $\mathcal{A}$ simulates all but one of the $M$ instances, and embeds the challenge oracle $\mathcal{C}_b$ into the randomly chosen instance. The claim follows. □

The loss of advantage in Lemma 4.14 is very coarse, and can be refined if better bounds on the distribution of the number of queries made by the distinguisher are available, see Corollary B.8. Also, Definition 4.13 and Lemma 4.14 generalise to any algebra-tailed runtime class, and one can give a triple-oracle variation of Definition 4.13 and Lemma 4.14, but we have no use for this.

## 4.4. Relative efficiency

By considering a posteriori runtime and designated adversaries, we lack a notion of "absolute" efficiency of an algorithm (or timeful system). We make up for this by using a relative notion of efficiency, which becomes a definitional cornerstone in our setting.

*Definition* 4.16 (Relative efficiency). Let $\mathsf{A}$ and $\mathsf{B}$ be two (interactive) algorithms (or timeful systems) with identical interfaces. We say that $\mathsf{B}$ is **weakly $(\mathcal{T}, \mathcal{S})$-efficient relative to** $\mathsf{A}$ *w.r.t. (implicit) runtime classes $\mathcal{T}$, $\mathcal{S}$*, if for all (algorithmic) 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 $\mathsf{B}$ is **weakly efficient relative to** $\mathsf{A}$ *w.r.t. an (implicit) runtime class $\mathcal{T}$*, if it is weakly $(\mathcal{T}, \mathcal{T})$-efficient relative to $\mathsf{A}$.

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. It is irrelevant if the simulation is efficient, even when the real protocol is not.

There are two problems with Definition 4.16. First, it is not stringent enough for our proof techniques. We fix that later. Second, it does not 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 $\mathsf{Sim}$ 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. To reassure the sceptical reader: All sketchiness can be removed in our use cases by syntactically inserting the definitions instead. (The main imprecision is our sketchiness w.r.t. to systems, algorithms, and interfaces. Nevertheless, we believe that this improves the presentation.)

*Definition* 4.17. A **mapping** of system (or algorithms) to systems (or algorithm) 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 generalises Definition 4.16; it can be recovered from the constant mappings $\mathrm{F} = \mathsf{A}$, $\mathrm{G} = \mathsf{B}$.

*Definition* 4.18. 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 $\mathcal{E}$,

$$\forall \mathcal{A} \in C_I\colon \quad \mathsf{time}_{\mathcal{E}+\mathrm{F}(\mathcal{A})}(\langle \mathcal{E}, \mathrm{F}(\mathcal{A}) \rangle) \in \mathcal{T} \implies \mathsf{time}_{\mathcal{E}+\mathrm{G}(\mathcal{A})}(\langle \mathcal{E}, \mathrm{G}(\mathcal{A}) \rangle) \in \mathcal{S}$$

Unfortunately, Definition 4.18 is not strong enough to be used in reductions, which is why we reserved the specification "*weakly*" for Definitions 4.16 and 4.18 (More precisely, we cannot prove or refute that it is (not) strong enough.) Therefore, we rely on following strengthening:

*Definition* 4.19. Let F, G, etc. be as in Definition 4.18. 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 $(t_0, \varepsilon_0)$-time, then $\mathsf{time}_{\mathrm{F}(\mathcal{A})}(\langle \mathcal{E}, \mathrm{G}(\mathcal{A}) \rangle)$ is virtually strict/expected $(t_1, \varepsilon_1)$-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 $\kappa$).

We stress that Definition 4.19 is unconditional w.r.t. the environment, i.e. uses timeful environments, and that the tightness bounds depend *only on* $\kappa$. Mixing strict and expected time (i.e. $\|\cdot\|_\infty$ and $\|\cdot\|_1$) in Definition 4.19 is possible and useful. For example when strict PPT protocols and adversaries are handled by EPT simulators.

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 $\mathfrak{C}$ of admissible environments with restriction beyond runtime.

## 4.5. From CEPT to EPT

The characterisation 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* 4.20. Let $\langle \mathcal{P}, \mathcal{V} \rangle$ be an interactive protocol. Suppose $\mathcal{P}$ sends a random message $r$. Suppose $\mathcal{V}$ picks a random number $s$, and if $r = s$, it loops forever. Otherwise the protocol finishes. Now, the bad events are $(r, r)$ (or some (ugly) 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 4.21** (Timeout oracles)**.** *Let* $\mathsf{A}$ *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* $(t, \varepsilon)$*. 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* $\mathsf{A}$ *aborts upon receiving* `timeout`*, then*[27] $\mathsf{time}_{\mathcal{O}}(\langle \mathsf{A}, \mathcal{O}' \rangle)$ *is EPT with expected runtime* $t + O(1)$ *(with small hidden constant).*[28] *The probability for a* `timeout` *message in* $\langle \mathsf{A}, \mathcal{O}' \rangle$ *is* $\varepsilon$*.*[29]

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 4.21 is roughly as follows: Replace $\mathcal{O}$ with the "imaginary" $\mathcal{O}'$. 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 "imaginary" 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 $\nu$-quantile cutoffs are achieved by extension of $\Omega_{\mathcal{O}}$, as usual.

*Proof of Lemma 4.21.* 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 $\nu$-quantile cutoff is constructed as usual, i.e. let $N$ be the minimal such that

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

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 $\nu$-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 3.8). □

In our setting, we usually deal with "multi-oracle" adversaries. For example, zero-knowledge needs input generation $\mathcal{G}$ and a malicious verifier $\mathcal{V}^*$ (and a distinguisher $\mathcal{D}$ which of lesser concern). Clearly, we can view $\mathcal{G}$ 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 $\mathcal{G}$ to produce inputs, and then continues as $\mathcal{V}^*$.

**Lemma 4.22** (Sequential timeout oracles)**.** *Let* $\mathsf{A}$ *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 4.21 hold for* $\mathcal{O}$*, where* $\Omega_{\mathcal{O}} = \Omega_{\mathcal{O}_1} \times \Omega_{\mathcal{O}_2}$*.*

[27] More formally, one should lift $\mathsf{A}$ to an algorithm which aborts upon receiving `timeout`, since `timeout` is a special symbol which $\mathsf{A}$ cannot receive. Using $\mathsf{A}$ unchanged, forces it to interpret (an encoding of) `timeout`. It may then continue to call $\mathcal{O}'$, even though $\mathcal{O}'$ does not respond anymore. In some machine models, an invocation of $\mathcal{O}$ increases $\mathcal{O}$'s runtime. Due to such models, we require $\mathsf{A}$ to explicitly abort.

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

[29] The probability space may be enlarged to achieve an exact cutoff, see Section 2.4.

*Moreover the probability $\varepsilon$ 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 4.22 follows essentially from Lemma 4.21 and the fact that the runtimes of $\mathcal{O}_1$ and $\mathcal{O}_2$ sum up. For the decomposition, one argues as in the proof of Lemma 4.21, 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 exactly by incorporating the steps of $\mathcal{O}_1'$. The proof is left to the reader.

In particular, we obtain following special case.

**Corollary 4.23.** *Let* $\mathsf{A}$ *be an interactive algorithm and let $\mathcal{G}$ be an input generating machine. Suppose* $\mathsf{time}_{\mathcal{O}+\mathcal{G}}(\langle \mathsf{A}, \mathcal{O}\rangle_{\mathcal{G}})$ *is CEPT with virtual runtime $(t,\varepsilon)$. Then there exists an environment $(\mathcal{G}', \mathcal{O}')$, modelled as* timeful *oracles (with stochastically dependent randomness and not necessarily efficiently computable), such that: $(\mathcal{G},\mathcal{O})$ and $(\mathcal{G}',\mathcal{O}')$ behave identically except that $(\mathcal{G}',\mathcal{O}')$ sends* `timeout` *to signal bad executions. If* $\mathsf{A}$ *aborts upon receiving* `timeout`*, then* $\mathsf{time}_{\mathcal{O}+\mathcal{G}}(\langle \mathsf{A}, \mathcal{O}'\rangle_{\mathcal{G}'})$ *is EPT with expected runtime $t + O(1)$ (with small hidden constant). Moreover, the probability for a* `timeout` *message in $\langle \mathsf{A}, \mathcal{O}'\rangle_{\mathcal{G}'}$ is $\varepsilon$. (The probability space may be extended.)*

*An analogous result holds if a distinguisher $\mathcal{D}$ is applied, i.e. for* $\mathsf{time}_{\mathcal{G}+\mathcal{O}+\mathcal{D}}(\mathcal{D}(\langle \mathsf{A}, \mathcal{O}\rangle_{\mathcal{G}}))$.

The precise form of Lemma 4.22 shows that, if we manipulate only interaction with $\mathcal{O}_2$ (which is $\mathcal{O}$ in Corollary 4.23), then only that portion of virtuality is relevant. For simplicity, do not use this precision in later results; indeed, this does not seem to help much.

*Remark* 4.24. In the setting of Lemma 4.22, 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 4.22.) Concretely, assuming total virtuality $\varepsilon$, one can apply Lemma 4.21 for an $\varepsilon$-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 $\varepsilon$.) Together, $(\mathcal{O}_1',\mathcal{O}_2')$ have overall timeout probability of (at most) $2\varepsilon$ and the expected runtime is $t + O(1)$. Unfortunately, the timeout probability of this construction is larger than $\varepsilon$.

Lastly, we note that 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 (in Lemma 4.22).

# 5. Application to zero-knowledge proofs

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 system for NP-languages.

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

**Efficiency:** There is a polynomial poly so that for all $(\kappa,x,w)$ 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 (computational) soundness, since we do not need it.

## 5.1. Zero-knowledge

Our main interest is the zero-knowledge property of proof systems. Completeness and soundness hold unconditionally anyway.

*Definition* 5.2. Let $\mathcal{T}, \mathcal{S} \in \{\mathcal{PPT}, \mathcal{CPPT}, \mathcal{EPT}, \mathcal{CEPT}\}$. Let $(\mathcal{P}, \mathcal{V})$ be a proof system (with efficient prover). A **universal simulator** $\mathsf{Sim}$ takes as input $(\mathsf{code}(\mathcal{V}^*), x, \mathit{aux})$ and simulates $\mathcal{V}^*$'s output. Let $(\mathcal{G}, \mathcal{V}^*, \mathcal{D})$ be an adversary.

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

where $(x, w, \mathit{aux}) \leftarrow \mathcal{G}$ and $(x,w) \in \mathcal{R}$ or $\mathsf{Real}$ resp. $\mathsf{Ideal}$ return a failure symbol.

The distinguishing advantage of $(\mathcal{G}, \mathcal{V}^*, \mathcal{D})$

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

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

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

- $\mathsf{time}_{\mathsf{Sim}}(\mathsf{Ideal}^{\mathsf{Sim}}_{\mathcal{G},\mathcal{V}^*}) \in \mathcal{S}$, i.e. $\mathsf{Sim}$ is weakly $(\mathcal{T}, \mathcal{S})$-efficient relative to $\mathcal{P}$, see Section 4.4. (Note that the runtime of $\mathsf{Sim}$ includes whatever time is spent to emulate $\mathcal{V}^*$, i.e. $\mathcal{V}^*$ is not viewed as a (bb-rw) oracle.)
- $\mathrm{Adv}^{\mathsf{zk}}_{\mathcal{G},\mathcal{V}^*,\mathcal{D}}(\kappa)$ is negligible

We define **(uniform) zero-knowledge w.r.t. (input) size-guarded** security as follows: For any (monotone) polynomial bound $\mathsf{gd}$ (called **size-guard**), the derived protocol, where prover and verifier abort with $\mathtt{gderr}$ 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, see Section 4.1.1.

In light of Definition 4.17, one should interpret $\mathsf{Sim}$ as a mapping of malicious verifier $\mathcal{V}^*$ to simulator $\mathsf{Sim}(\mathcal{V}^*)$, so that $\mathsf{Sim}(\mathcal{V}^*)$ and $\langle \mathcal{P}, \mathcal{V}^*\rangle$ have the same interface (i.e. expect $x, w$, *aux* and give outputs), and $\mathsf{Sim}$ (by definition) ignores $w$. Some more remarks are in order.

*Remark* 5.3 (Size-guards). The use of (input) size-guarded security is meant to address certain situation, which we may want to consider secure, but cannot due to runtime artifacts. 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 artifact, 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.

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 F.4 for more on size-guards.

*Remark* 5.4 (Efficiency of the simulation). Keep in mind that Definition 5.2 only ensures that $\mathsf{Sim}$ is *weakly* efficient relative to the prover. This is a source of trouble. Yet, (strong) relative efficiency is an unconditional property, and hence usually not possible (except if simulation is perfect).

One may expect that, by a standard reduction to PPT, w.l.o.g. $\mathcal{G}$ and $\mathcal{D}$ are a priori PPT. This is true when verifying the output *quality* of $\mathsf{Sim}$. However, it is false when verifying the *efficiency* of $\mathsf{Sim}$ (recall the size-guarding problem).

*Remark* 5.5. 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.

*Remark* 5.6. By a standard reduction to PPT (Corollary 4.4), we can assume that $\mathcal{D}$ is a priori PPT in Definition 5.2. Thus, the usual formulation of indistinguishable ensembles,

$$\begin{aligned}&\{(x, \mathit{aux}, \mathit{out}) \mid (x, w, \mathit{aux}) \leftarrow \mathcal{G}(\kappa); \mathit{out} \leftarrow \mathsf{out}_{\mathcal{V}^*}\langle \mathcal{P}(w), \mathcal{V}^*(\mathit{aux})\rangle(x)\}_\kappa\\ \overset{c}{\approx}\ &\{(x, \mathit{aux}, \mathit{out}) \mid (x, w, \mathit{aux}) \leftarrow \mathcal{G}(\kappa); \mathit{out} \leftarrow \mathsf{Sim}(\mathsf{code}(\mathcal{V}^*), x, \mathit{aux})\}_\kappa\end{aligned}$$

is an equivalent definition of (size-guarded) zero-knowledge (Definition 5.2).

*Remark* 5.7 (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 prevents *strict polynomial space simulation*, see Remark A.5.[30] We stress that we work in a setting where expected polynomial space and randomness are allowed, but we find it questionable to stipulate this generally.

*Remark* 5.8 (Inefficient provers and simulation tightness). 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.[31]

By viewing $\mathcal{P}$ as timeful and setting its runtime to its message length, Definition 5.2 extends to inefficient provers. Our results can be suitably adapted, however size-guarding is may be necessary, since the simulator's runtime is too limited.[32] Indeed, even efficient provers "become inefficient" in this setting, e.g. if $\mathcal{V}^*$ is linear in $|x|$, but $\mathcal{P}$ is quadratic, then problematic input distributions exist. Thus, in general, we cannot hope for efficient simulation without size-guarding or similar restrictions.

*Remark* 5.9. 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), $\mathcal{G}$ 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:** $\exists\mathsf{Sim}\forall\mathcal{V}^*\forall\mathcal{G}\forall\mathcal{D}$. Typical form of non-black-box simulation, e.g. in [Bar01].
**Existential simulator:** $\forall\mathcal{V}^*\exists\mathsf{Sim}\forall\mathcal{G}\forall\mathcal{D}$. Typical definition of zero-knowledge, e.g. in [Gol01].

We see in Section 5.2 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 "$\forall\mathcal{V}^*\forall\mathcal{G}\exists\mathsf{Sim}\forall\mathcal{D}$"), *weak simulation* (roughly "$\forall\mathcal{V}^*\forall\mathcal{D}\exists\mathsf{Sim}\forall\mathcal{G}$"), *weak distributional simulation* (roughly "$\forall\mathcal{V}^*\forall\mathcal{D}\forall\mathcal{G}\exists\mathsf{Sim}$"), see [Dwo+03; CLP15]. We have not pursued an adaption to CEPT for these notions.

## 5.2. 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 of $\mathcal{V}_{\text{univ}}$ is of the form $(\mathit{code}, \mathit{state}, \mathit{aux})$, and $\mathcal{V}_{\text{univ}}$ continues execution of the code *code* in state *state*. The universal adversary is well-behaved (and may try to "cooperate" with Sim). Despite "attempted cooperation" of $\mathcal{V}_{\text{univ}}$ and Sim, $\mathcal{V}_{\text{univ}}$ 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" depedency on $\mathcal{V}^*$ does not weaken the notion of zero-knowledge. Thus, in Definition 5.2, we do not give up any power.

**Lemma 5.10** (Equivalence of existential and universal simulation)**.** *Let $\mathcal{T}$ be a runtime class and let $(\mathcal{P}, \mathcal{V})$ be an interactive proof system. If this proof system is zero-knowledge w.r.t. to existential simulation, then it is zero-knowledge w.r.t. the universal simulator* Sim *defined as follows.*

[30] Our notion of bb-rw access is problematic in that aspect as well, but that may be salavageable, see Remark A.5.
[31] This entails some technical artifacts, e.g. a prover may be badly behaved for invalid inputs, e.g. not halting when given invalid. The complexity class for "good protocols" should be robust and prevent such behaviour.
[32] This may be "fixed" in a stupid way by preventing too efficient verifiers, e.g. using timelock puzzles.

*Let $\mathcal{V}_{\text{univ}}$ be the universal adversary and $\mathsf{Sim}_{\text{univ}}$ be the existential simulator for $\mathcal{V}_{\text{univ}}$. The $\mathsf{Sim}$ 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 $\mathcal{V}^*$. This also holds for (non-)uniform (size-guarded) zero-knowledge.*

*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

$$\mathsf{Ideal}^{\mathsf{Sim}}_{\mathcal{G},\mathcal{V}^*} \overset{\mathcal{D}}{\equiv} \mathsf{Ideal}^{\mathsf{Sim}_{\text{univ}}}_{\mathcal{G},\mathcal{V}_{\text{univ}}} \overset{c}{\approx} \mathsf{Real}_{\mathcal{G}_{\mathcal{V}^*},\mathcal{V}_{\text{univ}}} \overset{\mathcal{D}}{\equiv} \mathsf{Real}_{\mathcal{G},\mathcal{V}^*}.$$

Size-guarding makes no difference in the proof. Neither does non-uniform advice (and by assumption, only $\mathcal{G}$ is non-uniform). □

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 $\mathsf{Sim}$ cannot depend upon.

*Caution* 5.11. We crucially relied on our *a posteriori* runtime notion. For other notions of runtime, Lemma 5.10 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 $\mathsf{poly}$ steps, for some *fixed* $\mathsf{poly}$, whereas $\mathcal{V}^*$ may run (much) longer, say $\mathsf{poly} + 1$ steps. (There is a family $\mathcal{V}^n_{\text{univ}}$ with runtime bounds $\mathsf{poly}_n(\kappa) = n\kappa^n$, so a morally equivalent result does hold.)

## 5.3. 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, providing motivation for the definitions to come. We only prove zero-knowledge, as completeness and soundness are unconditional. Formal definitions of commitment schemes are in Appendix D.1. We assume *left-or-right oracles* in the hiding experiment for commitment schemes. Intuitively, we assume a built-in hybrid argument. (Commitments are secure against CEPT adversaries if they are against PPT adversaries, by a simple truncation argument.)

### 5.3.1. The protocol

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

- **(P0)** The prover sends $\text{ck}_{\text{hide}} \leftarrow \mathsf{Com}^{(\text{H})}.\mathsf{Setup}(\kappa)$. ($\text{ck}_{\text{bind}} \leftarrow \mathsf{Gen}^{(\text{B})}(\kappa)$ is deterministic.)
- **(V0)** $\mathcal{V}$ randomly picks challenge edges $e_i \leftarrow E$ for $i = 1, \ldots, N = \kappa \cdot \mathsf{card}(E)$, commits to them as $c^e_i = \mathsf{Com}^{(\text{H})}(\text{ck}_{\text{hide}}, e_i)$, and sends all $c^e_i$.
- **(P1)** $\mathcal{P}$ picks randomised colourings $\psi_i$ for all $i = 1, \ldots, N$ and commits to all node colours for all graphs in (sets of) commitments $\{\{c^{\psi}_{i,j}\}_{j\in V}\}_{i=1,\ldots,N}$ using $\mathsf{Com}^{(\text{B})}$. $\mathcal{P}$ 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 any opening is invalid ($e_i \notin E$). Otherwise, for all iterations $i = 1, \ldots, n$, $\mathcal{P}$ 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 opening is invalid, any edge not correctly coloured, or if $\text{ck}_{\text{hide}}$ is bad.

Testing $\text{ck}_{\text{hide}}$ only at the end of the weakens the requirements of $\mathsf{VfyCK}$, namely it can use the setup randomness as additional input. We follow [GK96] in this matter, but the reader may assume a check in step (V0) without loss.

### 5.3.2. Proof of zero-knowledge

Our goal is to show the following lemma.

**Lemma 5.12.** *Suppose* $\mathsf{Com}^{(\mathrm{H})}$ *and* $\mathsf{Com}^{(\mathrm{B})}$ *are a priori PPT algorithms. Then protocol* $\mathrm{G3C}_{GK}$ *in Section 5.3.1 is unguarded zero-knowledge with a bb-rw CEPT simulator against CEPT adversaries. Let* $(\mathcal{G}, \mathcal{V}^*)$ *be an adversary and suppose* $T := \mathsf{time}_{\mathcal{P}+\mathcal{V}^*}(\mathsf{Real}_{\mathcal{G},\mathcal{V}^*})$ *is* $(t, \varepsilon)$*-time. Then* $\mathsf{Sim}$ *handles* $(\mathcal{G}, \mathcal{V}^*)$ *in virtually expected runtime* $(t', 2\varepsilon + \varepsilon')$*. Here* $\varepsilon'$ *stems from an advantage against the hiding property of* $\mathsf{Com}^{(\mathrm{B})}$*, hence* $\varepsilon'$ *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 normalisation procedure in [GK96]. This is because a negligible failure is absorbed into the CEPT virtuality, namely $\varepsilon'$. 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 5.13.** *The claims in Lemma 5.12 hold if* $T \in \mathcal{EPT}$*, i.e.* $\varepsilon = 0$*.*

*Proof sketch.* We proceed in game hops. The initial game has perfect outputs, while the final game describes our simulator. We consider bb-rw simulation.

**Game** $\mathsf{G}_0$ is the real protocol. The output is the verifier's output.

**Game** $\mathsf{G}_1$**:** If the verifier opens the commitments in (V1) correctly, the game repeatedly rewinds it to (P1) and using fresh prover randomness, until it obtains a second run where $\mathcal{V}^*$ unveils the commitments correctly (in (V1)). The output is the $\mathcal{V}^*$'s output in this second successful run. If the verifier failed in the first run, the protocol proceeds as usual. The outputs of $\mathsf{G}_1$ and $\mathsf{G}_0$ are identically distributed. It can be shown that this modification preserves (perfect) EPT of the overall game, i.e. $\mathsf{G}_1$ 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.

**Game** $\mathsf{G}_2$**:** We assume that both (valid) openings of $\mathcal{V}^*$'s commitments in (P1) open to the same value. Otherwise, we output `ambig`, indicating equivocation of the commitment. The probability for `ambig` is negligible, since one can (trivially) reduce to an adversary against the binding property of $\mathsf{Com}^{(\mathrm{B})}$. (This modification preserves perfect EPT.)

In **Game** $\mathsf{G}_3$, the initial commitments (in (P1)) to 3-colourings are replaced with commitments to random colours. These commitments are never opened. Thus, we can reduce distinguishing Games 2 and 3 to breaking the hiding property of $\mathsf{Com}^{(\mathrm{H})}$ modelled as left-or-right indistinguishability. More precisely, the reduction constructs real and random colouring, and uses a *commitment oracle* $\mathcal{O}_b$ which receives two messages and commits to one of them. Suppose $\mathcal{O}_0$ commits to the real colouring *(left)*, whereas $\mathcal{O}_1$ commits to the random colouring *(right)*. The modification of $\mathsf{G}_2$ to "oracle committing" yields an EPT Game $\mathsf{G}_{2'}$ (instantiated with $\mathcal{O}_0$). The modification of $\mathsf{G}_3$ to $\mathsf{G}_{3'}$ (with $\mathcal{O}_1$) is *virtually* EPT. This follows immediately from the standard reduction to PPT, because Games $\mathsf{G}_{2'}$ and $\mathsf{G}_{3'}$ differ only in their oracle, and the case of $\mathcal{O}_0$ is EPT.

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 $\mathsf{G}_3$ and $\mathsf{G}_{3'}$ only differ by (not) using oracle calls. Incorporating these oracles does not affect CEPT (as $\mathcal{O}_1$ is an priori PPT oracle). Thus, the reduction is in fact efficient. Assuming the time to compute a commitment depends only on the message length, a precise analysis shows, that the virtually expected runtime is affected negligibly.

In **Game** $\mathsf{G}_4$, the commitments in the second round (Step 3) are replaced by commitments to a pseudo-colouring (such that the commitments revealed to $\mathcal{V}^*$ have different (random) colours). The argument for efficiency and indistinguishability of outputs is analogous to the one before. It relies on the built-in hybrid of the LR-hiding setting.

The simulator is defined as in $\mathsf{G}_4$: It makes a first test-run with a random colouring. If the verifier does not open its challenge, it 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 $2^\kappa$ unsuccessful rewinds. But this is not necessary for our results.) □

We point out some important parts: First, in Game $\mathsf{G}_1$, 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, 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 clear, whether the introduction of rewinding in $\mathsf{G}_1$ 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 at factor of 2. Thus, "bad" queries are only twice as likely as before. (We leave the verification to the reader.)

More concretely, using Lemma 4.21, we obtain a $\mathcal{G}'$ 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 $\mathsf{G}_1$ is at most twice that of $\mathsf{G}_0$, (and the virtually expected runtime is roughly doubled as well). Hence, $\mathsf{G}_1$ is CEPT. We formalise this "rewind-then-simulate" approach in Sections 5.4 and 5.5.

*Remark* 5.14. The simulator in [GK96] is also a CEPT simulator. For a proof, proceed as in Lemma 5.13. The advantage of simulator in [GK96] is, that it 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.

## 5.4. Rewinding strategies

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

*Terminology* 5.15. For the sake of simplicity, we treat a bb-rw oracle as though it were a NextMsg oracle.[33] Let $\mathcal{O}$ be an oracle and define $\mathsf{bbrw}(\mathcal{O})$ to be the oracle which gives bb-rw access to $\mathcal{O}$. A query to $\mathcal{O}$ is some $m_i$, and depends on the state of $\mathcal{O}$. Queries to bb-rw oracles need to specify the "history" from where to continue, i.e. a query to $\mathsf{bbrw}(\mathcal{O})$ is are a sequence of messages $(m_1, \dots, m_\ell)$; one may view $\mathsf{bbrw}(\mathcal{O})$ as essentially deterministic and stateless.

By abuse of notation, we typically write $\mathsf{A}^{\mathcal{O}}$ instead of $\mathsf{A}^{\mathsf{bbrw}(\mathcal{O})}$ if it is understood that $\mathsf{A}$ has bb-rw access to $\mathcal{O}$. To emphasise this difference, we sometimes speak of a **fully qualified query (fq-query)** *query* to a bb-rw oracle. Even with this, should always be clear from the type of oracle access $\mathsf{A}$ has.

### 5.4.1. Definitions and basic results

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

*Definition* 5.16. 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 an fq-query which RWS queried before (or abort).[34]

A rewinding strategy RWS has **runtime tightness** poly, if the following holds: Let be $(\mathcal{G}, \mathcal{V}^*)$ any adversary (modelled as a timeful oracle). Let $T := \mathsf{time}_{\mathcal{P}+\mathcal{V}^*}(\langle \mathcal{P}, \mathcal{V}^* \rangle_{\mathcal{G}})$, and let $S := \mathsf{time}_{\mathsf{RWS}+\mathcal{V}^*}(\mathsf{RWS}^{\mathcal{V}^*(x,aux)}(x,w))$ with input distribution $\mathcal{G}$. Then $\mathbb{E}(S) \leq \mathsf{poly} \cdot \mathbb{E}(T)$ for all $(\mathcal{G}, \mathcal{V}^*)$.[35]

[33] This technically violates compatibility requirements for EPT, but there are straightforward fixes, see Appendix A.3.

[34] More correctly, RWS "outputs" the oracle $\mathsf{bbrw}(\mathcal{V}^*)$ in its "final" state, and perhaps an abort message to indicate failure.

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

Equivalently, for *deterministic* timeful $\mathcal{G}$, 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. 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* 5.17 (Preservation of EPT). It follows trivially that a rewinding strategy RWS with *polynomial* runtime tightness **preserves EPT**, i.e. in the setting of Definition 5.16, if $\mathsf{time}_{\mathcal{P}+\mathcal{V}^*}(\langle \mathcal{P}, \mathcal{V}^* \rangle_{\mathcal{G}}) \in \mathcal{EPT}$, then $\mathsf{time}_{\mathsf{RWS}+\mathcal{V}^*}(\mathsf{RWS}^{\mathcal{V}^*(x,\mathit{aux})}) \in \mathcal{EPT}$.

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

*Definition* 5.18 (Properties of rewinding strategies.). Let $(\mathcal{P}, \mathcal{V})$ be a proof system and RWS a rewinding strategy. Let $\mathcal{FQ}$ be the set of all possible fq-queries (i.e. tuples of messages, recall Terminology 5.15). Suppose $\mathcal{V}^*$ is some (malicious) deterministic verifier (as a timeful oracle). Let $\kappa$, $(x, w)$, *aux* be inputs. Let *query* $\in \mathcal{FQ}$ be a (possible) fq-query to $\mathcal{V}^*$. Let $\mathsf{pr}_{\text{real}}(\mathit{query})$ 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[36]

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

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

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

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

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

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

RWS has **output skew** $\delta = \delta(\kappa)$, if for every (deterministic) $(\mathcal{G}, \mathcal{V}^*), (x, w, \mathit{aux}) \leftarrow \mathcal{G}(\kappa)$ is similarly defined by the sup-ratio being at most $1 + \delta(\kappa)$, now of the output $\mathsf{RWS}^{\mathcal{V}^*(\mathit{aux})}(x, w)$ over the real fq-query $\mathsf{qseq}_{\mathcal{P}}(\langle \mathcal{P}(x, w), \mathcal{V}^*(x, \mathit{aux}) \rangle)$.[37] We say RWS has **perfect output (distribution)** if the output skew is 0.

We note that the properties in Definition 5.18 are unconditional. Finally, we define our notion of normality. The definition is closely related to Goldreich's definition of normality in [Gol10].[38]

*Definition* 5.19 (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, e.g. as a stepping stone for zero-knowledge. Negligible output skew would suffice, but natural rewinding strategies seem to satisfy perfect output skew, so we require that for simplicity.

### 5.4.2. Basic results

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

**Lemma 5.20** (Normal rewinding strategies preserve CEPT)**.** *Let* RWS *be a normal rewinding strategy for* $(\mathcal{P}, \mathcal{V})$*. Let* $(\mathcal{G}, \mathcal{V}^*)$ *be a CEPT adversary for zero-knowledge, that is* $\mathsf{time}_{\mathcal{G}+\mathcal{P}+\mathcal{V}^*}(\langle \mathcal{P}, \mathcal{V}^* \rangle_{\mathcal{G}}) \in \mathcal{CEPT}$*. Then* $\mathsf{time}_{\mathcal{G}+\mathsf{RWS}+\mathcal{V}^*}(\mathsf{RWS}^{\mathcal{V}^*(x,\mathit{aux})}(x, w)) \in \mathcal{CEPT}$*, where* $(x, w, \mathit{aux}) \leftarrow \mathcal{G}(\kappa)$.

[36] By abuse of notation, we write $\mathsf{qseq}_{\mathcal{P}}(\langle \mathcal{P}(x, w), \mathcal{V}^*(x, \mathit{aux}) \rangle)$, but mean fully qualified queries, i.e. the set of prefixes of the query sequence.

[37] More correctly, the distribution of the state of the timed bb-rw $\mathcal{V}^*$ must satisfy this.

[38] Goldreich notes in [Gol10, Footnote 24] that his notion of normality of a simulator is probably satisfied if the running time analysis is *unconditional*. Rewinding strategies make this explicit. Indeed, since our notion of runtime and efficiency of simulators is *not unconditional*, we deem this separation necessary.

*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* $\mathsf{time}_{\mathscr{P}+\mathcal{V}^*}(\langle \mathscr{P}, \mathcal{V}^* \rangle_{\mathcal{G}})$ *is virtually* $(t, \varepsilon)$*-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* $\langle \mathscr{P}, \cdot \rangle$ *with runtime tightness* $(\mathsf{poly}_{\mathsf{time}}, \mathsf{poly}_{\mathsf{virt}})$.

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

*Proof.* By Lemma 4.21, we know that there is a modification $(\mathcal{G}', \mathcal{V}')$ of $(\mathcal{G}, \mathcal{V}^*)$, which runs in EPT (as timeful oracles). We call a fq-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 $\mathscr{P}$ is $\varepsilon$ (by construction). By assumption, RWS runs in EPT for $(\mathcal{G}', \mathcal{V}')$. (For this, note that RWS treats $\mathcal{V}^*$ as a (timed) black-box.)

The only case where RWS encounters a difference between $(\mathcal{G}, \mathcal{V}^*)$ and $(\mathcal{G}', \mathcal{V}')$ is if RWS asks a timeout query, i.e. if ($\mathcal{G}$ or) $\mathcal{V}'$ return `timeout`. By normality of RWS, the probability of asking a timeout query is only polynomially higher than the probability that $\mathscr{P}$ asks a timeout query. The latter is $\varepsilon$ (essentially by definition), 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 will see in Remark 5.29, that runtime tightness already implies probability tightness. However, the actual bound is far from optimal. Following lemma is a simple way to get a tight(er) bound on probability tightness.

**Lemma 5.21.** *Let* RWS *be a rewinding strategy for* $(\mathscr{P}, \mathcal{V})$ *and* $(\mathcal{G}, \mathcal{V}^*)$ *deterministic timeful adversaries. Write* $(x, w, aux) \leftarrow \mathcal{G}(\kappa)$ *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* $\mathsf{pr}_{\mathrm{rws}}(query)$ *resp.* $\mathsf{pr}_{\mathrm{real}}(query)$ *be the probability that* RWS *resp.* $\mathscr{P}$ *queries query, as in Definition 5.18. Write* $\mathcal{Q}_i[j]$ *for the* $j$*-th query in* $\mathcal{Q}_i$*. Suppose that for all* $i$ *and all fq-queries query of length* $i$

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

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

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

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

The basic idea behind Lemma 5.21 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 5.21 gives a general formalisation of this idea (without needing to condition on some $m'$).

It is often (almost) trivial to verify the conditions of Lemma 5.21. Moreover, we are not aware of (natural) rewinding strategies which do not satisfy normality, even outside the context of zero-knowledge.[39]

[39] If the "query space" (in some round) is small, e.g. if there are only two different queries, then one may deterministically query both without affecting runtime too much. In particular, it does not break probability tightness. Due to the deterministic choice, Lemma 5.21 is not applicable. However, randomising the order of the choices (assuming that is possible) yields a compatible RWS′.

*Proof of Lemma 5.21.* The proof is almost trivial. Consider the setting and notation of Lemma 5.21. Let *query* be a fq-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

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

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

The criterion in Lemma 5.21 is "global" and not "local", making it somewhat inconvenient. Instead of applying Lemma 5.21, it is often simple(r) to derive more precise bounds and directly prove normality.

*Remark* 5.22 (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* 5.23. 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 execution 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 5.21 deals with partial transcripts, expected polynomially many executions, and probability tightness (not runtime), but is otherwise similar to [HM98].

### 5.4.3. Examples of normal rewinding strategies

We give some examples for rewinding strategies which are normal. Most claims follows easily from Lemma 5.21.

*Example* 5.24 (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* 5.25 (Constant round zero-knowledge). Our motivating example [GK96] and the simplification of Rosen [Ros04] have a normal rewinding strategies.

*Example* 5.26 (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* 5.27 (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).

### 5.4.4. Connection between runtime and probability tightness

Following example illustrates, that normality is not automatic.

*Example* 5.28 (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 always as $0^{\kappa}$ as its first query, against 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 $2^{\kappa}$. 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* 5.29 (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) fq-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 5.29 may also be viewed as an indication that almost(?) all used rewinding strategies are normal. More generally, even an apriori PPT adversary can exploit a deviation 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 completely unclear how that could be usefully exploited.

*Remark* 5.30 (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 $2^\kappa$ 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.

## 5.5. Benign simulators

Our definition of a benign simulator abstracts the proof strategy for $\mathrm{G3C}_{\mathrm{GK}}$. Before we give the definition, we demonstrate the idea.

*Example* 5.31 (Structure of the security reduction for $\mathrm{G3C}_{\mathrm{GK}}$). Consider the protocol $\mathrm{G3C}_{\mathrm{GK}}$ in Section 5.3.1 and the security proof in Section 5.3.2. Let $(\mathcal{G}, \mathcal{V}^*, \mathcal{D})$ be an adversary. Since the simulator cannot depend on $\mathcal{G}$ 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". We suppress the inputs $x, w$, *aux* for prover and simulator.

Let $\mathsf{A}_0$ denote the algorithm $\mathsf{out}_{\mathcal{V}^*}\langle \mathcal{P}, \mathcal{V}^* \rangle$. Let $\widetilde{\mathsf{A}}_0$ denote the algorithm which introduces all rewindings, as in Section 5.3.2, $\mathsf{G}_1$. Moreover, $\widetilde{\mathsf{A}}_0$ makes any commitment computations into explicit calls to subroutines. (We 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 $\mathsf{A}_0(\mathcal{V}^*)$ is efficient, so is $\widetilde{\mathsf{A}}_0(\mathcal{V}^*)$. This is follows from Example 5.25. Here, efficient means CEPT, in the sense that for every $\mathcal{G}, \mathcal{D}$, if the completed system for $\mathsf{A}_0(\mathcal{V}^*)$ is CEPT, so is the completed system for $\widetilde{\mathsf{A}}_0(\mathcal{V}^*)$. (PPT, CPPT and EPT efficiency can be defined analogously.)

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

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 R, which encapsulates the reduction given in the games following $\mathsf{G}_1$ in Section 5.3.2, such that for R, it holds that $\widetilde{\mathsf{A}}_0(\mathcal{V}^*) \equiv \mathrm{R}^{\mathcal{O}_0}(\mathcal{V}^*)$ and $\mathrm{R}^{\mathcal{O}_1}(\mathcal{V}^*) \equiv \widetilde{\mathsf{A}}_1(\mathcal{V}^*)$. Moreover, $\mathrm{R}^{\mathcal{O}_0}(\mathcal{V}^*)$ is efficient if $\widetilde{\mathsf{A}}_0(\mathcal{V}^*)$ is. Furthermore, since $\mathcal{O}_0$ and $\mathcal{O}_1$ are indistinguishable, if $\mathrm{R}^{\mathcal{O}_0}(\mathcal{V}^*)$ is CEPT, so is $\mathrm{R}^{\mathcal{O}_1}(\mathcal{V}^*)$. (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:

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

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

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

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

Our definition of benign simulation requires a security proof as sketched in Example 5.31, and is basically an abstract formalisation of that proof strategy. For completeness, we give a more traditional approach in Appendix E, 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 necessary.

*Definition* 5.32 (Benign simulation). Let $(\mathscr{P}, \mathcal{V})$ be a proof system. Let $\mathsf{Sim}$ be a *(timed) bb-rw* simulator with **associated rewinding strategy** $\mathsf{RWS}$ and **associated simple reduction** $\mathrm{R}$ under simple assumption $(C_0, C_1)$. A *simple reduction* under an (implicit) simple assumption $(C_0, C_1)$ is an oracle algorithm $\mathrm{R}$ which expects access to an oracle $C_b$ and the code of the adversary (usually bb-rw access suffices), i.e. $\mathrm{R}^{C_b}(\mathcal{A})$.

Suppose that:

(1) $\mathsf{RWS}$ is a *normal* rewinding strategy.
(2) $\mathsf{RWS}^{\mathcal{V}^*} \equiv \mathrm{R}^{C_0}(\mathcal{V}^*)$ and $\mathrm{R}^{C_0}$ is efficient relative to $\mathsf{RWS}$ with runtime tightness $(\mathsf{poly}^{\mathsf{RWS}}_{\mathsf{time}}, \mathsf{poly}^{\mathsf{RWS}}_{\mathsf{virt}})$.
(3) $\mathrm{R}^{C_1}(\mathcal{V}^*) \equiv \mathsf{Sim}(\mathcal{V}^*)$ and $\mathsf{Sim}$ is efficient relative to $\mathrm{R}^{C_1}$ with runtime tightness $(\mathsf{poly}^{\mathsf{Sim}}_{\mathsf{time}}, \mathsf{poly}^{\mathsf{Sim}}_{\mathsf{virt}})$.
(4) $C_0$ and $C_1$ form a simple assumption, and are indistinguishable, i.e. $C_0 \overset{c}{\approx} C_1$.

Then $\mathsf{Sim}$ is **benign** (under the assumption $C_0 \overset{c}{\approx} C_1$).

Moreover, $\mathsf{Sim}$ is *benign under size-guarding*, if it is benign whenever a polynomial size-guard is imposed on the protocol.

To summarise, Definition 5.32 abstracts simulators whose security is proved in following steps: First, exhibit a normal rewinding strategy $\mathsf{RWS}$. Second, find an reduction between $\mathsf{RWS}$ and $\mathsf{Sim}$ which proves their outputs (and implicitly, also their queries) indistinguishable. By assuming $C_0$, $C_1$ are indistinguishable, we get the desired output quality and preserve CEPT. Third, we have to take care of runtime. This is done by requiring almost every step in the chain from $\langle \mathscr{P}, \cdot \rangle$ to $\mathsf{RWS}$ to $\mathrm{R}^{C_0}$ and from $\mathrm{R}^{C_1}$ to $\mathsf{Sim}$ to be relatively efficient with runtime tightness. (Normality of $\mathsf{RWS}$ implies this for the first step.) The exceptional step from $\mathrm{R}^{C_0}$ to $\mathrm{R}^{C_1}$ follows by indistinguishability of $C_0$ and $C_1$. Indeed, since (strong) relative efficiency is unconditional, we cannot expect $\mathsf{Sim}$ to be efficient relative to $\mathscr{P}$. Hence, it is necessary to break the chain at some point.

The use of simple assumptions will only become clear once we turn to sequential composition.

### 5.5.1. Iterated benign reductions

Our definition of benign allows only *one* "reduction step" using $C_0 \overset{c}{\approx} C_1$. Many security proofs can be squeezed into this setting. However, a simple relaxation is possible.

*Definition* 5.33 (Iterated benign). In the setting of Definition 5.32, we call $\mathsf{Sim}$ **iterated benign**, if there is a *constant* $k$ and a sequence of "intermediate simulators" $\mathsf{Sim}_0, \ldots, \mathsf{Sim}_k$, with the same interface as $\mathsf{Sim}$, so that

(1) $\mathsf{Sim}_0 = \langle \mathscr{P}, \cdot \rangle$ and $\mathsf{Sim}_k = \mathsf{Sim}$.
(2) $\mathsf{Sim}_i$ and $\mathsf{Sim}_{i+1}$ are related by a benign reduction (as in Definition 5.32, with oracles $C_{i,b}$, $i = 1, \ldots, k$, $b = 0, 1$.).

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, but see Remark 5.36. Thus, hybrid arguments must be fitted into the (simple) assumptions. Indeed, a general abstract way to cope with hybrid arguments in our a posteriori efficiency setting is an open problem.

### 5.5.2. Examples of (iterated) benign simulators

All of our examples can be easily expressed via (iterated) benign simulators, and we are not aware of any counterexamples (in the case of zero-knowledge). We stress that hybrid arguments must be incorporated into the (simple) assumptions $\mathcal{C}_{i,0} \overset{c}{\approx} \mathcal{C}_{i,1}$.

*Example* 5.34. The classic, the constant round, and the concurrent zero-knowledge protocol examples [GMW86; Blu86; GK96; Ros04; KP01; PTV14] from Section 5.4.3 have benign simulation.

### 5.5.3. 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 5.35.** *Suppose $(\mathcal{P}, \mathcal{V})$ is a proof system. Let* Sim *be a benign simulator (under size-guarding). Then* Sim *is a zero-knowledge simulator (under size-guarding) which handles CEPT adversaries (in CEPT).*

*Proof.* Suppose $(\mathcal{G}, \mathcal{V}^*, \mathcal{D})$ is any adversary which is CEPT in the real protocol. Recall that the output of a normal rewinding strategy RWS is distributed like the real protocols output. By our assumption, the "reduction" $\mathrm{R}^{\mathcal{C}_0}$ has output which is also distributed exactly like RWS. By normality of RWS and relative efficiency R, $\mathrm{R}^{\mathcal{C}_0}(\mathcal{V}^*)$ is CEPT. By indistinguishability of $\mathcal{C}_0$ and $\mathcal{C}_1$ and the standard reduction, $\mathrm{R}^{\mathcal{C}_1}(\mathcal{V}^*)$ 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 $\mathcal{G}, \mathcal{D}$). Since $\mathrm{R}^{\mathcal{C}_1}$ is distributed exactly as Sim, the output of $\mathsf{Sim}(\mathcal{V}^*)$ and $\langle \mathcal{P}, \mathcal{V}^* \rangle$ is indistinguishable. Thus Sim handles CEPT adversaries in CEPT. The argument applies without change in the case of size-guarded zero-knowledge simulation. □

By Lemma 5.35, all of our examples in Example 5.34 are not only secure against a priori PPT adversaries, but have CEPT simulation against *designated CEPT* adversaries.

*Remark* 5.36 (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 timing of commitment computations). 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. Sequential composition of zero-knowledge

We define and prove sequential composition for benign simulators. Trouble with the efficiency of hybrid arguments affects sequential composition of general zero-knowledge. Thus, it is unclear whether *auxiliary input* zero-knowledge as in Definition 5.2 composes sequentially in the usual sense. As a remedy, we define *sequential* zero-knowledge, which comes with a builtin hybrid argument. Moreover, we show that benign simulation composes sequentially and consequently satisfies sequential zero-knowledge.

## 6.1. Security definition

To have a somewhat liberal notion of sequential composition, we upgrade the input-generating machine $\mathcal{G}$ to an (interactive) environment $\mathcal{E}$. We also merge it with the distinguisher. The environment $\mathcal{E}$ provides all inputs for the protocol, but does not execute the protocol itself. (This would take us too far into environmental security and concurrent composition.) Instead, there still is an adversary $\mathcal{V}^*$. One should think of $\mathcal{V}^*$ as a universal adversary, which is told how to act by $\mathcal{E}$, but $\mathcal{V}^*$ can only communicate

its output after the protocol execution back to $\mathcal{E}$. In other words, our definition of sequential composition assumes adaptive sequential executions.

Ignoring efficiency issues, our definition of sequential zero-knowledge can be summarised 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 $\mathsf{rep}(\mathsf{Sim}(\mathcal{V}^*))$. The formal definition follows.

*Definition* 6.1 (Sequential zero-knowledge). Let $\mathcal{T}, \mathcal{S} \in \{\mathcal{PPT}, \mathcal{CPPT}, \mathcal{EPT}, \mathcal{CEPT}\}$. Let $(\mathcal{P}, \mathcal{V})$ be a proof system (with efficient prover). A **universal simulator** $\mathsf{Sim}$ takes as input $(x, \mathsf{code}(\mathcal{V}^*), \mathit{aux})$ and simulates $\mathcal{V}^*$'s output. Let $(\mathcal{E}, \mathcal{V}^*)$ be an adversarial environment $\mathcal{E}$ and an adversarial verifier $\mathcal{V}^*$. The environment is given access one of two oracles $\mathcal{O}_{\mathcal{P}}$, $\mathcal{O}_{\mathsf{Sim}}$, which take as input $(x, w, \mathit{aux})$ 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 $\mathsf{Sim}(x, \mathsf{code}(\mathcal{V}^*), \mathit{aux})$. $(\mathcal{O}_{\mathsf{Sim}} \mathrel{\widehat{=}} \mathsf{rep}(\mathsf{Sim}(\cdot)))$

W.l.o.g., the output of $\mathcal{E}$ is a bit.[40] 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:

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

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

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

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

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

- $\mathsf{time}_{\mathsf{Sim}}(\mathsf{Ideal}^{\mathsf{Sim}}_{\mathcal{E},\mathcal{V}^*}) \in \mathcal{S}$, i.e. $\mathsf{Sim}$ is weakly $(\mathcal{T}, \mathcal{S})$-efficient relative to $\mathcal{P}$.
- $\mathrm{Adv}^{\mathsf{zk}}_{\mathcal{E},\mathcal{V}^*}(\kappa)$ is negligible

We define **(uniform) sequential zero-knowledge w.r.t. (input) size-guarded** security as follows: For any polynomial size-guard $\mathsf{gd}$, the derived protocol, where prover and verifier abort with `gderr` on inputs $(x, w)$ where $|x| > \mathsf{gd}(\kappa)$, is sequential zero-knowledge (in the above sense).

The definition of **non-uniform** (size-guarded) sequential zero-knowledge is analogous to the above, but $\mathcal{E}(\kappa)$ has access to an advice via an additional input interface, see Section 4.1.1.

We also say that protocols with sequential zero-knowledge simulators *compose sequentially*. We dropped the input generating machine $\mathcal{G}$, since its complexity class is the same as that of the environment $\mathcal{E}$; to model precomputation, one may want to re-introduce it, see Appendix A.5.

*Remark* 6.2 (Auxiliary input and one-query sequential zero-knowledge). The definitions of auxiliary input zero-knowledge in Definition 5.2 does not (formally) coincide with sequential zero-knowledge restricted to one query, and is minimally weaker (Remark 4.2). However, we know of no example where a difference manifests.[41]

[40] More generally, the output of real and ideal executions must be indistinguishable. Since we require PPT (or equivalently CEPT) indistinguishability, such a distinguisher can be incorporated into $\mathcal{E}$.

[41] A bb-rw simulator has no access to *aux*. So *aux* can be used to pass messages to $\mathcal{D}$. Roughly, any simulator which does not "reverse-engineer" $\mathcal{V}^*$ and *aux* should satisfy this slight strengthening of auxiliary input zero-knowledge , i.e. we know of no counterexamples even for non-black-box simulators. Intuitively, $\mathcal{G}$ 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. It is easy to see that, if $\mathsf{Sim}$ is not one-query secure, then there is a (sequence of) "keys", such that the advantage of $(\mathcal{G}, \mathcal{D})$ is at least that of $\mathcal{E}$, 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 $(\mathcal{G}_k, \mathcal{D}_k)$ 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.

## 6.2. Sequential zero-knowledge from benign simulation

Sequential composition does not follow from auxiliary input zero-knowledge in our setting.[42] Sequential composition of "standard" zero-knowledge crucially relies on *a priori efficiency*. The problem when adapting the standard proof of sequential composition can be summarised as follows: Either the hybrid argument's "pre-processing phase" (to embed the simulator) or "post-processing phase" (to produce the output the distinguisher expects) are not obviously efficient. So the core problem is efficiency of repeated applications of $\mathsf{Sim}$, or more generally hybrid arguments in an a posteriori settings (which is further complictated by virtualities). The *weak* relative efficiency guarantee in Definition 5.2 is just insufficient. Thus, we failed to lift the standard proof to the designated adversary setting. In Appendix F.5, we discuss this problem in more detail.

In order to prove efficiency, we need some kind of "uniformity" for the runtime bounds (independent of the adversary), which a posteriori efficiency does not give. Relative efficiency with runtime tightness *does* provide a uniform (tightness) bound. But as noted in Remark 5.8, we cannot hope for an unconditional property to hold for a (*computational* zero-knowledge) simulator. The main culprit is therefore the accumulation of virtualities. To "uniformly" bound virtuality changes, we face the same problem as the (classic) hybrid argument: We need a common "anchor", a *constant* number of assumptions to reduce to. Moreover, due to sequential composition, these respective assumptions are used repeatedly. This brings us to simple assumptions: They are secure under repeated trials, and an (iterated) benign reduction relies only on (a constant number of) simple assumptions.

Following the idea sketched above, we show that benign simulators sequentially compose. Conceptually we do this by:

- 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".

By "composes sequentially", we mean that there are (a priori) known universal bounds or constructions which can be used, so there is no "uncontrolled" growth of runtime or advantage.

*Remark* 6.3 (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 interested (e.g. $\mathsf{RWS}$ 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 6.4** (Sequential zero-knowledge from benign simulation)**.** *Let $(\mathcal{P}, \mathcal{V})$ be a proof system. Suppose $\mathsf{Sim}$ is a benign simulator (for auxiliary input zero-knowledge). Then $(\mathcal{P}, \mathcal{V})$ is sequential zero-knowledge.*

*The analogous claim holds for size-guarded sequential zero-knowledge.*

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

**Step 1 (Sequential composition of $\mathsf{RWS}$):** Let $\mathsf{poly}_{\mathsf{time}}$ and $\mathsf{poly}_{\mathsf{virt}}$ be the runtime and probability tightness of $\mathsf{RWS}$. Let $\mathcal{O}_{\mathsf{RWS}} \mathrel{\widehat{=}} \mathsf{rep}(\mathsf{RWS})$ denote the oracle which replaces $\langle \mathcal{P}, \cdot \rangle$ with $\mathsf{RWS}(\cdot)$. We know that for *any input*, the state of $\mathcal{V}^*$ after $\mathsf{RWS}$ is identically distributed to the state after interaction with $\mathcal{P}$ (by normality). Hence, replacing $\mathcal{O}_{\mathcal{P}}$ with $\mathcal{O}_{\mathsf{RWS}}$ only affects the runtime. Now, we lift Lemma 5.20 to the sequential setting.

[42] More precisely, we cannot prove or disprove that it does.

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

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

where normality is applied for each $i$.

Suppose $(\mathcal{E}', \mathcal{V}')$ are `timeout`-modifications according to Lemma 4.21. 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; and we lifted Lemma 5.20.

**Step 2 (Relative efficiency composes sequentially):** Suppose Sim is efficient relative to $\mathrm{R}^{\mathcal{C}_1}$ with runtime tightness $(\mathsf{poly}_{\mathsf{time}}, \mathsf{poly}_{\mathsf{virt}})$. Then the oracle $\mathcal{O}_{\mathsf{Sim}}$ is efficient relative to $\mathcal{O}_{\mathrm{R}^{\mathcal{C}_1}}$ with runtime tightness poly. Namely, for any $(\mathcal{E}, \mathcal{V}^*)$,

$$\begin{aligned}\mathbb{E}(\mathsf{time}_{\mathsf{Sim}+\mathcal{V}^*}(\langle \mathcal{E}, \mathcal{O}_{\mathsf{Sim}}\rangle)) &= \sum_i \mathbb{E}(T_{\mathsf{Sim},i}) \\ &\leq \mathsf{poly}_{\mathsf{time}} \sum_i \mathbb{E}(T_{\mathrm{R}^{\mathcal{C}_1},i}) \\ &= \mathsf{poly}_{\mathsf{time}} \cdot \mathbb{E}(\mathsf{time}_{\mathrm{R}^{\mathcal{C}_1}}(\langle \mathcal{E}, \mathcal{O}_{\mathscr{P}}\rangle))\end{aligned}$$

where $T_{\mathsf{Sim},i}$ resp. $T_{\mathrm{R}^{\mathcal{C}_1}}$ denotes the time for the $i$-th invocation of the respective oracle. This again follows by looking 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 $(t, \varepsilon)$ with $\mathcal{O}_{\mathrm{R}^{\mathcal{C}_1}}$, 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 $\mathcal{O}_{\mathrm{R}^{\mathcal{C}_0}}$.

**Step 3 (Indistinguishability of $\mathcal{O}_{\mathrm{R}^{\mathcal{C}_0}}$ and $\mathcal{O}_{\mathrm{R}^{\mathcal{C}_1}}$):** It is obvious that indistinguishability of $\mathcal{O}_{\mathrm{R}^{\mathcal{C}_0}}$ and $\mathcal{O}_{\mathrm{R}^{\mathcal{C}_1}}$ reduces to indistinguishability of $\mathcal{C}_0$ and $\mathcal{C}_1$ under repeated trials. (Each invocation of $\mathcal{O}_{\mathrm{R}^{\mathcal{C}_0}}$ (resp. $\mathcal{O}_{\mathrm{R}^{\mathcal{C}_1}}$) is another trial.) By Corollary 4.15, simple assumptions are indistinguishable under repeated trials. (It is vital that $\mathrm{R}^{\mathcal{C}_0}$ 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 $\mathcal{O}_{\mathsf{Sim}}$ again CEPT. □

# 7. Conclusion and open problems

At the example of zero-knowledge, 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 CEPT and superpolynomial hardness assumptions.* The problem is the runtime-advantage trade-off which is possible for expected time. Concretely, consider one-way function where we assume that no PPT adversary can invert with probability better than $O(2^{-\kappa/2})$. W.r.t. CEPT, such assumptions cannot exist, since with probability $O(2^{-\kappa/4})$, a CEPT adversary may brute-force a preimage. One may consider a

CEPT-variation, where instead of negligible distinguishing advantage from EPT, advantage at most $O(2^{-\kappa/2})$ is required. However, if we do this, then all indistinguishability assumptions must be so strong. Mixing in a primitive with negligible advantage would allow a deviation from EPT which is too far.

It is a critical question, whether this is a fundamental problem, or just another technical artifact. If CEPT is incompatible with subexponential hardness assumption, then protocols which rely on such are very likely incompatible with CEPT. We expect a suitable redefinition of advantage to fix this problem in a natural way — but which definition is the "right" one?

**Broader applications.** For cryptographic applicability, the treatment of zero-knowledge gives hope that other simulation-based settings, e.g. *multi-party computation* or *environmental security*, are compatible with CEPT, or can be made compatible.

**Basic insights and quantifiability.** For a more quantifiable notion of security, we need to tackle the question of *tightness* of reductions, simulations, etc. The treatment of the virtuality error for a good notion of tightness is non-trivial. Moreover, any application of the statistical-to-computational reduction obliterates tightness.

Furthermore, a more insightful interpretation of (or solution to) the problem of *hybrid arguments* and (sequential) composition are of particular interest.

**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, and to Marcel Tiepelt, whose suggestions helped to improve the overall presentation.

## 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.

[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.

[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.

[CS02] Ronald Cramer and Victor Shoup. “Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption”. In: *EUROCRYPT*. Vol. 2332. Lecture Notes in Computer Science. Springer, 2002, pp. 45–64.

[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.

[FW93] Michael L. Fredman and Dan E. Willard. “Surpassing the Information Theoretic Bound with Fusion Trees”. In: *J. Comput. Syst. Sci.* 47.3 (1993), pp. 424–436.

[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.

[KW15] Eike Kiltz and Hoeteck Wee. “Quasi-Adaptive NIZK for Linear Subspaces Revisited”. In: *EUROCRYPT (2)*. Vol. 9057. Lecture Notes in Computer Science. Springer, 2015, pp. 101–128.

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

[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.

[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.

[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 requirements.[43] As our machine model, we have some RAM-like model in mind. Indeed, "concrete efficiency" is relatively important when dealing with *expected* time. For example, if a (runtime) distribution $T$ over $\mathbb{N}_0$ has finite expected value $\mathbb{E}(T)$, this need not be the case for $\mathbb{E}(T^2)$.[44] 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).

## 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. 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 convenient in our setting, where input (resp. output) lengths are not a priori bounded.

We do not formalise 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 levels of abstractions: *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 realisation of $S$ is a deterministic system. 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 callee when answering a query (or report `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 runtime*.

[43] Most likely, we forgot some requirements. Also, our specification is far from formal. Worse, these requirements partially depend on runtime and efficiency notions. However, polynomial time should be unproblematic,

[44] Consider distributions $T_\kappa$ over $\mathbb{N}$ as follows: $\mathbb{P}(T_\kappa = 2^\kappa) = 2^{-\kappa}$, and $\mathbb{P}(T_\kappa = 1) = 1 - 2^\kappa$. Then $\mathbb{E}(T_\kappa) = 1$, but $\mathbb{E}(T_\kappa^2) \geq 2^\kappa$.

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. But hardness assumptions, such as timelock puzzles are void against timeful oracles. Thus, they are mostly a convenient way to formalise unconditional runtime guarantees for algorithms with oracle-access, e.g. bb-rw simulators. 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:[45]

**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.[46]
**Interactive machines** which communicate and are activated in some sensible way.
**A sensible notion of runtime** which is *local* in case of interactive machines and subroutine calls. That is, one can separate between time spent within some machine, subroutine, or oracle, and account accordingly.
**Efficient emulation** ensures that one can efficiently execute some code (e.g. of the adversary or an interactive system) "in-the-head", or in modern terms, efficiently run one (or many interacting) "virtual machines".

Let us formalise 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 B.3. 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 constant size tuples.

Now, we somewhat formalise the locality of runtime. Let $\mathsf{A}^{\mathcal{O}_1,\dots,\mathcal{O}_N}$ be an oracle machine (with access to $N$ oracles).[47] We write $\mathsf{time}_{\mathsf{A}}(\mathsf{A}^{\mathcal{O}_1,\dots,\mathcal{O}_N})$ or $\mathsf{time}(\mathsf{A}^{\mathcal{O}_1,\dots,\mathcal{O}_N})$ to denote the runtime[48] of $\mathsf{A}$ only, where each oracle invocation costs a single unit of time. We call this notion of time **oracle-excluded time**. By $\mathsf{time}_{\mathsf{A}+\mathcal{O}_1+\dots+\mathcal{O}_N}(\mathsf{A}^{\mathcal{O}_1,\dots,\mathcal{O}_N})$, we denote the time spent by all of the machines. We call this **oracle-included-time**. We define intermediate times, e.g. $\mathsf{time}_{\mathsf{A}+\mathcal{O}_2}(\mathsf{A}^{\mathcal{O}_1,\dots,\mathcal{O}_N})$ in the obvious way. We generally require that

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

or something morally equivalent. (Note that our algorithm takes no input. In case of randomised algorithms, the runtimes for $\mathsf{A}$, $\mathcal{O}_1$, ...$\mathcal{O}_N$ are *not* stochastically independent.)

Finally, a sensible machine model guarantees efficient emulation. Namely, if the *oracle-included time* of $\mathsf{A}^{\mathcal{O}_1,\dots,\mathcal{O}_N}$ is efficient so is the runtime of the algorithm $\mathcal{B}$ which *emulates* the execution on the oracles.

[45] Another requirement, which is natural enough that we did not prominently require it, is that writing (or sending) a message of length $n$ incurs $n$ steps. Otherwise, message length and runtime efficiency are "uncorrelated".

[46] 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.

[47] Notations and properties for runtime of interactive machines are analogous.

[48] Recall that we say runtime for *runtime distribution* and that we consider algorithms without input (for defining runtime).

In other words, converting an oracle (or interactive) machine into a single machine $\mathcal{B}$ by incorporating the oracle (via its code) should *preserve efficiency*. Furthermore, emulation should efficiently allow to gather (and act upon) 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). That is $k$ timesteps of the oracle(s) can be emulated in at most $\mathsf{emuovhd}(k)$ timesteps (of emulation). Usually, the security parameter $\kappa$ and number of oracles $N$ are suppressed.

Lastly, we define **timeful oracles** (or *timeful systems*) as oracles which also return their "elapsed time" to the machine model (and not as regular output). This is a means to give oracles a "special" notion of time, e.g. because the oracle is "imaginary" and not (efficiently) computable. For consistency, their elapsed time must adhere to lower bounds, such as an $m$-word output requiring at least $m$ steps. This "manipulation of reality" is a convenient tool.

**The interaction model.** We will assume an interaction 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.

## 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.1 (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, the $\mathcal{O}$ is *deterministic* in the sense that the randomness of the emulated programs is sampled and fixed prior to interaction.[49] We do *not* let the oracle algorithm 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 $a$.
- **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

[49] That is, $\mathcal{O}$ reacts deterministically to queries in one execution, but not necessarily over different executions. When such an oracle is implemented, the "random tape" (or the respective notion in the machine model) is sampled (and fixed) lazily. For example, a random oracle is deterministic.

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) allow the state of the emulated program to be stored and loaded. Typically, a state is identified by its partial transcript of (previous) queries. Other means of identification, such as handles, are more efficient. Loading, storing, and deleting program states is done by special types of messages.[50]

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 in setting.

*Example* A.2 (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. Thus, it requires about $\sum_{i=1}^{n} i \approx \frac{1}{2}n^2$ steps.

*Remark* A.3 (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. (Poly-logarithmic overhead is admissible by Corollary A.7.)

*Remark* A.4. 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.5 (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*, but *strict poly-space*, is a "more natural" class of feasible computation than expected poly-time and expected poly-space.[51]

Unfortunately, this unveils technical artifacts. Depending on the implementation of the randomness interface (e.g. input, read-only tape, coin-toss, ...) emulation and bb-rw oracle implementations may become inefficient, 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 derandomisation, e.g. with an (a priori PPT) pseudorandom function, to simulate a long enough random string with small 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 is related to [BG11].

Similar problems apply to non-uniform advice, but their effect is worse, since non-uniform advice cannot be "resampled" by a uniform simulator. So for general (pathological) algorithms, it is plausible,

[50] 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.

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

that *uniform* bb-rw simulation requires higher space complexity.[52,53] Viewing non-uniform advice as part of the algorithm's code, this is not surprising, since the algorithm had a higher space complexity to begin with.

## 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.[54] Following trivial lemma is useful to see that efficient emulation is not hard to achieve, even for *expected* time.

**Lemma A.6.** *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 $\kappa$). 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.7.** *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.6 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 $2^\kappa$ steps.[55] This induces a statistically negligible change in the output of any expected polynomial time algorithm. 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.8 (Interaction of Corollary A.7 and virtuality). CEPT and CPPT ignore negligible events, because they can be hidden in the virtuality. So, Corollary A.7 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. (It can be a problem for EPT, if an algorithm is not truncated to $2^\kappa$ steps. Unfortunately, such a truncation can affect *perfect* properties, such as perfect correctness, leading to technical artifacts.)

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

*Example* A.9 (Single-tape Turing machines are no good). 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 recognise this language. Thus, the emulation overhead is most likely quadratic. Hence, it is very unlikely that single-tape Turing machines are an admissible model of computation.

*Example* A.10 (RAM models). Various RAM models seem appropriate for our cause. A transdichotomous model of computation [FW93], in which the RAM's word size grows with the problem, seems particularly well-suited for cryptography; indeed security parameter $\kappa$ is a natural measure for the problem size.

[52] Given an oracle $\mathcal{O}$, it can be made deterministic by coin-fixing. Moreover, assuming (strong) problems exist which are hard for uniform machines, but easy for non-uniform machines, one can make execution depend on solutions for such problems. An emulation $\mathcal{O}'$ of $\mathcal{O}$ can check for this and aborts if no solution is provided. For example, assuming suitable (multi-)collision-resistance properties of a hash function, the $i$-th emulated step may check whether a collision for $H(i)$ is provided in the advice. A uniform bb-rw simulation needs to "compress" the collisions, or suffer from linear space consumption in the runtime of $\mathcal{O}$.

[53] For *non-uniform existential* simulators Sim, suitable non-uniform advice may exist. (E.g. it may be constructed from the advice of $\mathcal{O}$, by repeating the advice per rewind round, and perhaps suitably intermingling.) Note that, except for existential advice of Sim, Sim may be black-box. Nevertheless, this shows that *non-uniform* existential simulation and universal simulation need not coincide if access to *super-polynomial* advice is given, because a universal simulator's advice must be independent of the adversary's advice.

[54] We have not carried out formal proofs.

[55] Technically, we have to do an earlier cutoff, since emulation and cutoff also consume runtime. But this is a minor issue.

### A.5. Precomputation and non-uniformity

In our setting, the input generating machine, may also explicitly model (adversarial) precomputation. For simplicity, we only deal with the simple case of non-uniform advice.

To have a definition of non-uniform *expected time* machines, we propose an advice interface just like the randomness interface.[56] That is, the advice string has infinite length. Alternatively, one could restrict to strict polynomial size advice, but this conflates machine model and security model. Indeed, an expected polynomial time input generating machine may generate an expected polynomial size "advice" (whose size is not strictly polynomially bounded). Note that non-uniformity comes with its own more or less subtle anomalies, see e.g. [KM13].

## B. Technical lemmata

In this section, we gather some lemmata for various purposes. Appendix B.1 contain some simple facts on statistical distance. In Appendix B.2, some cryptographic results concerning distinguishing and general hybrid arguments are given. And Appendix B.3 contains naive closeness tests.

The reader should skip to Section 2.4.1 for the definition and notation of *tail bounds*, which are used in Appendix B.2 and Appendix B.3.

### B.1. 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.

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

**Lemma B.1.** *Let $p_i, q_i \in [0,1]$ for $i = 1, \dots, n$. Then*

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

*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*

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

*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* B.2. Let $X$ be random variable and let $Y$ independently distributed like $X$ conditioned on some event of probability $\varepsilon$. 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.

[56] The advice interface should follow the same restrictions as the "random tape' (see Remark A.5), in particular it should not provide memory to not conflate advice complexity with space complexity.

**Corollary B.3.** *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 $\ell_1$-norms and the claim follows by general inequalities, see Lemma B.4. □

**Lemma B.4.** *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. □

## B.2. 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.

### B.2.1. Runtime truncations

We give generic variants of runtime truncation lemmata.

**Corollary B.5.** *Suppose* $\mathsf{A}$ *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 B.5 bounds the quality loss when converting expected to strict time algorithms. For example, if $\mathsf{A}$ is a distinguisher with advantage $\varepsilon$, 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 B.6** (Non-asymptotic generic "standard reduction")**.** *Suppose $\mathcal{D}^{\mathcal{O}}$ is a distinguisher with advantage $\varepsilon$ 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 4.4) in the general setting. We point out, that runtime is not the only (complexity) measure of interest which can be used in Corollary B.6. 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 B.6 generalises 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 $\delta$ 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 $\frac{\varepsilon}{4}$.

### B.2.2. 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* B.7 (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 $\kappa$ protocols by simulations, the remainder can be replaced in a single step, because more than $\kappa$ repetitions are necessary only with probability $2^{-\kappa}$.

We state in general the truncation approach from Example B.7.

**Corollary B.8** (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 $\mathsf{Z}$ be an algorithm as follows: $\mathsf{Z}$ 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 $\varepsilon$, 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 $\mathsf{bnd}$ 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*[57]

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

*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 $\mathsf{Z}$.*

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

*Proof of Corollary B.8.* 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 $\mathcal{O}^0, \dots, \mathcal{O}^{N_\alpha}$. Because, by the very definition $N_\alpha$ 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.[58] It allows an a priori infinite number of hybrids. And it postulates a bound $\mathsf{bnd}$ on the closeness of the $i$-th and final hybrid. Typically $\mathsf{bnd}$ bounds the statistical distance of the $i$-th and final hybrid and is derived as a tail bound, e.g. Markov bound (Lemma 2.8) 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* B.9 (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. Suppose $\mathcal{A}$ is an adversary which perfectly distinguishes pseudo-random and random strings. Let $\varepsilon$ be a "target advantage" and suppose $N = 1$ with probability $1 - \varepsilon$, and $N = q := \lceil \varepsilon^{-1} \rceil$ with probability $\varepsilon$. Construct the adversary $\mathcal{B}$ which draws $N$, guesses randomly if $N < q$. Else, $\mathcal{B}$ runs $\mathcal{A}$ on all challenges. If $r_1, \dots, r_i$ are pseudo-random, $\mathcal{B}$ outputs the correct guess with probability $\frac{i}{q}\varepsilon$ (and a random guess otherwise). Thus, $\mathcal{B}$ has advantage $\varepsilon$ by construction, yet the hybrid argument achieves advantage $\frac{\varepsilon}{q}$ at best.

Example B.9 shows that tails of distributions are a limiting factor, and it is non-obvious how to improve the (truncated) hybrid argument Corollary B.8 in a black-box-like way. Nevertheless, Corollary B.8 is useful and generally good enough, though it may have poor tightness properties.

## B.3. 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* B.10 (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 $X \overset{\mathcal{D}}{\equiv} Y$ or $\Delta(X, Y) > \varepsilon$. A tester $\mathsf{A}$ 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) $\delta$, if

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

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

**Lemma B.11.** *Let $X, Y$ be distributions on $\{0, \dots, n\}$ and consider the closeness promise problem. Let $\varepsilon, \delta \in (0, 1]$. Then there is an algorithm $\mathsf{A}$ which solves the closeness promise problem with error $\delta$ and requires*

$$N = \lceil 6(n+1)\varepsilon^{-2} \log(2\delta^{-1}) \rceil$$

*samples (of both $X$ and $Y$). Moreover, $\mathsf{A}$ is makes a linear number of arithmetic operations (in $N$).*

[58] 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 B.8.

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 B.11.* 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 $X \overset{\mathcal{D}}{\not\equiv} Y$, and using the sharp Dvoretzky–Kiefer–Wolfowitz inequality by Massard in case $X \overset{\mathcal{D}}{\equiv} Y$, 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 B.11 to this case.

**Corollary B.12.** *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 $n' = \mathsf{tail}_X^\dagger(\alpha)$. Then there is an algorithm $\mathsf{A}$ which solves the closeness promise problem with error $\delta$ and requires*

$$N' = \lceil 6(n'+1)\varepsilon'^{-2} \log(2\delta^{-1}) \rceil$$

*samples (of both $X$ and $Y$). Moreover, $\mathsf{A}$ 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 B.12 holds.

*Proof.* The algorithm simply maps the distributions $X, Y$ to new distributions by mapping any sample $s$ to $\max\{s, n\}$.[59] This changes the statistical distance by at most $\mathsf{tail}_X(n)$, see Corollary B.3. Now, apply Lemma B.11. □

Following remark, while a triviality, points out one core tool of this work.

*Important Remark* B.13 (Statistical and computational indistinguishability coincide for "small" support). From Lemma B.11 and Corollary B.12, we already observe the following: Asymptotically, any pair of (families of) distributions $X$, $Y$, where one, say $X$, has (essentially) polynomial sized support $\{0, \dots, \mathsf{poly}(\kappa)\}$ are *computationally* indistinguishable in polynomial time, if and only if, they are *statistically* indistinguishable (under repeated sampling).

*Remark* B.14. 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 $\{0, \dots, n\}$. 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.

# C. 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.

[59] 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.

## C.1. Preliminaries: Bound algebras

Most of our arguments work for runtime classes related to bound algebras, for example, the algebra of polynomials.

*Definition* C.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.[60]
- $\mathcal{B}$ 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 $\kappa$).
- $\mathcal{B}$ is "asymptotically monotone": If $(x_\kappa)_\kappa \in \mathcal{B}$, then so is $(y_\kappa)_\kappa$ with $y_\kappa := \max_{i=1}^{\kappa} x_i$.

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* C.2. Suitable function algebras, e.g. polynomials, or polylogarithmic functions, or $f(\kappa) = n^{\mathsf{polylog}(\kappa)}$, etc., induce a (general) 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 $\mathcal{B}$.

## C.2. Runtime distributions

Our definitions of (polynomial) runtime are such that an algorithm's (or protocol's) runtime is bounded *in the security parameter $\kappa$ alone*. The input space of an algorithm is (a family) $\mathcal{X}_\kappa$.[61] 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* C.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 $\kappa$. The security parameter is (implicit) "input" to every algorithm. In fact, usually, $\mathsf{A}$ is given no inputs (but $\kappa$). Similarly, runtime obviously depends on the machine model even though we do not mention this.

*Definition* C.4 (Runtime distribution). A *(input-free)* **runtime (distribution)** $T$ is a family $(T_\kappa)_\kappa$ of distributions $T_\kappa \in \mathsf{Dists}(\mathbb{N}_0 \cup \{\infty\})$ parameterised by $\kappa$; 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 $\mathsf{A}$ if $T_\kappa = \mathsf{time}_{\mathsf{A}}(\mathsf{A}(\kappa))$. We typically suppress $\kappa$ and simply write $T = \mathsf{time}_{\mathsf{A}}(\mathsf{A})$.

We allow the symbol `timeout` in a runtime distribution $T$ (formally changing to $\mathsf{Dists}(\mathbb{N}_0 \cup \{\texttt{timeout}\})$).[62]

*Remark* C.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 $\kappa$. It is **induced** by $\mathsf{A}$ 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* C.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 $\mathsf{A}$ and $\mathcal{O}$).

[60] The associated subalgebra of $\mathcal{B}$ is unique.

[61] Recall a well-known problem: The input space may not be (efficiently) recognisable. 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.

[62] We could also allow $\infty$ there, but generally timeouts stop overlong executions.

## C.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 $\mathcal{EPT}$.

*Definition* C.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 $S \in \mathcal{T}$.[63]
**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 $\mathsf{A}$ 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 F.11 for more.

*Example* C.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 $\mathsf{poly}$ 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 $\mathsf{poly}$ such that $\mathbb{E}(T) \leq \mathsf{poly}$, i.e. $\mathbb{E}(T_\kappa) \leq \mathsf{poly}(\kappa)$ for all $\kappa$.
**Polynomial $\|\cdot\|_q$-time:** By polynomially bounding $\|T\|_q$ (for $q \in [1, \infty]$), we generalise 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 generalise polynomial time bounds to algebra bounds. For that, we need following definition.

*Definition* C.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 $\mathsf{bnd}_1$ can be computed in $\mathcal{T}$-time. More concretely, $\mathsf{bnd}_1(\kappa)$ can be computed in time $T_\kappa$ for $T \in \mathcal{T}$.

We call $\mathcal{T}$ **(strongly) compatible** with $\mathcal{B}$ if additionally strict $\mathcal{B}$-time (see Example C.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.

*Example* C.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 C.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 generalise 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 $\kappa$.[64]

We also refer to algebra-bounded times via *strict (or expected)* $\mathcal{B}$*-time*.

By Lemma 2.8, 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.

[63] More precisely, $S \leq T$ iff for all $\kappa$ we have $S_\kappa \leq T_\kappa$, i.e. $T_\kappa$ dominates $S_\kappa$ in distribution.
[64] Recall that asymptotics, should be part of $\mathcal{B}$, so we use for *all* $\kappa$ (and not for almost all).

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 F.11.

Lastly, we define "abstract" runtime cutoffs.

*Definition* C.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.[65]

*Remark* C.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 Example C.10.

## C.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 $Z \overset{\mathcal{D}}{\equiv} X_b$ mirrors the fact that an algorithm can be (independently) executed many times, and should still remain efficient.[66] 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 $\mathsf{A}_1$. To simplify, we assume sampling access to both $X_0$ and $X_1$.

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* C.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** $b' \overset{?}{=} b$

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 $\mathsf{time}_{\mathcal{D}}(\mathrm{Exp}^{\text{3-dist}}_{\mathcal{A},\mathcal{O}_0,\mathcal{O}_1}) \in \mathcal{T}$.[67] 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 $\mathcal{D}$,

$$\mathrm{Adv}^{\text{3-dist}}_{\mathcal{D},\mathcal{O}}(\kappa) \in o(1).$$

[65] 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 recognise $t > N$ and then return `timeout`.

[66] 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].

[67] Equivalently, $\mathsf{time}_{\mathcal{D}}(\mathcal{D}^{\mathcal{O}_0,\mathcal{O}_1,\mathcal{O}_b}) \in \mathcal{T}$ for $b = 0, 1$.

that is, any distinguisher has asymptotically vanishing advantage. Put differently, a computational distinguisher must have constant advantage $c > 0$ (for infinitely many $\kappa$). We define **$\mathcal{T}$-time statistical indistinguishability** as $\mathcal{T}$-query indistinguishability, that is we only count a query to an oracle as a step (costing unit time).

We use Definition C.13 only for (runtime) distributions, and not general oracle-indistinguishability.

*Remark* C.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 $1 - \frac{1}{\kappa}$. One reason is our focus on indistinguishability of runtimes, not in general distributions. Our techniques mostly involve truncation and repetition, and thus runtime-vs-advantage trade-offs. In Section 7, we sketch concrete problems for CEPT in "practice". But even in theory, there are obstructions to a useful generalisation. We explain this in the following, but assume familiarity with CPPT, runtime closure, etc. The reader should skip this in a first reading. In the following, we consider PPT.

*The "low advantage regime":* The interaction of subexponential advantage with PPT leads to two problems: One, if CPPT is defined by poly-negligible deviation from PPT, then there are no useful subexponentially hard problems, since with poly-negligible probability, a CPPT algorithm can brute-force a solution, hence has at least poly-negligible advantage. On the other hand, if we strengthen to subexp-CPPT, i.e. subexp-negligible deviation from PPT, then our simplistic arguments, e.g. closeness testing of Lemma B.11 fail. A treatment of this is an interesting open question.

*The "high advantage regime":* High advantage classes, such as $1 - \frac{1}{\kappa}$, can be reached by amplification in the triple-oracle setting. So they are only interesting for one-shot distinguishing. However, in such settings transitivity of indistinguishability is lost. A treatment of this is beyond this work.

Since it is a useful point of view, we slightly generalise distinguishing. Namely, instead of directly outputting a verdict, one may output some processed information, which is fed into another distinguisher (perhaps repeatedly).

*Remark* C.15 (Generalised distinguisher). Let us call a distinguisher, which outputs not only 0 or 1, but different or additional information, a *generalised distinguisher*. Clearly, if two distributions are (computationally) indistinguishable, then the output of any generalised 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* C.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 $\mathcal{T}$-time (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 one-shot indistinguishability only in Appendix C.7

## C.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* C.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 1.2 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* C.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 $\mathcal{U}$.[68]

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.

*Remark* C.19. We can translate generating sets to the setting of bound algebras. Indeed, in Example C.10, we require a generating set of efficiently computable bounds.

The perhaps most important relation between sets of runtimes is the following.

*Definition* C.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 B.5.

**Lemma C.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* C.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 C.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 F.11.

[68] 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.

**Lemma C.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 $\overline{\mathcal{T}}$.[69] 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* C.24 (Closures are small). For any "benign" runtime class $\mathcal{T}$, $\mathcal{T}$ is runtime d-dense in $\overline{\mathcal{T}}$.

We expect that runtime classes where Conjecture C.24 fails behave rather strangely. While we do not know what "benign" runtime classes are or how to prove Conjecture C.24 in general, it is simple for algebra-tailed runtime classes.

**Lemma C.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 $\kappa$ and constant $\varepsilon$). 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 $\mathsf{bnd}$, 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* C.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* C.27 (Non-negligible advantage). Lemma C.25 immediately extends to advantage $\varepsilon = 1/\mathsf{bnd}(\kappa)$ (for infinitely many $\kappa$). 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 C.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 C.21 and C.23. (Any element in $\overline{\mathcal{T}}$ also lies $\mathcal{S}$.) □

Let us consider a simple concrete example.

*Example* C.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 C.6, we find that statistical and computational closure coincide, hence "CPPT = SPPT". By definition, $\mathcal{CPPT}$ is

$$\mathcal{CPPT} = \{T \mid \exists \mathsf{poly}, \mathsf{negl}\colon \mathbb{P}(T_\kappa \geq \mathsf{poly}(\kappa)) \leq \mathsf{negl}(\kappa)\}.$$

[69] Triple-oracle indistinguishability is transitive for any *constant* number of hops.

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 C.21 and Lemma C.28.

## C.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 C.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 B.11 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 B.11. Let $\mathsf{bnd}_0 \in \mathcal{B}$ bound the support size of the distributions $X, Y$.[70] 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 $\mathsf{bnd}$, and hence $X^{\{\mathsf{bnd}\}}, Y^{\{\mathsf{bnd}\}}$ are statistically close, for any (statistical) distinguisher. A contradiction to triple-oracle distinguishability.

We invoke Lemma B.11 with $n = \mathsf{bnd}_0$, $\varepsilon = \frac{1}{2\mathsf{bnd}_{\mathsf{stat}}}$, and $\delta$ 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 generalise Lemma C.30 just like we generalised Lemma B.11 to Corollary B.12.

**Corollary C.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 $\mathsf{tail}_X$ 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 B.12 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 B.12 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

$$N = \lceil 6N_\alpha(\varepsilon - \alpha)^{-2} \log(2\delta^{-1}) \rceil$$

[70] To be precise, it is lower-bounded only for infinitely many $\kappa$.

samples, where $N_\alpha := \mathsf{tail}^\dagger_X(\alpha)$ 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 C.30. Indeed, we rely on the general hybrid lemma (Corollary B.8) and hence on tail bounds. Suppose the statistical distinguisher has advantage $\geq c$ (infinitely often for constant $c$). By a standard hybrid argument, Corollary B.8, we find a distinguisher which accesses the challenge oracle only once, and has advantage at least

$$\frac{c-\beta}{N_\beta} \quad \text{where} \quad N_\beta := \mathsf{tail}^\dagger_{\mathcal{D}_{\mathsf{stat}}}(\beta) \quad \text{for any } \beta \in [0,c].$$

Consequently, $\Delta(X,Y) \geq \frac{c-\beta}{N_\beta}$ for any choice of $\beta$. (Note that $\varepsilon$ and $N_\beta$ vary in $\kappa$.)

**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 $\kappa$. 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 $\beta$, 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 C.27, one can directly generalise to non-negligible advantage.

**Corollary C.32.** *The result of Corollary C.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.*

## C.7. From oracles to emulation and one-shot 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 one-shot 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 realisation 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 $\mathsf{bnd}$ 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* C.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. $O(k\mathsf{polylog}(k))$).[71] 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.

[71] 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.

*Example* C.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 one-shot indistinguishability (under repeated sampling).

**Lemma C.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 $\mathsf{bnd}$ 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 B.6. 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 B.6. 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 $\mathsf{bnd}$ 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 C.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 C.35. This finally yields the equivalence we wanted.

**Corollary C.36** (Equivalence of triple-oracle and one-shot 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* one-shot *distinguishable with non-$\mathcal{B}$-negligible advantage. (There is $\mathcal{B}$-factor of loss involved in the reduction.)*

Finally, we stress that Corollary C.36 is a very loose reduction.

# D. Supplementary definitions

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

## D.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 $\mathcal{M}_\kappa = \{0,1\}^\kappa$.

### D.1.1. Non-interactive commitment schemes

*Definition* D.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 $\forall \mathrm{ck} \leftarrow \mathsf{Gen}(\kappa)\colon \mathsf{VfyCK}(\mathrm{ck}) = \texttt{ACC}$ and $\forall \mathrm{ck} \leftarrow \mathsf{Gen}(\kappa),\ m \in \mathcal{M}_\kappa,\ (c,d) \leftarrow \mathsf{Com}(\mathrm{ck})\colon \mathsf{VfyOpen}(\mathrm{ck}, c, m, d) = \texttt{ACC}$.

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* D.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 `win` iff $\mathsf{VfyOpen}(\mathrm{ck}, c, m_b, d_b) = \texttt{ACC}$ for $b = 0, 1$ and $m_0 \neq m_1$.

Let $\mathrm{Adv}^{\mathrm{bind}}_{\mathsf{Com},\mathcal{A}}(\kappa) = \mathbb{P}(\mathsf{Bind}_{\mathsf{Com},\mathcal{A}}(\kappa) = \texttt{win})$. Then Com is computationally (resp. statistically, resp. perfectly) **binding** for $\mathcal{T}$-time (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 oracle-excluded-time of the adversary in the 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* D.3 (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 $\mathsf{VfyCK}(\mathrm{ck}) = \texttt{REJ}$, return `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[72] returns $c_b = \mathsf{Com}(\mathrm{ck}, m_b)$.
- Return `win` if $b = b'$ else `lose`

Let $\mathrm{Adv}^{\mathrm{hide}}_{\mathsf{Com},\mathcal{A}}(\kappa) = |2\,\mathbb{P}(\mathsf{Bind}_{\mathsf{Com},\mathcal{A}}(\kappa) = \texttt{win}) - 1|$. Then Com is computationally (resp. statistically, resp. perfectly) **hiding** for any $\mathcal{T}$-time (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. "$= 0$").

Note that in Definition D.3, we could instead expose $\mathcal{O}_0$ and $\mathcal{O}_1$ accept and verify ck, i.e. completely absorb the game into the oracles. The advantage of a "rewired" $\mathcal{A}'$ is identical to that of $\mathcal{A}$. Finally, we note that CEPT adversaries are "no better" than a priori PPT adversaries.

[72] The message length is always $\kappa$ in this case.

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

*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. □

We recall that multi-challenge left-or-right hiding is equivalent to the single-challenge setting via a standard hybrid argument.

*Remark* D.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.

# E. 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.

## E.1. Query-sequences indistinguishability

Our notion of "indistinguishable queries" for simulators is similar in spirit to [KL08].

*Definition* E.1 (Query-indistinguishability). Let A and B be oracle algorithms. The distinguishing advantage $\mathrm{Adv}^{\mathrm{qseq}}_{(\mathcal{G},\mathcal{O},\mathcal{D}),\mathsf{A},\mathsf{B}}(\kappa)$ for queries *including output* of A, B by an adversary $(\mathcal{G},\mathcal{O},\mathcal{D})$ is defined as the distinguishing advantage $\mathrm{Adv}^{\mathrm{dist}}_{\mathcal{D},X,Y}(\kappa)$ for the distributions

$$X \coloneqq \{(x,y,r,\mathsf{A}^{\mathcal{O}(y;r)}(x;r_{\mathsf{A}}),\mathsf{qseq}_{\mathcal{O}}(\mathsf{A}^{\mathcal{O}(y;r)}(x;r_{\mathsf{A}}))) \mid (x,y) \leftarrow \mathcal{G}(\kappa)\}_\kappa$$
$$Y \coloneqq \{(x,y,r,\mathsf{B}^{\mathcal{O}(y;r)}(x;r_{\mathsf{B}}),\mathsf{qseq}_{\mathcal{O}}(\mathsf{B}^{\mathcal{O}(y;r)}(x;r_{\mathsf{B}}))) \mid (x,y) \leftarrow \mathcal{G}(\kappa)\}_\kappa$$

Here $r$ denotes the *accessed* randomness of $\mathcal{O}$.[73] (We make explicit the randomness $r_{\mathsf{A}}$ and $r_{\mathsf{B}}$ only to make it evident, that the output and query sequence refer to the same run.)

We say that A and B satisfy **($\mathcal{T}$-time) query-indistinguishability (Q-IND)**, if for all adversaries $(\mathcal{G},\mathcal{O},\mathcal{D})$ such that $\mathsf{time}_{\mathcal{G}+\mathcal{O}+\mathcal{D}}(\mathsf{A}^{\mathcal{O}}) \in \mathcal{T}$ and $\mathsf{time}_{\mathcal{G}+\mathcal{O}+\mathcal{D}}(\mathsf{B}^{\mathcal{O}}) \in \mathcal{T}$ the advantage $\mathrm{Adv}^{\mathrm{qseq}}_{(\mathcal{G},\mathcal{O},\mathcal{D}),\mathsf{A},\mathsf{B}}(\kappa)$ 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 E.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.

[73] Typical machine models offer an infinite pool of (independent) randomness, e.g. a random tape. Thus, we "restrict" to accessed randomness.

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* E.2. For CEPT and CPPT, it suffices in Definition E.1 to require that $T = \mathsf{time}_{\mathcal{G}+\mathcal{O}+\mathcal{D}}(\mathsf{A}^{\mathcal{O}}) \in \mathcal{T}$. If $\mathsf{time}_{\mathcal{G}+\mathcal{O}+\mathcal{D}}(\mathsf{B}^{\mathcal{O}}) \notin \mathcal{T}$, then this is a distinguishing statistic. Indeed, by a standard reduction to PPT, any distinguisher $(\mathcal{G}, \mathcal{O}, \mathcal{D})$ 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 $\mathcal{D}' \mathrel{\widehat{=}} (\mathcal{G}, \mathcal{O}, \mathcal{D})$ as interacting with oracles $\mathsf{A}$ or $\mathsf{B}$, and apply Corollary 4.4.)

*Remark* E.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 E.1 in terms of an "environment" $\mathcal{E}$ which encompasses $\mathcal{G}$ 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 E.2, we may assume $\mathcal{E}$ is a priori PPT. Also, (one-guess) "environmental" security is equivalent to Definition E.1, because the state of $\mathcal{E}$ can be encoded as part of $y$.

## E.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.[74] After this motivation, we generalise the result to our setting.

*Definition* E.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 $(x, w, \mathit{aux})$ (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 E.5** (Auxiliary input zero-knowledge)**.** *Let $(\mathcal{P}, \mathcal{V})$ be a proof 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.

Due to size-guarding, we can assume that $\mathcal{G}$ outputs $x$ with $|x| \leq \mathsf{poly}_{\mathcal{G}}(\kappa)$. Therefore, by assumption, Sim is *a priori* EPT with bound $\mathsf{poly}_{\mathsf{Sim}}(\mathsf{poly}_{\mathcal{G}}(\kappa))$, *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 $(x, w, \mathit{aux}) \leftarrow \mathcal{G}$. 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.

[74] These problems do no surface in [KL08; Gol10] since they define $\kappa = |x|$.

**Lemma E.6** (Sequential zero-knowledge)**.** *Let $(\mathcal{P}, \mathcal{V})$ be a size-guarded zero-knowledge proof system, with a simulator satisfying the conditions in Lemma E.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 E.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 $\mathsf{Sim}$. By Remarks E.2 and E.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 E.3, to the reader.

Since $\mathsf{Sim}$ 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 $\mathsf{Sim}$ and $\mathsf{RWS}$ for PPT distinguishers reduces, by a hybrid argument, to standard Q-IND. The hybrid distinguisher is efficient, because $\mathsf{Sim}$ is Goldreich-normal (and $\mathsf{RWS}$ normal). Consequently, $\mathsf{Sim}$ and $\mathsf{RWS}$ cannot be Q-IND. A contradiction. Hence, sequential Q-IND holds for $\mathsf{RWS}$ and $\mathsf{Sim}$. In particular, $\mathsf{rep}(\mathsf{Sim})$ satisfies all conditions in Lemma E.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 $\mathsf{RWS}$ and $\mathsf{Sim}$, *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 $\mathsf{Sim}$ cannot be too far from $\mathsf{RWS}$, and thus $\mathsf{Sim}$ is efficient whenever $\mathsf{RWS}$ is. This generalises the proof of Lemma E.6.

## E.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 $\mathsf{RWS}$ and $\mathsf{Sim}$. 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* E.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 $\Omega_\theta$, and where $D$ is the input space of a query distinguisher (as in Definition E.1). Let $\mathsf{A}$ 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 $\mathsf{A}$ (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 $\theta$ 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 $\mathsf{A}$ conditioned on $z$.
- $t_{\mathsf{A}+\mathcal{O}}(\kappa, z)$ like $t_{\mathsf{A}}$, but using $\mathsf{time}_{\mathsf{A}+\mathcal{O}}(\ldots)$.

We say that $\theta$ 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 $\kappa$: $t_\theta(\kappa, z) \leq \mathsf{poly}(\kappa) \cdot t_{\mathsf{A}+\mathcal{O}}(\kappa, z)$. Moreover, $\theta$ 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, $\theta$ 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* E.8 (Sketched application of runtime estimators). Consider a simulator Sim and its rewinding strategy RWS. If $T = \mathsf{time}_{\mathsf{RWS}+\mathcal{V}^*}(\mathsf{RWS}^{\mathcal{V}^*})$ is CEPT, then the (output of the) runtime estimate $\theta$ is CEPT if it lower-bounds $T$. If $\theta$ upper-bounds $S = \mathsf{time}_{\mathsf{Sim}+\mathcal{V}^*}(\mathsf{RWS}^{\mathcal{V}^*})$, then $S$ is CEPT if (the output of) $\theta$ is. Since $\theta$ only depends on the information available to a Q-IND adversary, assuming RWS and Sim are Q-IND, the runtime bound provided by $\theta$ only changes negligibly, hence if $T$ is CEPT, so is $S$. This provides a central link between the runtime RWS and Sim.

*Remark* E.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 $\theta$ is as simple as the total number of queries. Without size-guards, the behaviour is more fickle.

## E.4. Efficiency from query-truncation

We already saw in Lemmas E.5 and E.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 E.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* E.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 $\mathsf{poly}_0$ there is a polynomial $\mathsf{poly}_1$, 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 $\mathsf{poly}_0$. For zero-knowledge, the input space is $\mathcal{R} \times \{0,1\}^*$.

*Example* E.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* E.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.

## E.5. Query-benign simulators

Now, we bring together our definitions to define an alternative of benign, which we call query-benign.

*Definition* E.13 (Query-benign simulator). Let $(\mathcal{P}, \mathcal{V})$ be a proof 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*;
(2) for all a priori PPT adversaries $(\mathcal{G}, \mathcal{V}^*)$, 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 E.6, we directly show sequential zero-knowledge.

**Lemma E.14** (Query-benign implies sequential zero-knowledge)**.** *Suppose $(\mathcal{P}, \mathcal{V})$ is a proof 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.

**Step 1 (Replacing RWS):** As in Lemma 6.4, using $\mathsf{rep}(\mathsf{RWS}(\cdot))$ instead of $\mathsf{rep}(\langle\mathcal{P}, \cdot\rangle)$ in the sequential zero-knowledge experiment is still CEPT.

**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 $\mathsf{B}_0$ is the poly-query truncation of B. Since $\mathsf{B}_0$ 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 E.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.[75] Importantly, $\mathcal{E}$ only reads a strict polynomial prefix of the full (sequential) query sequence.

If we replace Sim by a truncation $\mathsf{Sim}_0$, which stops after the $\mathsf{poly}_{\mathcal{E}}$ queries, we know that $\mathsf{Sim}_0$ 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).

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 $\mathsf{Sim}_0$ and Sim. For symmetry, let $\mathsf{RWS}_0$ be defined analogously to $\mathsf{Sim}_0$. (Formally, we could use RWS, since there are no efficiency problems with RWS.)

Now, we can use that $\mathsf{Sim}_0$ is Goldreich-normal, to show via a hybrid argument as in Lemma E.6 that if $\mathcal{E}$ can distinguish $\mathsf{RWS}_0$ and $\mathsf{Sim}_0$ for "sequential Q-IND", there is a Q-IND distinguisher $\mathcal{D}$ for $\mathsf{RWS}_0$ and $\mathsf{Sim}_0$. And hence, there is a Q-IND distinguisher for RWS and Sim (since the constructed hybrid distinguisher $\mathcal{D}$ also sees no difference between $\mathsf{Sim}_0$ and Sim (resp. $\mathsf{RWS}_0$ and RWS)). Thus, "sequential Q-IND" holds.

**Step 4 (Sim is CEPT if RWS is):** Now we make use of the runtime estimator $\theta$. More precisely, we extend $\theta$ to the sequential setting by applying the underlying $\theta$ for each invocation separately, and

[75] This is a technical requirement. Depending on the machine model, $\mathcal{E}$ may have random access to the output. That would make our later argumentation 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.

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 $\theta$ (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 $z_{\mathsf{RWS}}$ 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 $\theta$ 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. (Note that, even if we know Sim is CEPT, to apply a hybrid argument to show good output quality assuming only zero-knowledge, we have to argue that the hybrid distinguisher is CEPT.) □

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.

# F. Additional discussions

## F.1. Levin's relaxation and CEPT

We noted in Footnote 4, that $\sum_{n=1}^{\infty} n^{1+\varepsilon} < \infty$ for $\varepsilon > 0$ gives rise to distribution $Z_{1+\varepsilon}$ over $\mathbb{N}$ via normalising 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 $Y_k = X_{(\cdot \le k) \mapsto 0} = (X^{\le k})_{\mathtt{timeout} \mapsto 0}$. It follows immediately that $\mathbb{E}(Y_k) = \mathbb{E}(X_{(\cdot \le k) \mapsto 0}) \ge \frac{1}{2}k^2$ for any $k \in \mathbb{N}$. Thus, for any superpolynomial cutoff $K$, we find $\mathbb{E}(Y_K) \ge \frac{1}{2}K^2$ is superpolynomial, and as a consequence, there is no superpolynomial cutoff which makes $X$ EPT. (Here, we interpret $X$ (and $Y_K$) as a constant family of runtimes, i.e. $X_\kappa = X$ for all $\kappa$.)

Formally, CEPT uses $\nu$-quantile cutoffs. But it is easy to see that any $\nu$-quantile cutoff for negligible $\nu$ corresponds to a superpolynomial truncation: $\nu$ corresponds to to some truncation $k$. If $k$ were polynomial, then (due to "fat tails") $\nu$ must also be polynomial.

All in all, the runtime distribution $X_\kappa = X$ is allowed by Levin's relaxation, but is not CEPT.

## F.2. 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 `win`/`lose` (or 1/0). A hardness assumption is an upper bound for $\mathbb{P}(\mathsf{out}_{\mathcal{C}}\langle \mathcal{A}, \mathcal{C}\rangle = \mathtt{win})$, e.g. $\mathsf{negl}$ for one-wayness or $\frac{1}{2} + \mathsf{negl}$ for IND-CPA, where $\mathsf{negl}$ 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 `win`, then $\mathcal{O}_b$ sends $b$ to the adversary. If the verdict is `lose`, then $\mathcal{O}_b$ sends $\bot$ instead. A straightforward calculation shows

$$\mathbb{P}(\mathsf{out}_{\mathcal{C}}\langle \mathcal{A}, \mathcal{C}\rangle = \mathtt{win}) = \mathbb{P}(\mathcal{D}^{\mathcal{O}_1} = 1) - \mathbb{P}(\mathcal{D}^{\mathcal{O}_0} = 1)$$

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 $\bot$ 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 formalisations 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).

## F.3. Relaxing benign

The definition of benign simulation is very restrictive and does not mirror the actual security proof for $\mathrm{G3C}_{\mathrm{GK}}$ very well. For simplicity, we conflated all application and uses of assumptions into one single game "hop", the switch from $\mathrm{R}^{\mathcal{C}_0}$ to $\mathrm{R}^{\mathcal{C}_1}$. We defined iterated benign as a relaxation, but even that does not mirror the proof that well. There are only multiple reduction steps, but not multiple rewinding strategies.

Fortunately, it is easy to see that one may allow constantly many "reduction hops" $\mathrm{R}_i$ and simple assumptions $\mathcal{C}^i_{0/1}$. The relaxation of benign with multi-hop reduction is as follows: The reduction R is an algorithm which gets as additional input the index $i$. We require that $\mathrm{R}_i^{\mathcal{C}^i_1} \equiv \mathrm{R}_{i+1}^{\mathcal{C}^{i+1}_0}$ and that $\mathrm{R}_{i+1}^{\mathcal{C}^{i+1}_0}$ is efficient relative to $\mathrm{R}_i^{\mathcal{C}^i_1}$ (with polynomial runtime tightness). Lastly, we require $\mathsf{RWS} \equiv \mathrm{R}_0^{\mathcal{C}^0_0}$ and $\mathrm{R}_k^{\mathcal{C}^k_1} \equiv \mathsf{Sim}$ and the usual relative efficiency. Since each reduction step needs a relative efficiency bound, we can only allow constantly many "hops", i.e. $k$ is a constant.

As a matter of fact, this relaxation brings nothing new. Since the effects of RWS can technically be absorbed by relative efficiency, and the trivial rewinding strategy can be used in iterated benignness, this relaxation of benign is equivalent to iterated benign.

That being said, one can prove polynomial hybrid steps secure for a (natural) class of more restricted reductions. However, it appears incurred technical complexity is better hidden in the simple assumptions, e.g. using the left-right hiding game instead of the usual single-challenge hiding game, so that a constant number "game hops" suffice.

## F.4. Size-guards

We recall the need for size-guards, discuss two approaches to generalising 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).[76]

**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 Remark 5.3. 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.

[76] 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.)

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 $\mathsf{A}$ has a priori *strict (probabilistic) polynomial space* (SPS) (in analogy to PPT) if there exists a polynomial $\mathsf{poly}(\kappa)$ 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 $\mathsf{A}$ at all. Thus, for a priori SPS adversaries, size-guarded security and normal security are equivalent.

For "classical" SPS, the space of $\mathsf{A}$ may depend on the input size, i.e. $\mathsf{poly}(\kappa, |x|)$. All protocols of interest satisfy SPS. We find (classical, a posteriori and a priori) EPT SPS algorithms an appealing 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.5.

**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 Remark 5.3. 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.

## F.5. The trouble of a posteriori runtime and composition

To further motivate our choices and definitions, it is instrumental to discuss the interaction of virtuality, a posteriori runtime and designated adversaries at the example of sequential composition.

Our notion of (auxiliary input) zero-knowledge allows designated adversaries in the strongest sense. Namely, we only require that $(\mathcal{G}, \mathcal{V}^*, \mathcal{D})$ is efficient. As noted, we can equivalently reduce to $(\mathcal{G}, \mathcal{V}^*)$, since there is always an a priori PPT $\mathcal{D}$ if the adversary breaks zero-knowledge. The big problem for

sequential composition is that there is no efficiency guarantee for $\mathcal{V}^*$ alone. More to the point: When replacing $\mathcal{G}$ with $\mathcal{G}'$, it is unclear whether $(\mathcal{G}', \mathcal{V}^*)$ is still efficient. The standard hybrid proof for sequential composition, changes $\mathcal{G}$, which spells trouble.

Consider sequential repetition of a zero-knowledge proof. Using a hybrid argument which replaces $\mathcal{P}$ by $\mathsf{Sim}$ starting from the last repetition, there is no problem with $\mathcal{G}$ at the point of embedding. We started from the back, so "$\mathcal{G}_{i-1}$" for the $i$-th hybrid is distributed as in the real protocol. However, after embedding a challenge at $i$, the hybrid argument needs to "post-simulate", i.e. use $\mathsf{Sim}$ in the remaining iterations. And here, "$\mathcal{G}_j$" for $j > i$ is modified. Given only weak relative efficiency guarantees for $\mathsf{Sim}$, we do not know how to prove a superconstant number of repetitions are efficient. Such $\mathsf{Sim}$ only guarantees it is CEPT if $(\mathcal{G}, \mathcal{V}^*)$ is, but gives no concrete relation for the runtimes. If $\mathsf{Sim}$'s runtime depends in a problematic way on $\mathcal{G}$, e.g. exponentially growing constants, that would still be true. Such behaviour is implausible, in particular since $\mathsf{Sim}$ itself is independent of $\mathcal{G}$, so only the output distribution of $\mathcal{G}$ is relevant. Nevertheless, we could not prove superconstant repetition secure.

These problems appear for PPT and EPT simulation against a priori PPT adversaries as well. However, there simulators which are EPT when excluding the time spent in $\mathcal{V}^*$, are also EPT when including it. Thus, they are implicitly relatively efficient with runtime tightness. (See also Definition E.4.) Efficiency of sequential repetitions thus follows easily. To emulate this, we replace weak relative efficiency with (strong) relative efficiency with runtime tightness.

This is not yet enough for CEPT simulation. There, a core problem is that virtuality growth is *not unconditional*. Different adversaries spend different effort trying to distinguish simulation from reality (e.g. brute-force commitments), and there is always a stronger one (which runs longer). Thus, there is no universal (tightness) bound for the virtuality of a simulation. Indeed, if the hardness assumptions were false, then an adversary can distinguish reality from simulation, and make the runtime explode.

Our solution to this problem is to provide an explicit "common anchor", the simple assumption $\mathcal{C}_0 \overset{c}{\approx} \mathcal{C}_1$ in a benign simulator, to which the change in virtuality can be reduced to. Hence, the (uniform) reduction R, independent of the adversary, is our computational analogue to the runtime tightness bounds. (For query-benign simulators, the common anchor is the query-indistinguishability.)

One may try to escape the problems by restricting the adversary more, and giving up full-fledged designated adversaries. However, considering that not even "*expected polynomial time in any interaction*" gives runtime guarantees for rewinding simulators [KL08], this essentially leads to [Gol10].

Ultimately, it seems that composition results, such as sequential, parallel and concurrent composition, do not follow trivially from "auxiliary input" notions in our setting. For parallel and concurrent composition, this is well-known even for a priori PPT adversaries (but for different reasons). In our setting, this also extends to sequential composition.

## F.6. Absolute notions of relative efficiency

In Section 4.4, 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 $\mathsf{A}$ is efficient, so is $\mathsf{A}$. 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* F.1 (Weak absolute relative efficiency). Let $\mathcal{T}$ be a runtime class and $\mathsf{A}$ be an algorithm. Then $\mathsf{A}$ 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 $\mathsf{time}_{\mathsf{A}+\mathcal{O}}(\langle \mathsf{A}, \mathcal{O} \rangle) \in \mathcal{T}$.

The problems with weak relative efficiency apply in this absolute setting as well, i.e. weakly ar-eff seems too weak for general use. Thus, we resort to following stronger definition.

*Definition* F.2 (Absolute relative efficiency). Let $\mathsf{A}$ be an algorithm and $\mathcal{O}$ an oracle. Then $\mathsf{A}$ 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 amortised* since $\mathcal{O}$ is timeful and can abort at any time. Indeed, allowing a timeful $\mathcal{O}$ is one of the main reasons for the brittleness of ar-eff. Alas, conditional notions of relative efficiency are much less useful.

It is usually trivial to check that an algorithm is ar-eff. Namely, one verifies the stronger claim, that it is *PPT (resp. EPT) per activation.* As a rule of thumb, the *non-adversarial parties 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* F.3 (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. on a bb-rw simulator. It is entirely independent of a given adversary.

*Example* F.4 (G3C is not ar-eff under size-guarding). 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 $\kappa$ and the prover is not ar-eff.[77]

This problem is mitigated by size-guards. For a variation of $\mathrm{G3C}_{\mathrm{GK}}$ with graph hamiltonicity, this problem would not occur as $\kappa$ parallel repetitions suffice, independent of $G$. (Modulo technical complications.)

All in all, Example F.4 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.

## F.7. Standard non-uniformity

Our proposed notion of non-uniform security is still *probabilistic*. More concretely, we propose in Appendix A.5 to give a probabilistic machine tape-like access to a (possibly 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* F.5. For oracle-distinguishing, we saw in Corollary 4.4, 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 $\mathcal{D}$.

[77] This can be seen as a technical artifact 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 $m$, $\mathsf{A}$ 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 $\mathsf{A}$ as efficient in any sense.

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 Remark 5.3. Thus, an equivalence with the standard setting of non-uniform security is only guaranteed if security with size-guarding is considered, see Remark 5.3. While size-guarded security is a natural notion, imposing it when it is not needed is wasteful.

## F.8. 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 4-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 $k \sim \mathsf{Geo}(p)$ rewinds to obtains a second convincing answer. Let $R$ bet the number of rewinds. Then the expectation of $R$ is

$$\|R\|_1 = (1-p)\cdot 1 + p\frac{1-p}{p} \leq 2.$$

If we consider $\|R\|_2$, then we find the

$$\|R\|_2 \geq p\frac{1-p}{p^2} \geq \frac{1}{p}.$$

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,

$$(f(\kappa))_\kappa = (0,1,2,1,3,1,2,1,4,1,2,\ldots)$$

It is easy to see that $p$ is not negligible. However, for any polynomial $\mathsf{poly}$, 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. $q = \infty$).

## F.9. 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,[78] 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 $\sigma$-algebras.)

The image of $Z_i$ lies is in a countable space, which we equip with the discrete $\sigma$-algebra. The sample space of process $Z(i, \omega) = Z_i(\omega)$ is given the induced $\sigma$-algebra, but is not countable anymore. (Recall that the $\sigma$-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 $\mathsf{A}$, $\mathsf{B}$ 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 $\mathsf{A}$, $\mathsf{B}$.)

**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 $\nu$-quantile cutoffs) is not immediately possible. One can introduce some "irrelevant action", e.g. a zero-th 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 $\sigma$-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.)

[78] In particular, a *timeful* system must have measurable purported runtime.

## F.10. Infinitely-often efficiency

We comment on yet another notion of efficiency, *infinitely-often* efficient algorithms. The definition is as one would expect: An algorithm A is efficient, if there is an infinite increasing sequence $(\kappa_0, \kappa_1, \ldots)$ of security parameters for which A is efficient. (Recall that we consider closed systems, i.e. A has no inputs. But this easily extends to common notions of asymptotic efficiency.)

The advantage of this approach is that constructing, or rather describing, reductions becomes simpler. Many reduction rely on advantage $\varepsilon$ which is bounded below by $\varepsilon' = 1/\mathsf{poly}$ *infinitely often*, and have runtime polynomial in $\varepsilon'^{-1}$. With infinitely-often efficiency, runtimes of the form $\mathsf{poly}(\varepsilon^{-1})$ are efficient as well, so the infinitely-often polynomial lower bound becomes superfluous.

This "simplification" has one big problem: Non-negligible advantage of infinitely-often adversaries is meaningless. The adversary could brute-force for even $\kappa$, and immediately halt for odd, thus satisfying infinitely-often efficiency and having non-negligible advantage. Hence, every property must now be phrased w.r.t. to (any) infinite subsequence $(\kappa_0, \ldots)$ for which an algorithm is efficient.

In other words: It is possible to get rid of the (repeated) specification of "infinitely often" properties by switching to "infinitely-often" notions, which incorporate this property by definition.

We are not aware of an example, where an infinitely-often efficient reduction works, but there is no standard efficient reduction. On the other hand, there are results, which construct "infinitely-often objects" by reduction. All in all, it appears that being explicit about infinitely-ofteness has more upsides than downsides.

## F.11. Musings on runtime classes

Instead of dealing with runtime *distributions* only, a runtime class should deal with *random variables*. For this, fix some (family of) universal probabilistic space(s) $\Omega_\kappa$ and redefine runtime classes as follows:

*Definition* F.6. 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:[79] 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* F.7. A "good" runtime class $\mathcal{T}$ should satisfy following properties:

**Constants:** The constant 0 and constant 1 runtime are in $\mathcal{T}$.[80]

**Closed under domination:** For any $T \in \mathcal{T}$, all its dominated runtimes $S$ are contained in $\mathcal{T}$ as well, i.e. $\forall S \forall T \in \mathcal{T}\colon S \overset{\mathcal{D}}{\leq} T \implies S \in \mathcal{T}$. Recall that $S \overset{\mathcal{D}}{\leq} T$ is defined pointwise w.r.t. $\kappa$, i.e. $\forall \kappa\colon S_\kappa \overset{\mathcal{D}}{\leq} T_\kappa$, 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 $S_\kappa := \max\{T_1, \ldots, T_\kappa\}$. Then $S \in \mathcal{T}$. This statement is of nonsensical if $\Omega_i \neq \Omega_j$. Therefore, it is a statement about distributions.

*Remark* F.8. 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

[79] Perhaps even further, a runtime class should be a function, mapping a probability space $\Omega$ to a runtime class $\mathcal{T}(\Omega)$ with suitable compatibility rules.

[80] 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.

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 $\kappa$ only increases admitted runtimes, e.g. efficiency of constant runtimes can be tested for $\kappa = 1$.

*Remark* F.9 (Weak composition). We lack a generalisation 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.[81]

*Remark* F.10. 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 F.7, 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.

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* F.11 (A candidate for $\mathsf{RTC}_\infty(\mathcal{T})$). In the proof of Corollary C.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 C.31 also requires "weak composability" (in the sense of Remark F.9), 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 C.31, and to "equivalence of statistical and computational indistinguishability". Indeed, the relation of these three properties appears to be of central importance. Characterising 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 summarise, 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.

[81] 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.

# Contents